Schulinternes Curriculum für die
gymnasiale Oberstufe am
Gymnasium Paulinum

Informatik

Stand: August 2022

Inhaltsverzeichnis

1 Die Fachgruppe Informatik am Gymnasium Paulinum 2
2 Entscheidungen zum Unterricht 4
2.1 Unterrichtsvorhaben 4
2.1.1 Übersichtsraster der Unterrichtsvorhaben.....6
2.1.2 Konkretisierte Unterrichtsvorhaben.....9
2.2 Grundsätze der fachmethodischen und fachdidaktischen Arbeiten.....57
2.3 Grundsätze der Leistungsbewertung und Leistungsrückmeldung.....58
2.3.1 Verbindliche Absprachen im Beurteilungsbereich der „Schriftlichen Leistungen“.....58
2.3.2 Verbindliche Absprachen im Beurteilungsbereich „Sonstige Leistungen im Unterricht“ 60
2.3.3 Grundsätze zur Leistungsrückmeldung und Beratung.....61
2.4 Lehr- und Lernmittel.....62
3 Entscheidungen zu fach- und unterrichtsübergreifenden Fragestellungen.....62
4 Qualitätssicherung und Evaluation 65

1 Die Fachgruppe Informatik am Gymnasium Paulinum

Das Gymnasium Paulinum, eine der ältesten Schulen Deutschlands, liegt im Herzen Münsters. Der Einzugsbereich der Schule ist groß und umfasst das gesamte Stadtgebiet sowie das eher ländlich geprägte Umland. Das Paulinum ist ein durchgängig vierzügiges Gymnasium mit etwa 1000 Schüler:innen und 75 Lehrer:innen.

In seinem Schulprogramm betont das Paulinum die Bedeutung ganzheitlicher Bildung, die „Wertvorstellungen der europäischen Tradition mit intellektueller Anstrengungsbereitschaft und Aufgeschlossenheit für die Belange einer sich wandelnden Welt“1 verbindet. Kaum ein anderes Ereignis befördert diesen anhaltenden, globalen gesellschaftlichen Wandel in gleichem Maße wie die voranschreitende Digitalisierung und Computerisierung unserer Welt.

Es ist daher eine zentrale Aufgabe der Schule, den Schüler:innen die Fertigkeiten zu vermitteln, um sich souverän in dieser Welt zu bewegen: „Die Schüler:innen erwerben im Laufe der Sekundarstufe I eine umfassende Kompetenz im Umgang mit digitalen Medien in allen Aspekten des Methoden- und Medienkonzeptes. Die zu erwerbenden Kompetenzen werden fachspezifisch konkretisiert, curricular verankert und ermöglichen aufbauendes Lernen.“2

In der gymnasialen Oberstufe wird auf diesen Grundlagen aufgebaut, während der Fokus auf einer zunehmend vertieften und fachwissenschaftlich orientierten Behandlung informatischer Kernkonzepte liegt. Der Informatikunterricht der Sekundarstufe II wird am Gymnasium Paulinum derzeit in der Einführungsphase als Grundkurs sowie in der Qualifikationsphase zur Wahl als Grund- oder Leistungskurs angeboten und entsprechend jeweils dreistündig im GK oder fünfstündig im LK (zu je 45 Minuten) unterrichtet.

Der Informatikunterricht in der gymnasialen Oberstufe ist am Gymnasium Paulinum als Einstiegsunterricht konzipiert, der von Schüler:innen angewählt werden kann, unabhängig davon, ob sie in der Sekundarstufe I im Wahlpflichtbereich II das Fach Informatik belegt haben. In den Kursen der Einführungsphase sind daher keine fachlichen Vorkenntnisse erforderlich und es werden alle Grundlagen gelegt, die den Schüler:innen die weitere erfolgreiche Teilnahme am Kurs ermöglichen.

Die schulinternen Curricula für den Wahlpflichtbereich II und die gymnasiale Oberstufe sind dabei in solcher Weise aufeinander abgestimmt, dass inhaltliche Dopplungen vermieden werden, damit auch Schüler:innen, die neben dem Pflichtunterricht in der

1 Schulprogramm des Gymnasium Paulinum vom 28.9.2021, S. 1.
2 A. a. O., S. 5

Jahrgangsstufe 6 auch bereits zwei Jahre Informatikunterricht im Wahlpflichtbereich hatten, von Beginn an Neues lernen.

Der Informatikunterricht am Gymnasium Paulinum orientiert sich am Kernlehrplan des Landes Nordrhein-Westfalen für das Fach Informatik in der gymnasialen Oberstufe sowie an den Bildungsstandards Informatik für die Sekundarstufe II der Gesellschaft für Informatik (GI). Schwerpunkte des Unterrichts sind die Inhaltsfelder „Daten und ihre Strukturierung“, „Formale Sprachen und Automaten“, „Algorithmen“, „Informatiksysteme“ sowie „Informatik, Mensch und Gesellschaft“. Die inhaltliche Gestaltung des Unterrichts ist von Phasen der kooperativen Teamarbeit geprägt, in der die Schüler:innen Kompetenzen der Kommunikation, des selbstständigen Lernens und gemeinschaftlichen Problemlösens entwickeln.

Für den Unterricht stehen am Gymnasium Paulinum zwei Computerräume zur Verfügung. Auf den Rechnern ist die für den Unterricht relevante Software installiert, darunter neben üblichen Textverarbeitungs- und Tabellenkalkulationsprogrammen ist das in der gymnasialen Oberstufe vor allem Software zur Entwicklung eigener Programme. Entsprechend den aktuellen Abiturvorgaben des Landes verwenden wir am Paulinum dazu die objektorientierte Programmiersprache *Java*. Dazu wird die didaktische Entwicklungsumgebung *BlueJ* verwendet.

Die Lehrkräfte sowie die Schüler:innen verfügen über individuelle Zugangsdaten und können somit alle Rechner zur Arbeit verwenden. Der technische Support wird durch den städtischen IT-Dienstleister *citeq* übernommen. Zudem besteht am Paulinum ab der Jahrgangsstufe 7 eine 1:1-Ausstattung mit iPads, auf denen zwar bedingt durch das Betriebssystem nur eingeschränkt programmiert werden kann, die aber ansonsten in vielen Unterrichtseinheiten produktiven Einsatz finden.

Die Fachgruppe Informatik am Gymnasium Paulinum besteht derzeit aus fünf Lehrkräften: Herrn Spallek, Herrn Hendrik Becker, Herrn Christoph Becker, Frau Plieth und Herrn Vejvoda. Alle entwickelten Unterrichtsmaterialien sowie Dokumente, die den Informatikunterricht am Paulinum betreffen, werden von den Mitgliedern der Fachgruppe digital über ein gemeinsames Netzwerkverzeichnis geteilt. Durch einen häufigen, direkten Austausch zwischen den Lehrkräften wird sichergestellt, dass Material nach gemeinsamer Absprache und einvernehmlich entwickelt wird.

Die kollegiale Entwicklung von Unterrichtsvorhaben sowie die gemeinsame Evaluation von Lehr- und Lernprozessen einschließlich der Modifikation dieses Lehrplans durch die Fachgruppe Informatik stellen einen wesentlichen Beitrag zur Qualitätssicherung und -entwicklung des Unterrichts dar.

2 Entscheidungen zum Unterricht

2.1 Unterrichtsvorhaben

Die Darstellung der Unterrichtsvorhaben im schulinternen Lehrplan besitzt den Anspruch, die im Kernlehrplan aufgeführten Kompetenzen vollständig abzudecken. Die entsprechende Umsetzung erfolgt auf zwei Ebenen: der Übersichts- und der Konkretisierungsebene.

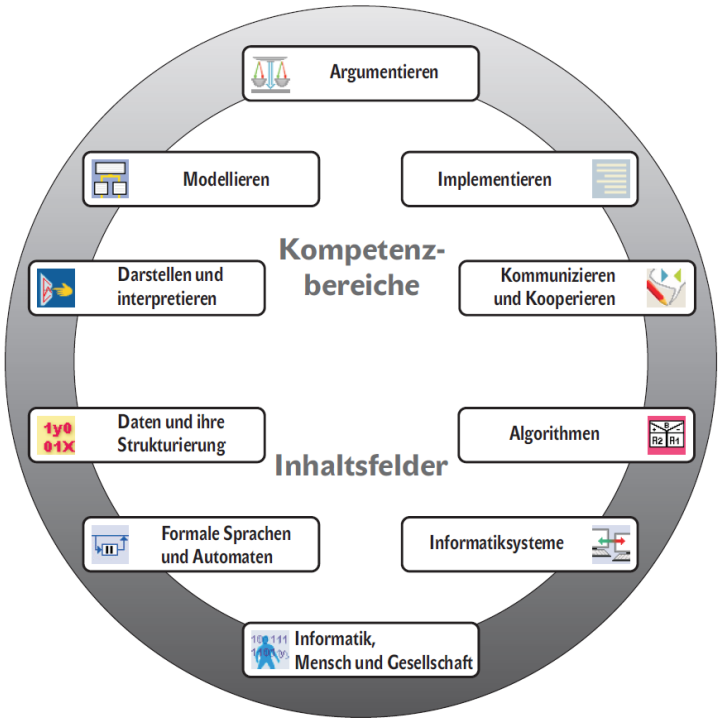

Abbildung 1: Die Kompetenzbereiche und Inhaltsfelder in Informatik in der Sekundarstufe II gemäß dem Kernlehrplan Informatik für die gymnasiale Oberstufe in NRW

Im folgenden „Übersichtsraster der Unterrichtsvorhaben“ (Kapitel 2.1.1) wird die für alle Lehrerinnen und Lehrer verbindliche Verteilung der Unterrichtsvorhaben dargestellt. Das Übersichtsraster dient dazu, den Kolleg:innen einen schnellen Überblick über die Zuordnung der Unterrichtsvorhaben zu den einzelnen Jahrgangsstufen sowie den im Kernlehrplan genannten Kompetenzen der Inhaltsfelder und Kompetenzbereiche sowie den inhaltlichen Schwerpunkten zu verschaffen.

Um Klarheit für die Lehrkräfte herzustellen und die Übersichtlichkeit zu gewährleisten, werden in der Kategorie „Kompetenzen“ an dieser Stelle nur die übergeordneten Kompetenzerwartungen ausgewiesen, während die konkretisierten Kompetenzerwartungen erst auf der Ebene konkretisierter Unterrichtsvorhaben Berücksichtigung finden. Der

ausgewiesene Zeitbedarf versteht sich als grobe Orientierungsgröße, die nach Bedarf über- und unterschritten werden kann. Um Spielraum für Vertiefungen, individuelle Förderung, besondere Schülerinteressen, aktuelle Themen bzw. die Erfordernisse anderer besonderer Ereignisse (z. B. Praktika, Klassenfahrten, o. ä.) zu erhalten, wurden im Rahmen dieses schulinternen Lehrplans nur ca. 75 % der Bruttounterrichtszeit verplant.

Während der Fachkonferenzbeschluss zum „Übersichtsraster der Unterrichtsvorhaben“ zur Gewährleistung vergleichbarer Standards sowie zur Absicherung von Lerngruppenübertritten und Lehrkraftwechseln für alle Mitglieder der Fachkonferenz Bindekraft haben soll, besitzen die didaktischen Hinweise der exemplarischen Ausweisung „konkretisierter Unterrichtsvorhaben“ (Kapitel 2.1.2) lediglich empfehlenden Charakter. Referendarinnen und Referendaren sowie neuen Kolleg:innen dienen diese vor allem zur standardbezogenen Orientierung in der neuen Schule, aber auch zur Verdeutlichung von unterrichtsbezogenen fachgruppeninternen Absprachen zu didaktisch-methodischen Vorgängen, fachübergreifenden Kooperationen, Lernmitteln und -orten sowie vorgesehenen Leistungsüberprüfungen, die im Einzelnen auch den Kapiteln 2.2 bis 2.4 zu entnehmen sind.

Abweichungen von den vorgeschlagenen Vorgehensweisen bezüglich der konkretisierten Unterrichtsvorhaben sind im Rahmen der pädagogischen Freiheit der Lehrkräfte jederzeit möglich. Sicherzustellen bleibt allerdings auch hier, dass im Rahmen der Umsetzung der Unterrichtsvorhaben insgesamt alle ausgewiesenen inhalts- und prozessbezogenen Kompetenzen Berücksichtigung finden.

Da das Fach Informatik in der gymnasialen Oberstufe derzeit am Paulinum ausschließlich im Grundkurs unterrichtet wird, werden die Leistungskursvorhaben an dieser Stelle nicht ausgeführt.

2.1.1 Übersichtsraster der Unterrichtsvorhaben

2.1.1.1 Einführungsphase

<table>
<tr><th colspan="2">Einführungsphase</th></tr>
<tr><td>Unterrichtsvorhaben EF-I

Thema: Einführung in die Nutzung von Informatiksystemen und in grundlegende Begrifflichkeiten

Schwerpunkte: Einsatz von Informatiksystemen; Dateisysteme, Einzelrechner
Inhaltsfelder:
• Informatiksysteme
• Informatik, Mensch und Gesellschaft
Zentrale Kompetenzen:
• Darstellen und Interpretieren
• Kommunizieren und Kooperieren
• Argumentieren
Zeitbedarf: 9 Stunden</td><td>Unterrichtsvorhaben EF-II

Thema: Grundlagen der objektorientierten Analyse, Modellierung und Implementierung anhand von statischen Grafikszenen

Schwerpunkte: Objekte und Klassen; Syntax und Semantik einer Programmiersprache
Inhaltsfelder:
• Daten und ihre Strukturierung
• Formale Sprachen und Automaten
Zentrale Kompetenzen:
• Modellieren
• Implementieren
Zeitbedarf: 15 Stunden</td></tr>
<tr><td>Unterrichtsvorhaben EF-III

Thema: Algorithmische Grundstrukturen in Java anhand einfacher Animationen

Schwerpunkte: Objekte und Klassen; Analyse, Entwurf, Implementierung einfacher Algorithmen; Syntax und Semantik
Inhaltsfelder:
• Daten und ihre Strukturierung
• Algorithmen
Zentrale Kompetenzen:
• Modellieren
• Implementieren
• Kommunizieren und Kooperieren
Zeitbedarf: 21 Stunden</td><td>Unterrichtsvorhaben EF-IV

Thema: Modellierung und Implementierung von Klassen und Objektbeziehungen anhand von grafischen Spielen und Simulationen

Schwerpunkte: Objekte und Klassen; Analyse, Entwurf, Implementierung einfacher Algorithmen; Syntax und Semantik
Inhaltsfelder:
• Daten und ihre Strukturierung
• Algorithmen
Zentrale Kompetenzen:
• Argumentieren
• Modellieren
• Implementieren
• Darstellen und Interpretieren
Zeitbedarf: 18 Stunden</td></tr>
<tr><td>Unterrichtsvorhaben EF-V

Thema: Such- und Sortieralgorithmen anhand kontextbezogener Beispiele

Schwerpunkte: Algorithmen zum Suchen und Sortieren; Analyse, Entwurf und Implementierung einfacher Algorithmen
Inhaltsfelder:
• Algorithmen
Zentrale Kompetenzen:
• Argumentieren
• Modellieren
• Kommunizieren und Kooperieren
• Darstellen und Interpretieren
Zeitbedarf: 15 Stunden</td><td>Unterrichtsvorhaben EF-VI

Thema: Geschichte der digitalen Datenverarbeitung und die Grundlagen des Datenschutzes

Schwerpunkte: Wirkungen der Automatisierung; Geschichte der automatischen Datenverarbeitung; Digitalisierung
Inhaltsfelder:
• Informatiksysteme
• Informatik, Mensch und Gesellschaft
Zentrale Kompetenzen:
• Argumentieren
• Modellieren
• Darstellen und Interpretieren
Zeitbedarf: 12 Stunden</td></tr>
<tr><td colspan="2">Summe: 90 Stunden</td></tr>
</table>

2.1.1.2 Qualifikationsphase 1 (GK und LK)

<table>
<tr><th colspan="2">Qualifikationsphase 1 (Grundkurs/Leistungskurs)</th></tr>
<tr><td>Unterrichtsvorhaben Q1-I

Thema: Wiederholung der objektorientierten Modellierung und Programmierung auch unter Berücksichtigung der Gestaltung einer Benutzungsoberfläche

Schwerpunkte: Objekte und Klassen; Analyse, Entwurf und Implementierung von Algorithmen; Syntax und Semantik einer Programmiersprache
Inhaltsfelder:
• Daten und ihre Strukturierung
• Algorithmen
• Formale Sprachen und Automaten
Zentrale Kompetenzen:
• Modellieren und Implementieren
• Darstellen und Interpretieren
• Kommunizieren und Kooperieren
Zeitbedarf: 15/25 Stunden</td><td>Unterrichtsvorhaben Q1-II

Thema: Modellierung und Implementierung von dynamischen linearen Datenstrukturen sowie von Anwendungen unter Verwendung dynamischer, linearer Datenstrukturen

Schwerpunkte: Objekte und Klassen; Analyse, Entwurf und Implementierung von Algorithmen; Syntax und Semantik einer Programmiersprache
Inhaltsfelder:
• Daten und ihre Strukturierung
• Formale Sprachen und Automaten
Zentrale Kompetenzen:
• Argumentieren
• Modellieren und Implementieren
• Darstellen und Interpretieren
Zeitbedarf: 21/30 Stunden</td></tr>
<tr><td>Unterrichtsvorhaben Q1-III

Thema: Suchen und Sortieren auf linearen Datenstrukturen sowie Modellierung, Implementierung und Komplexitätsanalyse von Such- und Sortieralgorithmen

Schwerpunkte: Analyse, Entwurf und Implementierung von Algorithmen; Algorithmen in ausgewählten Kontexten; Syntax und Semantik einer Programmiersprache
Inhaltsfelder:
• Formale Sprachen und Automaten
• Algorithmen
Zentrale Kompetenzen:
• Argumentieren
• Modellieren
• Implementieren
• Darstellen und Interpretieren
• Kommunizieren und Kooperieren
Zeitbedarf: 18/30 Stunden</td><td>Unterrichtsvorhaben Q1-IV.1

Thema: Modellierung, inkl. Implementierung, und Nutzung relationaler Datenbanken in Anwendungskontexten

Schwerpunkte: Datenbanken; Algorithmen in ausgewählten informatischen Kontexten; Syntax und Semantik einer Programmiersprache; Sicherheit
Inhaltsfelder:
• Daten und ihre Strukturierung
• Algorithmen
• Formale Sprachen und Automaten
• Informatik, Mensch und Gesellschaft
Zentrale Kompetenzen:
• Argumentieren
• Modellieren
• Implementieren
• Darstellen und Interpretieren
Zeitbedarf: 24/30 Stunden</td></tr>
<tr><td>Unterrichtsvorhaben Q1-IV.2

Thema: Projektorientierte Softwareentwicklung am Beispiel einer Anwendung mit Datenbankanbindung

Schwerpunkte: Objekte und Klassen; Analyse, Entwurf und Implementierung von Algorithmen; Datenbanken; Algorithmen in ausgewählten inf. Kontexten; Sicherheit
Inhaltsfelder:
• Daten und ihre Strukturierung
• Algorithmen
• Informatik, Mensch und Gesellschaft
Zentrale Kompetenzen:
• Argumentieren
• Modellieren
• Implementieren
• Kommunizieren und Kooperieren
• Darstellen und Interpretieren
Zeitbedarf: 15 Stunden</td><td>Unterrichtsvorhaben Q1-V

Thema: Grundlagen der Netzwerkkommunikation und Sicherheit und Datenschutz in Netzwerken sowie Modellierung und Implementierung von Client-Server-Anwendungen in kontextbezogenen Problemstellungen

Schwerpunkte: Algorithmen; Einzelrechner und Rechnernetzwerke; Nutzung von Informatiksystemen, Wirkung der Automatisierung
Inhaltsfelder:
• Algorithmen
• Informatiksysteme
• Informatik, Mensch und Gesellschaft
Zentrale Kompetenzen:
• Modellieren und Implementieren
• Darstellen und Interpretieren
• Kommunizieren und Kooperieren
Zeitbedarf: 12/20 Stunden</td></tr>
<tr><td colspan="2">Summe: 90/150 Stunden</td></tr>
</table>

2.1.1.3 Qualifikationsphase 2 (GK/LK)

<table>
<tr><th colspan="2">Qualifikationsphase 2 (Grundkurs/ Leistungskurs)</th></tr>
<tr><td>Unterrichtsvorhaben Q2-I

Thema: Modellierung und Implementierung von dynamischen, nicht-linearen Datenstrukturen sowie von Anwendungen mit dynamischen, nichtlinearen Datenstrukturen

Schwerpunkte: Objekte und Klassen; Analyse, Entwurf und Implementierung von Algorithmen; Algorithmen in ausgewählten informatischen Kontexten; Syntax, Semantik einer Programmiersprache
Inhaltsfelder:
• Daten und ihre Strukturierung
• Algorithmen
• Formale Sprachen und Automaten
Zentrale Kompetenzen:
• Argumentieren
• Modellieren
• Implementieren
• Darstellen und Interpretieren
• Kommunizieren und Kooperieren
Zeitbedarf: 24/35 Stunden</td><td>Unterrichtsvorhaben Q2-II

Thema: Endliche Automaten, Kellerautomaten und formale Sprachen sowie Modellierung und Implementierung eines Parsers zu einer formalen Sprache

Schwerpunkte: Endliche Automaten; Grammatiken regulärer und kontextfreier Sprachen; Möglichkeiten und Grenzen von Automaten und formalen Sprachen
Inhaltsfelder:
• Endliche Automaten und formale Sprachen
Zentrale Kompetenzen:
• Argumentieren
• Modellieren
• Implementieren
• Darstellen und Interpretieren
• Kommunizieren und Kooperieren
Zeitbedarf: 21/30 Stunden</td></tr>
<tr><td>Unterrichtsvorhaben Q2-III

Thema: Prinzipielle Arbeitsweise eines Computers inkl. Modellierung und Implementierung eines Scanners, Parsers und Interpreters für eine einfache maschinennahe Sprache sowie Grenzen der Automatisierung

Schwerpunkte: Einzelrechner und Rechnernetzwerke; Grenzen der Automatisierung; Scanner, Parser und Interpreter
Inhaltsfelder:
• Formale Sprachen und Automaten
• Algorithmen
• Informatiksysteme
• Informatik, Mensch und Gesellschaft
Zentrale Kompetenzen:
• Argumentieren
• Kommunizieren und Kooperieren
• Modellieren
• Implementieren
Zeitbedarf: 15/20 Stunden</td><td>Unterrichtsvorhaben Q2-VI

Thema: Entwicklung eines Netzwerkspiels mit Durchführung eines vollständigen Softwareentwicklungszyklus

Schwerpunkte: Objekte und Klassen; Analyse, Entwurf und Implementierung von Algorithmen; Syntax und Semantik einer Programmiersprache; Einzelrechner und Rechnernetzwerke; Nutzung von Informatiksystemen
Inhaltsfelder:
• Daten und ihre Strukturierung
• Algorithmen
• Informatiksysteme
• Formale Sprachen und Automaten
Zentrale Kompetenzen:
• Argumentieren
• Modellieren
• Implementieren
• Darstellen und Interpretieren
• Kommunizieren und Kooperieren
Zeitbedarf: 15 Stunden</td></tr>
<tr><td colspan="2">Summe: 60/100 Stunden</td></tr>
</table>

Hinweis: Um an dieser Stelle einen einfachen Vergleich der Themen und Inhalte im Grund- und im Leistungskurs zu ermöglichen und Parallelen aufzuzeigen, sind beide Raster in einer Übersicht zusammengefasst. Inhalte, die ausschließlich im Leistungskurs obligatorisch sind, sind farblich hervorgehoben.

2.1.2 Konkretisierte Unterrichtsvorhaben

2.1.2.1 Einführungsphase

Die folgenden Kompetenzen aus dem Bereich *Kommunizieren und Kooperieren* werden in allen Unterrichtsvorhaben der Einführungsphase vertieft und sollen aus Gründen der Lesbarkeit nicht in jedem Unterrichtsvorhaben separat aufgeführt werden:

Die Schüler:innen

- verwenden Fachausdrücke bei der Kommunikation über informatische Sachverhalte (K),
- präsentieren Arbeitsabläufe und -ergebnisse (K),
- kommunizieren und kooperieren in Gruppen und in Partnerarbeit (K),
- nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung und gemeinsamen Verwendung von Daten unter Berücksichtigung der Rechteverwaltung (K).

UV EF-I: Einführung in die Nutzung von Informatiksystemen und in grundlegende Begrifflichkeiten	
Leitfragen: ▪ Womit beschäftigt sich die Wissenschaft der Informatik? ▪ Wie kann die in der Schule vorhandene informatische Ausstattung genutzt werden?	
Vorhabenbezogene Konkretisierungen: Das erste Unterrichtsvorhaben stellt eine allgemeine Einführung in das Fach Informatik dar. Dabei ist zu berücksichtigen, dass für manche Schüler:innen in der Einführungsphase der erste Kontakt mit dem Unterrichtsfach Informatik stattfindet, so dass zu Beginn Grundlagen des Fachs behandelt werden müssen. Zunächst wird auf den Begriff der Information eingegangen und die Möglichkeit der Kodierung in Form von Daten thematisiert. Anschließend wird auf die Übertragung von Daten im Sinne des Sender-Empfänger-Modells eingegangen. Dabei wird eine überblickartige Vorstellung der Kommunikation von Rechnern in Netzwerken erarbeitet. Des Weiteren soll der grundlegende Aufbau eines Rechnersystems im Sinne der Von-Neumann-Architektur erarbeitet werden und mit dem grundlegenden Prinzip der Datenverarbeitung (Eingabe-Verarbeitung-Ausgabe) in Beziehung gesetzt werden. Bei der Beschäftigung mit Datenkodierung, Datenübermittlung und Datenverarbeitung ist jeweils ein Bezug zur konkreten Nutzung der informatischen Ausstattung der Schule herzustellen. So wird in die verantwortungsvolle Nutzung dieser Systeme eingeführt.	
Zu entwickelnde Kompetenzen: Die Schüler:innen ▪ beschreiben und erläutern den Aufbau und die Arbeitsweise singulärer Rechner am Beispiel der „Von-Neumann-Architektur“ (A), ▪ nutzen die im Unterricht eingesetzten Informatiksysteme selbstständig, sicher, zielführend und verantwortungsbewusst (D), ▪ nutzen das Internet zur Recherche, zum Datenaustausch und zur Kommunikation (K).	
Unterrichtssequenzen	*Beispiele, Medien, Materialien*
1. Information, deren Kodierung und Speicherung ▪ Informatik als Wissenschaft der Verarbeitung von Informationen ▪ Darstellung von Informationen in Schrift, Bild und Ton ▪ Speichern von Daten mit informatischen Sys-	*Beispiel*: Textkodierung Kodierung und Dekodierung von Texten mit unbekannten Zeichensätzen (z.B. Wingdings) *Beispiel*: Bildkodierung Kodierung von Bildinformationen in Raster- und Vektorgrafiken

temen am Beispiel der Schulrechner ▪ Vereinbarung von Richtlinien zur Datenspeicherung auf den Schulrechnern (z.B. Ordnerstruktur, Dateibezeichner usw.)	
2. Informations- und Datenübermittlung in Netzen ▪ „Sender-Empfänger-Modell“ und seine Bedeutung für die Eindeutigkeit von Kommunikation ▪ Informatische Kommunikation in Rechnernetzen am Beispiel des Schulnetzwerks (z.B. Benutzeranmeldung, Netzwerkordner, Zugriffsrechte, Client-Server) ▪ Grundlagen der technischen Umsetzung von Rechnerkommunikation am Beispiel des Internets (z.B. Netzwerkadresse, Paketvermittlung, Protokoll) ▪ Richtlinien zum verantwortungsvollen Umgang mit dem Internet	*Beispiel*: Rollenspiel zur Paketvermittlung im Internet Schüler:innen übernehmen die Rollen von Clients und Routern. Sie schicken spielerisch Informationen auf Karten von einem Schüler-Client zum anderen. Jede Schülerin und jeder Schüler hat eine Adresse, jeder Router darüber hinaus eine Routingtabelle. Mit Hilfe der Tabelle und einem Würfel wird entschieden, wie ein Paket weiter vermittelt wird.
3. Aufbau informatischer Systeme ▪ Identifikation typischer Komponenten informatischer Systeme und anschließende Beschränkung auf das Wesentliche, Herleitung der „Von-Neumann-Architektur“ ▪ Identifikation des EVA-Prinzips (Eingabe-Verarbeitung-Ausgabe) als Prinzip der Verarbeitung von Daten und Grundlage der „Von-Neumann-Architektur“	*Material*: Demonstrationshardware Durch Demontage eines Demonstrationsrechners entdecken Schüler:innen die verschiedenen Hardwarekomponenten eines Informatiksystems. Als Demonstrationsrechner bietet sich ein ausrangierter Schulrechner an.

UV EF-II: Grundlagen der objektorientierten Analyse, Modellierung und Implementierung anhand von statischen Grafikszenen

Leitfrage:
- Wie lassen sich Gegenstandsbereiche informatisch modellieren und im Sinne einer Simulation informatisch realisieren?

Vorhabenbezogene Konkretisierungen:
Ein zentraler Bestandteil des Informatikunterrichts der Einführungsphase ist die Objektorientierte Programmierung. Dieses Unterrichtsvorhaben führt in die Grundlagen der Analyse, Modellierung und Implementierung in diesem Kontext ein.

Dazu werden zunächst konkrete Gegenstandsbereiche aus der Lebenswelt der Schüler:innen analysiert und im Sinne des Objektorientierten Paradigmas strukturiert. Dabei werden die grundlegenden Begriffe der Objektorientierung und Modellierungswerkzeuge wie Objektkarten, Klassenkarten oder Beziehungsdiagramme eingeführt.

Im Anschluss wird mit der Realisierung erster Projekte mit Hilfe der didaktischen Programmierumgebung GLOOP begonnen. Die von der Bibliothek vorgegebenen Klassen werden von Schüler:innen in Teilen analysiert und entsprechende Objekte anhand einfacher Problemstellungen erprobt. Dazu muss der grundlegende Aufbau einer Java-Klasse thematisiert und zwischen Deklaration, Initialisierung und Methodenaufrufen unterschieden werden.

Da bei der Umsetzung dieser ersten Projekte konsequent auf die Verwendung von Kontrollstrukturen verzichtet wird und der Quellcode aus einer rein linearen Sequenz besteht, ist auf diese Weise eine Fo-

kussierung auf die Grundlagen der Objektorientierung möglich, oh-ne dass algorithmische Probleme ablenken. Natürlich kann die Arbeit an diesen Projekten unmittelbar zum nächsten Unterrichtsvorhaben führen. Dort stehen unter anderem Kontrollstrukturen im Mittelpunkt.

Zu entwickelnde Kompetenzen: Die Schüler:innen

- ermitteln bei der Analyse einfacher Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M),
- modellieren Klassen mit ihren Attributen, ihren Methoden und Assoziationsbeziehungen (M),
- stellen die Kommunikation zwischen Objekten grafisch dar (M),
- implementieren einfache Algorithmen unter Beachtung der Syntax und Semantik einer Programmiersprache (I),
- stellen den Zustand eines Objekts dar (D).

Unterrichtssequenzen	*Beispiele, Medien, Materialien*
1. Identifikation von Objekten ▪ Am Beispiel eines lebensweltnahen Beispiels werden Objekte im Sinne der Objektorientierten Modellierung eingeführt. ▪ Objekte werden mit Objektkarten visualisiert und mit sinnvollen Attributen und „Fähigkeiten", d.h. Methoden versehen. ▪ Manche Objekte sind prinzipiell typgleich und werden so zu einer Objektsorte bzw. Objektklasse zusammengefasst. ▪ Vertiefung: Modellierung weiterer Beispiele ähnlichen Musters	*Beispiel:* Vogelschwarm Schüler:innen betrachten einen Vogelschwarm als Menge gleichartiger Objekte, die in einer Klasse mit Attributen und Methoden zusammengefasst werden können. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator - Allgemeine Objektorientierung (Download EF-II.1)
2. Analyse von Klassen didaktischer Lernumgebungen ▪ Objektorientierte Programmierung als modularisiertes Vorgehen (Entwicklung von Problemlösungen auf Grundlage vorhandener Klassen) ▪ Teilanalyse der Klassen der didaktischen Lernumgebungen GLOOP	*Materialien:* Dokumentation der didaktischen Bibliothek GLOOP (Download EF-II.2)
3. Implementierung dreidimensionaler, statischer Szenen ▪ Grundaufbau einer Java-Klasse ▪ Konzeption einer Szene mit Kamera, Licht und sichtbaren Objekten ▪ Deklaration und Initialisierung von Objekten ▪ Methodenaufrufe mit Parameterüber-gabe zur Manipulation von Objekteigenschaften (z.B. Farbe, Position, Drehung)	*Beispiel:* Skulpturengarten Schüler:innen erstellen ein Pro-gramm, das mit Hilfe von geometrischen Objekten der GLOOP-Umgebung einen Skulpturengarten auf den Bildschirm bringt. *Beispiel:* Olympische Ringe Die Schüler:innen bilden das Emblem der olympischen Spiele mit Hilfe von GLOOP-Objekten nach. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator - Sequenzielle Programmierung *(Download EF-II.3)*

UV EF-III: Grundlagen der objektorientierten Programmierung und algorithmischer Grundstrukturen in Java anhand von einfachen Animationen

Leitfrage:

- Wie lassen sich Animationen und Simulationen optischer Gegenstandsbereiche unter Berücksichtigung von Tastatureingaben realisieren?

Vorhabenbezogene Konkretisierungen:
Der Schwerpunkt dieses Unterrichtsvorhabens liegt auf der Entwicklung mehrerer Projekte, die durch Eingaben des Benutzers gesteuerte Animationen aufweisen. Zunächst wird ein Projekt bearbeitet, das in Anlehnung an das vorangegangene Unterrichtsvorhaben eine Szene darstellt, die lediglich aus Objekten besteht, zu denen das didaktische System Klassen vorgibt. Einzelne Objekte der Szene werden animiert, um ein einfaches Spiel zu realisieren oder die Szene optisch aufzuwerten. Für die Umsetzung dieses Projekts werden Kontroll-strukturen in Form von Schleifen und Verzweigungen benötigt und eingeführt.

Sind an einem solchen Beispiel im Schwerpunkt Schleifen und Verzweigungen eingeführt worden, sollen diese Konzepte an weiteren Beispielprojekten eingeübt werden. Dabei muss es sich nicht zwangsläufig um solche handeln, bei denen Kontrollstrukturen lediglich zur Animation verwendet werden. Auch die Erzeugung größerer Mengen grafischer Objekte und deren Verwaltung in einem Feld kann ein Anlass zur Verwendung von Kontrollstrukturen sein.

Das Unterrichtsvorhaben schließt mit einem Projekt, das komplexere grafische Elemente beinhaltet, so dass die Schüler:innen mehr als nur die Klasse erstellen müssen, welche die Szene als Ganzes darstellt. Elemente der Szene müssen zu sinnhaften eigenen Klassen zusammengefasst werden, die dann ihre eigenen Attribute und Dienste besitzen. Auch dieses Projekt soll eine Animation, ggf. im Sinne einer Simulation, sein, bei der Attributwerte von Objekten eigener Klassen verändert werden und diese Veränderungen optisch sichtbar gemacht werden.

Komplexere Assoziationsbeziehungen zwischen Klassen werden in diesem Unterrichtsvorhaben zunächst nicht behandelt. Sie stellen den Schwerpunkt des folgenden Vorhabens dar.

Zu entwickelnde Kompetenzen: Die Schüler:innen
- analysieren und erläutern einfache Algorithmen und Programme (A),
- entwerfen einfache Algorithmen und stellen sie umgangssprachlich und grafisch dar (M),
- ermitteln bei der Analyse einfacher Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M),
- modellieren Klassen mit ihren Attributen, ihren Methoden und Assoziationsbeziehungen (M),
- ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare Datensammlungen zu (M),
- ordnen Klassen, Attributen und Methoden ihren Sichtbarkeitsbereich zu (M),
- modifizieren einfache Algorithmen und Programme (I),
- implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I),
- implementieren Algorithmen unter Verwendung von Variablen und Wertzuweisungen, Kontrollstrukturen sowie Methodenaufrufen (I),
- implementieren einfache Algorithmen unter Beachtung der Syntax und Semantik einer Programmiersprache (I),
- testen Programme schrittweise anhand von Beispielen (I),
- interpretieren Fehlermeldungen und korrigieren den Quellcode (I).

Unterrichtssequenzen	*Beispiele, Medien, Materialien*
1. Bewegungsanimationen am Beispiel einfacher grafischer Objekte (GLObjekte) ▪ Kontinuierliche Verschiebung eines GLObjekts mit Hilfe einer Schleife (While-Schleife) ▪ Tastaturabfrage zur Realisierung einer Schleifenbedingung für eine Animationsschleife ▪ Mehrstufige Animationen mit mehreren sequenziellen Schleifen	*Beispiel*: Wurfspiel Die Schüler:innen realisieren mit Objekten der GLOOP-Umgebung ein Spiel, bei dem ein Ball über den Bildschirm bewegt und auf eine runde Zielscheibe geworfen werden soll. *Materialien*: Ergänzungsmaterialien zum Lehrplannavigator – Kontrollstrukturen (Download EF-III.1)

▪ Berechnung von Abständen zwischen GLObjekten mit Hilfsvariablen ▪ Meldungen zur Kollision zweier GLObjekte mit Hilfe von Abstandsberechnungen und Verzweigungen (IF-Anweisungen)	
2. Erstellen und Verwalten größerer Mengen einfacher grafischer Objekte (GLObjekte) ▪ Erzeugung von Objekten mit Hilfe von Zählschleifen (FOR-Schleife) ▪ Verwaltung von Objekten in eindimensionalen Feldern (Arrays) ▪ Animation von Objekten, die in eindimensionalen Feldern (Arrays) verwaltet werden ▪ Vertiefung: Verschiedene Feldbeispiele	*Beispiel*: Hubschrauberlandeplatz Die Schüler:innen realisieren einen runden Hubschrauberlandeplatz und eine Reihe von Landemarkierungen, die in einem Feld verwaltet werden. Mit Hilfe der Landemarkierungen werden verschiedene Lauflichter realisiert. *Beispiel*: Schachbrett Die Schüler:innen realisieren mit Hilfe mehrerer Quader ein Schachbrett. *Beispiel*: Magischer Würfel Die Schüler:innen erstellen einen großen Würfel, der aus mehreren kleineren, farbigen Würfeln besteht. *Materialien*: Ergänzungsmaterialien zum Lehrplannavigator - Kontrollstrukturen (Download EF-III.2)
3. Modellierung und Animation komplexerer grafisch repräsentierbarer Objekte ▪ Modellierung eines Simulationspro-gramms mit eigenen Klassen, die sich selbst mit Hilfe von einfachen GLObjekten zeigen mit Hilfe eines Implementationsdiagramms ▪ Implementierung eigener Methoden mit und ohne Parameterübergabe ▪ Realisierung von Zustandsvariablen ▪ Thematisierung des Geheimnisprinzips und des Autonomitätsprinzips von Objekten ▪ Animation mit Hilfe des Aufrufs von selbstimplementierten Methoden ▪ Vertiefung: Weitere Projekte	*Beispiel*: Kerzensimulation Die Schüler:innen modellieren und erstellen eine Klasse, mit deren Hilfe Kerzen simuliert werden können. Eine Kerze kann angezündet und gelöscht werden. Abgesehen davon brennen Kerzen abhängig von ihrer Dicke unterschiedlich schnell ab. *Beispiel*: Uhren Die Schüler:innen erstellen eine Simulation mehrerer Uhren für verschiedene Zeit-zonen. *Beispiel*: Ampeln Die Schüler:innen erstellen eine Ampelkreuzung mit mehreren Ampelanlagen an einem Bahnübergang. *Materialien*: Ergänzungsmaterialien zum Lehrplannavigator – Eigene Klassen (Download EF-III.3)

UV EF-IV: Modellierung und Implementierung von Klassen- und Objektbeziehungen anhand von grafischen Spielen und Simulationen
Leitfrage: ▪ Wie lassen sich komplexere Datenflüsse und Beziehungen zwischen Objekten und Klassen realisieren?
Vorhabenbezogene Konkretisierungen: Dieses Unterrichtsvorhaben beschäftigt sich im Schwerpunkt mit dem Aufbau komplexerer Objektbezie-

hungen. Während in vorangegangenen Unterrichtsvorhaben Objekte nur jeweils solchen Objekten Nachrichten schicken konnten, die sie selbst erstellt haben, soll in diesem Unterrichtsvorhaben diese hierarchische Struktur aufgebrochen werden.

Dazu bedarf es zunächst einer präzisen Unterscheidung zwischen Objektreferenzen und Objekten, so dass klar wird, dass Dienste eines Objektes von unterschiedlichen Objekten über unterschiedliche Referenzen in Anspruch genommen werden können. Auch der Aufbau solcher Objektbeziehungen muss thematisiert werden. Des Weiteren wird das Prinzip der Vererbung im objektorientierten Sinne angesprochen. Dazu werden die wichtigsten Varianten der Vererbung anhand von verschiedenen Projekten vorgestellt. Zunächst wird die Vererbung als Spezialisierung im Sinne einer einfachen Erweiterung einer Oberklasse vorgestellt. Darauf folgt ein Projekt, welches das Verständnis von Vererbung um den Aspekt der späten Bindung erweitert, indem Dienste einer Oberklasse überschrieben werden. Modellierungen sollen in Form von Implementationsdiagrammen erstellt werden.

Zum Abschluss kann kurz auf das Prinzip der abstrakten Klasse eingegangen werden. Dieser Inhalt ist aber nicht obligatorisch für die Einführungsphase.

Zu entwickelnde Kompetenzen: Die Schüler:innen

- analysieren und erläutern eine objektorientierte Modellierung (A),
- stellen die Kommunikation zwischen Objekten grafisch dar (M),
- ermitteln bei der Analyse einfacher Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M),
- modellieren Klassen mit ihren Attributen, ihren Methoden und Assoziationsbeziehungen (M),
- ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare Datensammlungen zu (M),
- ordnen Klassen, Attributen und Methoden ihren Sichtbarkeitsbereich zu (M),
- modellieren Klassen unter Verwendung von Vererbung (M),
- implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I),
- testen Programme schrittweise anhand von Beispielen (I),
- interpretieren Fehlermeldungen und korrigieren den Quellcode (I),
- modifizieren einfache Algorithmen und Programme (I),
- stellen Klassen, Assoziations- und Vererbungsbeziehungen in Diagrammen grafisch dar (D),
- dokumentieren Klassen durch Beschreibung der Funktionalität der Methoden (D).

Unterrichtssequenzen	*Beispiele, Medien, Materialien*
1. Vertiefung des Referenzbegriffs und Einführung des Prinzips der dynamischen Referenzierung ▪ Einführung der GLOOP-Objektselektion mit der Maus ▪ Einführung der Klasse GLObjekt als Oberklasse aller sichtbaren Objekte in GLOOP ▪ Steuerung einfacher grafischer Objekte über eine Referenz aktuell, die jeweils durch eine Klickselektion mit der Maus auf ein neues Objekt gesetzt werden kann.	*Beispiel:* Seifenblasen Die Schüler:innen entwickeln ein Spiel, bei dem Seifenblasen über den Bildschirm schweben und durch Anklicken mit der Maus zum Zerplatzen gebracht werden können. *Beispiel:* Sonnensystem Die Schüler:innen entwickeln eine Simulation des Sonnensystems bei der Daten zum angeklickten Planeten ausgegeben werden.
2. Entwicklung eines Spiels mit der Notwendigkeit von Kollisionskontrollen zwischen zwei oder mehr grafischen Objekten ▪ Modellierung des Spiels ohne Berücksichtigung der Kollision mit Hilfe eines Implementationsdiagramms ▪ Dokumentation der Klassen des Projekts ▪ Implementierung eines Prototypen ohne Kol-	*Beispiel:* Ufospiel Die Schüler:innen entwickeln die Simulation eines Ufos, das Asteroiden ausweichen soll, mit denen eine Kollision möglich ist. *Beispiel:* Billardkugeln Die Schüler:innen entwickeln ein Spiel, bei dem tickende Billardkugeln mit einer beweglichen Box

lision ▪ Ergänzung einer Kollisionsabfrage durch zusätzliche Assoziationsbeziehungen in Diagramm, Dokumentation und Quellcode ▪ Verallgemeinerung der neuen Verwendung von Objektreferenzen ▪ Vertiefung: Entwicklung weiterer Spiele und Simulationen mit vergleichbarer Grundmodellierung	eingefangen werden sollen. *Beispiel:* Autospiel Die Schüler:innen entwickeln ein Autospiel, bei dem ein Auto durch einen Wald fahren und mit Bäumen kollidieren kann. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator – Assoziationen (Download EF-IV.1) *Informationsblatt:* Implementationsdiagramme (Download EF-IV.2)
3. Erarbeitung einer Simulation mit grafischen Objekten, die sich durch unterschiedliche Ergänzungen voneinander unterscheiden (Vererbung durch Spezialisierung ohne Überschreiben von Methoden) ▪ Analyse und Erläuterung einer Basisversion der grafischen Klasse ▪ Realisierung von grafischen Erweiterungen zur Basisklasse mit und ohne Vererbung (Implementationsdiagramm und Quellcode) ▪ Verallgemeinerung und Reflexion des Prinzips der Vererbung am Beispiel der Spezialisierung	*Beispiel:* Schneemann Die Schüler:innen erstellen eine Simulation von Schneemännern, die unterschiedliche Kopfbedeckungen tragen. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator – Vererbung (Download EF-IV.3)
4. Entwicklung einer komplexeren Simulation mit grafischen Elementen, die unterschiedliche Animationen durchführen (Vererbung mit Überschreiben von Methoden) ▪ Analyse und Erläuterung einer einfachen grafischen Animationsklasse ▪ Spezialisierung der Klasse zu Unterklassen mit verschiedenen Animationen durch Überschreiben der entsprechenden Animationsmethode ▪ Reflexion des Prinzips der späten Bindung ▪ Vertiefung: Entwicklung eines vergleichbaren Projekts mit einer (abstrakten) Oberklasse	*Beispiel:* Flummibälle Die Schüler:innen entwickeln eine Simulation von Flummibällen, bei der unter-schiedliche Bälle unterschiedliche Bewegungen durchführen. *Beispiel:* Weihnachtsbaum Die Schüler:innen entwickeln eine Simulation eines Weihnachtsbaums mit Hilfe einer abstrakten Klasse Schmuck. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator – Vererbung (Download EF-IV.4)

UV EF-V: Such- und Sortieralgorithmen anhand kontextbezogener Beispiele
Leitfrage: ▪ Wie können Objekte bzw. Daten effizient sortiert werden, so dass eine schnelle Suche möglich wird?
Vorhabenbezogene Konkretisierungen: Dieses Unterrichtsvorhaben beschäftigt sich mit der Erarbeitung von Such- und Sortieralgorithmen. Der Schwerpunkt des Vorhabens liegt dabei auf den Algorithmen selbst und nicht auf deren Implementierung in einer Programmiersprache, auf die in diesem Vorhaben vollständig verzichtet werden soll. Zunächst erarbeiten die Schüler:innen mögliche Einsatzszenarien für Such- und Sortieralgorithmen, um sich der Bedeutung einer effizienten Lösung dieser Probleme bewusst zu werden. Anschließend werden

Strategien zur Sortierung mit Hilfe eines explorativen Spiels von den Schüler:innen selbst erarbeitet und hinsichtlich der Anzahl notwendiger Vergleiche auf ihre Effizienz untersucht.

Daran anschließend werden die erarbeiteten Strategien systematisiert und im Pseudocode notiert. Die Schüler:innen sollen auf diese Weise das Sortieren durch Vertauschen, das Sortieren durch Auswählen und mindestens einen weiteren Sortieralgorithmus, kennen lernen.

Des Weiteren soll das Prinzip der binären Suche behandelt und nach Effizienzgesichtspunkten untersucht werden.

Zu entwickelnde Kompetenzen: Die Schüler:innen

- beurteilen die Effizienz von Algorithmen am Beispiel von Sortierverfahren hinsichtlich Zeit und Speicherplatzbedarf (A),
- entwerfen einen weiteren Algorithmus zum Sortieren (M),
- analysieren Such- und Sortieralgorithmen und wenden sie auf Beispiele an (D).

Unterrichtssequenzen	*Beispiele, Medien, Materialien*
1. Explorative Erarbeitung eines Sortierverfahrens ▪ Sortierprobleme im Kontext informatischer Systeme und im Alltag (z.B. Dateisortierung, Tabellenkalkulation, Telefonbuch, Bundesligatabelle, usw.) ▪ Vergleich zweier Elemente als Grundlage eines Sortieralgorithmus ▪ Erarbeitung eines Sortieralgorithmus durch die Schüler:innen	*Beispiel:* Sortieren mit Waage Die Schüler:innen bekommen die Aufgabe, kleine, optisch identische Kunststoffbehälter aufsteigend nach ihrem Gewicht zu sortieren. Dazu steht ihnen eine Balkenwaage zur Ver-fügung, mit deren Hilfe sie das Gewicht zweier Behälter vergleichen können. *Materialien:* Computer Science Unplugged – Sorting Algorithms, URL: www.csunplugged.org/sorting-algorithms abgerufen: 30. 03. 2014
2. Systematisierung von Algorithmen und Effizienzbetrachtungen ▪ Formulierung (falls selbst gefunden) oder Erläuterung von mehreren Algorithmen im Pseudocode (auf jeden Fall: Sortieren durch Vertauschen, Sortieren durch Auswählen) ▪ Anwendung von Sortieralgorithmen auf verschiedene Beispiele ▪ Bewertung von Algorithmen anhand der Anzahl der nötigen Vergleiche ▪ Variante des Sortierens durch Auswählen (Nutzung eines einzigen oder zweier Felder bzw. lediglich eines einzigen zusätzlichen Ablageplatzes oder mehrerer neuer Ablageplätze) ▪ Effizienzbetrachtungen an einem konkreten Beispiel bezüglich der Rechenzeit und des Speicherplatzbedarfs ▪ Analyse des weiteren Sortieralgorithmus (sofern nicht in Sequenz 1 und 2 bereits geschehen)	*Beispiele:* Sortieren durch Auswählen, Sortieren durch Vertauschen, Quicksort Quicksort ist als Beispiel für einen Algorithmus nach dem Prinzip *Teile und Herrsche* gut zu behandeln. Kenntnisse in rekursiver Programmierung sind nicht erforderlich, da eine Implementierung nicht angestrebt wird. *Materialien:* Computer Science Unplugged – Sorting Algorithms, URL: www.csunplugged.org/sorting-algorithms
3. Binäre Suche auf sortierten Daten ▪ Suchaufgaben im Alltag und im Kontext informatischer Systeme	*Beispiel:* Simulationsspiel zur binären Suche nach Tischtennisbällen Mehrere Tischtennisbälle sind nummeriert, sortiert

▪ Evtl. Simulationsspiel zum effizienten Suchen mit binärer Suche ▪ Effizienzbetrachtungen zur binären Suche	und unter Bechern verdeckt. Mit Hilfe der binären Suche kann sehr schnell ein bestimmter Tischtennisball gefunden werden. *Materialien:* Computer Science Unplugged – Searching Algorithms, URL: www.csunplugged.org/searching-algorithms

UV EF-VI: Geschichte der digitalen Datenverarbeitung und die Grundlagen des Datenschutzes	
Leitfrage: ▪ Welche Entwicklung durchlief die moderne Datenverarbeitung und welche Auswirkungen ergeben sich insbesondere hinsichtlich neuer An-forderungen an den Datenschutz daraus?	
Vorhabenbezogene Konkretisierungen: Das folgende Unterrichtsvorhaben stellt den Abschluss der Einführungsphase dar. Schüler:innen sollen selbstständig informatische Themenbereiche aus dem Kontext der Geschichte der Datenverarbeitung und insbesondere den daraus sich ergebenen Fragen des Datenschutzes bearbeiten. Diese Themenbereiche werden in Kleingruppen bearbeitet und in Form von Plakatpräsentationen vorgestellt. Schüler:innen sollen dabei mit Unterstützung des Lehrenden selbstständige Recherchen zu ihren Themen anstellen und auch eine sinnvolle Eingrenzung ihres Themas vornehmen. Anschließend wird verstärkt auf den Aspekt des Datenschutzes eingegangen. Dazu wird das Bundesdatenschutzgesetz in Auszügen behandelt und auf schülernahe Beispielsituationen zur Anwendung gebracht. Dabei steht keine formale juristische Bewertung der Beispielsituationen im Vordergrund, die im Rahmen eines Informatikunterrichts auch nicht geleistet werden kann, sondern vielmehr eine persönliche Einschätzung von Fällen im Geiste des Datenschutzgesetzes.	
Zu entwickelnde Kompetenzen: Die Schüler:innen ▪ bewerten anhand von Fallbeispielen die Auswirkungen des Einsatzes von Informatiksystemen (A), ▪ erläutern wesentliche Grundlagen der Geschichte der digitalen Datenverarbeitung (A), ▪ stellen ganze Zahlen und Zeichen in Binärcodes dar (D), ▪ interpretieren Binärcodes als Zahlen und Zeichen (D), ▪ nutzen das Internet zur Recherche, zum Datenaustausch und zur Kommunikation. (K).	
Unterrichtssequenzen	*Beispiele, Medien, Materialien*
1. Selbstständige Erarbeitung von Themen durch die Schüler:innen ▪ Mögliche Themen zur Erarbeitung in Kleingruppen: o „Eine kleine Geschichte der Digitalisierung: vom Morsen zum modernen Digitalcomputer“ o „Eine kleine Geschichte der Kryptographie: von Caesar zur Enigma“ o „Von Nullen, Einsen und mehr: Stellenwertsysteme und wie man mit ihnen rechnet“ o „Kodieren von Texten und Bildern: ASCII, RGB und mehr“ o „Auswirkungen der Digitalisierung: Veränderungen der Arbeitswelt und Datenschutz“ ▪ Vorstellung und Diskussion durch Schüler:innen	*Beispiel:* Ausstellung zu informatischen Themen Die Schüler:innen bereiten eine Ausstellung zu informatischen Themen vor. Dazu werden Stellwände und Plakate vorbereitet, die ggf. auch außerhalb des Informatikunterrichts in der Schule ausgestellt werden können. *Materialien:* Schüler:innen recherchieren selbst-ständig im Internet, in der Schulbibliothek, in öffentlichen Bibliotheken, usw.

2. Vertiefung des Themas Datenschutz ▪ Erarbeitung grundlegender Begriffe des Datenschutzen ▪ Problematisierung und Anknüpfung an die Lebenswelt der Schüler:innen ▪ Diskussion und Bewertung von Fallbeispielen aus dem Themenbereich „Datenschutz“	*Beispiel:* Fallbeispiele aus dem aktuellen Tagesgeschehen Die Schüler:innen bearbeiten Fallbeispiele aus ihrer eigenen Erfahrungswelt oder der aktuellen Medienberichterstattung. *Materialien:* Materialblatt zum Bundesdatenschutzgesetz (Download EF-VI.1)

2.1.2.2 Qualifikationsphase (Grundkurs)

Die folgenden Kompetenzen aus dem Bereich Kommunizieren und Kooperieren werden in allen Unterrichtsvorhaben der Qualifikations-phase vertieft und sollen aus Gründen der Lesbarkeit nicht in jedem Unterrichtsvorhaben separat aufgeführt werden:

Die Schüler:innen

- verwenden die Fachsprache bei der Kommunikation über informatische Sachverhalte (K),
- nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung von Dateien unter Berücksichtigung der Rechteverwaltung (K),
- organisieren und koordinieren kooperatives und eigenverantwortliches Arbeiten (K),
- strukturieren den Arbeitsprozess, vereinbaren Schnittstellen und führen Ergebnisse zusammen (K),
- beurteilen Arbeitsorganisation, Arbeitsabläufe und Ergebnisse (K),
- präsentieren Arbeitsabläufe und -ergebnisse adressatengerecht (K).

UV Q1-I: Wiederholung der objektorientierten Modellierung und Programmierung
Leitfragen: ▪ Wie modelliert und implementiert man zu einer Problemstellung in einem geeigneten Anwendungskontext Java-Klassen inklusive ihrer Attribute, Methoden und Beziehungen? ▪ Wie kann man die Modellierung und die Funktionsweise der Anwendung grafisch darstellen?
Vorhabenbezogene Konkretisierungen: Zu einer Problemstellung in einem Anwendungskontext soll eine Java-Anwendung entwickelt werden. Die Problemstellung soll so gewählt sein, dass für diese Anwendung die Verwendung einer abstrakten Oberklasse als Generalisierung verschiedener Unterklassen sinnvoll erscheint und eine Klasse durch eine Unterklasse spezialisiert werden kann. Um die Aufgabe einzugrenzen, können (nach der ersten Problemanalyse) einige Teile (Modellierungen oder Teile von Java-Klassen) vorgegeben werden. Die Schüler:innen erläutern und modifizieren den ersten Entwurf und modellieren sowie implementieren weitere Klassen und Methoden für eine entsprechende Anwendung. Klassen und ihre Beziehungen werden in einem Implementationsdiagramm dargestellt. Dabei werden Sichtbarkeitsbereiche zugeordnet. Exemplarisch wird eine Klasse dokumentiert. Der Nachrichtenaustausch zwischen verschiedenen Objekten wird verdeutlicht, indem die Kommunikation zwischen zwei ausgewählten Objekten grafisch dargestellt wird. In diesem Zusammenhang wird das Nachrichtenkonzept der objektorientierten Programmierung wiederholt.
Zu entwickelnde Kompetenzen: Die Schüler:innen ▪ analysieren und erläutern objektorientierte Modellierungen (A), ▪ beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), ▪ modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter

Angabe von Multiplizitäten (M),
- ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M),
- modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M),
- implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I),
- nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I),
- wenden eine didaktisch orientierte Entwicklungsumgebung zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I),
- interpretieren Fehlermeldungen und korrigieren den Quellcode (I),
- stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D),
- dokumentieren Klassen (D),
- stellen die Kommunikation zwischen Objekten grafisch dar (D).

Unterrichtssequenzen	*Beispiele, Medien, Materialien*
1. Wiederholung und Erweiterung der objektorientierten Modellierung und Programmierung durch Analyse und Erweiterung eines kontextbezogenen Beispiels ▪ Analyse der Problemstellung ▪ Analyse der Modellierung (Implementationsdiagramm) ▪ Erweiterung der Modellierung im Implementationsdiagramm (Vererbung, abstrakte Klasse) ▪ Kommunikation zwischen mindestens zwei Objekten (grafische Darstellung) ▪ Dokumentation von Klassen ▪ Implementierung der Anwendung oder von Teilen der Anwendung	*Beispiel:* Wetthuepfen Für ein Wetthüpfen zwischen einem Hasen, einem Hund und einem Vogel werden die Tiere gezeichnet. Alle Tiere springen wiederholt nach links. Die Höhe und Weite jedes Hüpfers ist zufällig. Evtl. marschieren sie anschließend hintereinander her. oder *Beispiel:* Tannenbaum Ein Tannenbaum soll mit verschiedenen Arten von Schmuckstücken versehen werden, die durch unterschiedliche geometrische Objekte dargestellt werden. Es gibt Kugeln, Päckchen in der Form von Würfeln und Zuckerringe in Form von Toren. Ein Prototyp, der bereits mit Kugeln geschmückt werden kann, kann zur Verfügung gestellt werden. Da alle Schmuckstücke über die Funktion des Auf- und Abschmückens verfügen sollen, liegt es nahe, dass entsprechende Methoden in einer gemeinsamen Oberklasse realisiert werden. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.1-Wiederholung (Download Q1-I.1)

UV Q1-II: Modellierung und Implementierung von Anwendungen mit dynamischen, linearen Datenstrukturen

Leitfrage:
- Wie können beliebig viele linear angeordnete Daten im Anwendungskontext verwaltet werden?

Vorhabenbezogene Konkretisierungen:
Nach Analyse einer Problemstellung in einem geeigneten Anwendungskontext, in dem Daten nach dem First-In-First-Out-Prinzip verwaltet werden, werden der Aufbau von Schlangen am Beispiel dargestellt und die Operationen der Klasse Queue erläutert. Anschließend werden für die Anwendung notwendige Klas-

sen modelliert und implementiert. Eine Klasse für eine den Anforderungen der Anwendung entsprechende Oberfläche sowie die Klasse Queue wird dabei von der Lehrkraft vorgegeben. Anschließend wird die Anwendung modifiziert, um den Umgang mit der Datenstruktur zu üben.

Anhand einer Anwendung, in der Daten nach dem Last-In-First-Out-Prinzip verwaltet werden, werden Unterschiede zwischen den Datenstrukturen Schlange und Stapel erarbeitet. Um einfacher an Objekte zu gelangen, die zwischen anderen gespeichert sind, wird die Klasse List eingeführt und in einem Anwendungskontext verwendet. In mindestens einem weiteren Anwendungskontext wird die Verwaltung von Daten in Schlangen, Stapeln oder Listen vertieft. Modellierungen werden dabei in Entwurfs- und Implementationsdiagrammen dargestellt.

Zu entwickelnde Kompetenzen: Die Schüler:innen

- erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A),
- analysieren und erläutern Algorithmen und Programme (A),
- beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A),
- ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu (M),
- ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M),
- modifizieren Algorithmen und Programme (I),
- implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I),
- nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I),
- interpretieren Fehlermeldungen und korrigieren den Quellcode (I),
- testen Programme systematisch anhand von Beispielen (I),
- stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D).

Unterrichtssequenzen	*Beispiele, Medien, Materialien*
1. Die Datenstruktur Schlange im Anwendungskontext unter Nutzung der Klasse Queue ▪ Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen ▪ Erarbeitung der Funktionalität der Klasse Queue ▪ Modellierung und Implementierung der Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse Queue	*Beispiel:* Patientenwarteschlange (jeder kennt seinen Nachfolger bzw. alternativ: seinen Vorgänger) Sobald ein Patient in einer Arztpraxis eintrifft, werden sein Name und seine Krankenkasse er-fasst. Die Verwaltung der Patientenwarteschlange geschieht über eine Klasse, die hier als Wartezimmer bezeichnet wird. Wesentliche Operationen sind das „Hinzufügen“ eines Patienten und das „Entfernen“ eines Patienten, wenn er zur Behandlung gerufen wird. Die Simulationsanwendung stellt eine GUI zur Verfügung, legt ein Wartezimmer an und steuert die Abläufe. Wesentlicher Aspekt des Projektes ist die Modellierung des Wartezimmers mit Hilfe der Klasse Queue. Anschließend wird der Funktionsumfang der Anwendung erweitert: Patienten können sich zusätzlich in die Warteschlange zum Blutdruckmessen einreihen. Objekte werden von zwei Schlangen verwaltet. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.2 – Warteschlange (Download Q1-II.1)
2. Die Datenstruktur Stapel im Anwendungskon-	*Beispiel:* Heftstapel

text unter Nutzung der Klasse Stack ▪ Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen ▪ Erarbeitung der Funktionalität der Klasse Stack ▪ Modellierung und Implementierung der Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse Stack	In einem Heftstapel soll das Heft einer Schülerin gefunden werden. oder *Beispiel:* Kisten stapeln In einem Stapel nummerierter Kisten soll eine bestimmte Kiste gefunden und an einen Kunden geliefert werden. Dazu müssen Kisten auf verschiedene Stapel gestapelt und wieder zurückgestellt werden.
3. Die Datenstruktur lineare Liste im Anwendungskontext unter Nutzung der Klasse List ▪ Erarbeitung der Vorteile der Klasse List im Gegensatz zu den bereits bekannten linearen Strukturen ▪ Modellierung und Implementierung einer kontextbezogenen Anwendung unter Verwendung der Klasse List.	*Beispiel:* Abfahrtslauf Bei einem Abfahrtslauf kommen die Skifahrer nacheinander an und werden nach ihrer Zeit in eine Rangliste eingeordnet. Diese Rangliste wird in einer Anzeige ausgegeben. Ankommende Ab-fahrer müssen an jeder Stelle der Struktur, nicht nur am Ende oder Anfang eingefügt werden können. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.2 - Listen (Download Q1-II.2)
4. Vertiefung - Anwendungen von Listen, Stapeln oder Schlangen in mindestens einem weiteren Kontext	*Beispiel:* Skispringen Ein Skispringen hat folgenden Ablauf: Nach dem Sprung erhält der Springer eine Punktzahl und wird nach dieser Punktzahl in eine Rangliste eingeordnet. Die besten 30 Springer qualifizieren sich für den zweiten Durchgang. Sie starten in umgekehrter Reihenfolge gegenüber der Platzierung auf der Rangliste. Nach dem Sprung erhält der Springer wiederum eine Punktzahl und wird nach der Gesamtpunktzahl aus beiden Durchgängen in die endgültige Rangliste eingeordnet. *Beispiel:* Terme in Postfix-Notation Die sog. UPN (Umgekehrt-Polnische-Notation) bzw. Postfix-Notation eines Terms setzt den Operator hinter die Operanden. Um einen Term aus der gewohnten Infixschreibweise in einen Term in UPN umzuwandeln oder um den Wert des Terms zu berechnen, kann ein Stack verwendet werden. *Beispiel:* Rangierbahnhof Auf einem Güterbahnhof gibt es drei Gleise, die nur zu einer Seite offen sind. Wagons können also von einer Seite auf das Gleis fahren und nur rückwärts wieder hinausfahren. Die Wagons tragen Nummern, wobei die Nummer jedoch erst eingesehen werden kann, wenn der Wagon der vorderste an der offenen Gleisseite ist. (Zwischen den Wagons herumzuturnen, um die anderen Wagonnummern zu lesen, wäre zu gefährlich.) Zunächst stehen alle Wagons unsortiert auf einem Gleis. Ziel ist es, alle Wagons in

	ein anderes Gleis zu fahren, so dass dort die Nummern der Wagons vom Gleisende aus aufsteigend in richtiger Reihenfolge sind. Zusätzlich zu diesen beiden Gleisen gibt es ein Abstellgleis, das zum Rangieren benutzt werden kann. *Beispiel:* Autos an einer Ampel zur Zufahrtsregelung Es soll eine Ampel zur Zufahrtsregelung in Java simuliert werden. An einem geradlinigen, senk-recht von unten nach oben verlaufenden Straßenstück, das von Autos nur einspurig in eine Richtung befahren werden kann, ist ein Halte-punkt markiert, an dem die Ampel steht. Bei einem Klick auf eine Schaltfläche mit der Aufschrift „Heranfahren“ soll ein neues Auto an den Halte-punkt heranfahren bzw. bis an das letzte Auto, das vor dem Haltepunkt wartet. Grünphasen der Ampel werden durch einen Klick auf eine Schalt-fläche mit der Aufschrift „Weiterfahren“ simuliert. In jeder Grünphase darf jeweils nur ein Auto weiterfahren. Die anderen Autos rücken nach. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1-II.3 – Anwendungen für lineare Datenstrukturen (Download Q1-II.3)

UV Q1-III: Suchen und Sortieren auf linearen Datenstrukturen
Leitfrage: ▪ Wie kann man gespeicherte Informationen günstig (wieder-)finden?
Vorhabenbezogene Konkretisierungen: In einem Anwendungskontext werden zunächst Informationen in einer linearen Liste bzw. einem Feld gesucht. Hierzu werden Verfahren entwickelt und implementiert bzw. analysiert und erläutert, wobei neben einem iterativen auch ein rekursives Verfahren thematisiert wird und mindestens ein Verfahren selbst entwickelt und implementiert wird. Die verschiedenen Verfahren werden hinsichtlich Speicherbedarf und Zahl der Vergleichsoperationen miteinander verglichen. Anschließend werden Sortierverfahren entwickelt und implementiert (ebenfalls für lineare Listen und Felder). Hierbei soll auch ein rekursives Sortierverfahren entwickelt werden. Die Implementationen von Quicksort sowie dem Sortieren durch Einfügen werden analysiert und erläutert. Falls diese Verfahren vorher schon entdeckt wurden, sollen sie hier wiedererkannt werden. Die rekursive Abarbeitung eines Methodenaufrufs von Quicksort wird grafisch dargestellt. Abschließend werden verschiedene Sortierverfahren hinsichtlich der Anzahl der benötigten Vergleichsoperationen und des Speicherbedarfs beurteilt.
Zu entwickelnde Kompetenzen: Die Schüler:innen ▪ analysieren und erläutern Algorithmen und Programme (A), ▪ beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), ▪ beurteilen die Effizienz von Algorithmen unter Berücksichtigung des Speicherbedarfs und der Zahl der Operationen (A),

▪ entwickeln iterative und rekursive Algorithmen unter Nutzung der Strategien „Modularisierung“ und „Teilen und Herrschen“ (M), ▪ modifizieren Algorithmen und Programme (I), ▪ implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), ▪ implementieren und erläutern iterative und rekursive Such- und Sortierverfahren (I), ▪ nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), ▪ interpretieren Fehlermeldungen und korrigieren den Quellcode (I), ▪ testen Programme systematisch anhand von Beispielen (I), ▪ stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D).	
Unterrichtssequenzen	*Beispiele, Medien, Materialien*
1. Suchen von Daten in Listen und Arrays ▪ Lineare Suche in Listen und in Arrays ▪ Binäre Suche in Arrays als Beispiel für rekursives Problemlösen ▪ Untersuchung der beiden Suchverfahren hinsichtlich ihrer Effizienz (Laufzeitverhalten, Speicherbedarf)	*Beispiel:* Karteiverwaltung Für ein Adressverwaltungsprogramm soll eine Methode zum Suchen einer Adresse geschrieben werden. oder *Beispiel:* Bundesjugendspiele Die Teilnehmer an Bundesjugendspielen nehmen an drei Disziplinen teil und erreichen dort Punktzahlen. Diese werden in einer Wettkampfkarte eingetragen und an das Wettkampfbüro gegeben. Zur Vereinfachung sollte sich das Modell auf die drei Disziplinen „Lauf”, „Sprung“ und „Wurf“ beschränken. Im Wettkampfbüro wird das Ergebnis erstellt. Das Programm soll dafür zunächst den Besten einer Disziplin heraussuchen können und später das gesamte Ergebnis nach gewissen Kriterien sortieren können. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.3 - Suchen und Sortieren (Download Q1-III.1)
2. Sortieren in Listen und Arrays - Entwicklung und Implementierung von iterativen und rekursiven Sortierverfahren ▪ Entwicklung und Implementierung eines einfachen Sortierverfahrens für eine Liste ▪ Implementierung eines einfachen Sortierverfahrens für ein Feld ▪ Entwicklung eines rekursiven Sortierverfahren für ein Feld (z.B. Sortieren durch Mischen)	*Beispiel:* Karteiverwaltung (s.o.) oder *Beispiel:* Bundesjugendspiele (s.o.) *Materialien:* (s.o.)
3. Untersuchung der Effizienz der Sortierverfahren „Sortieren durch direktes Einfügen“ und „Quicksort“ auf linearen Listen ▪ Grafische Veranschaulichung der Sortierver-	*Beispiel:* Karteiverwaltung (s.o.) oder

fahren ▪ Untersuchung der Anzahl der Vergleichsoperationen und des Speicherbedarf bei beiden Sortierverfahren ▪ Beurteilung der Effizienz der beiden Sortierverfahren	*Beispiel:* Bundesjugendspiele (s.o.) *Materialien:* (s.o.)

UV Q1-IV: Modellierung und Nutzung von relationalen Datenbanken in Anwendungskontexten

Leitfragen:
- Wie können Fragestellungen mit Hilfe einer Datenbank beantwortet werden?
- Wie entwickelt man selbst eine Datenbank für einen Anwendungskontext?

Vorhabenbezogene Konkretisierungen:
Ausgehend von einer vorhandenen Datenbank entwickeln Schüler:innen für sie relevante Fragestellungen, die mit dem vorhandenen Datenbestand beantwortet werden sollen. Zur Beantwortung dieser Fragestellungen wird die vorgegebene Datenbank von den Schüler:innen analysiert und die notwendigen Grundbegriffe für Datenbanksysteme sowie die erforderlichen SQL-Abfragen werden erarbeitet.

In anderen Anwendungskontexten müssen Datenbanken erst noch entwickelt werden, um Daten zu speichern und Informationen für die Beantwortung von möglicherweise auftretenden Fragen zur Verfügung zu stellen. Dafür ermitteln Schüler:innen in den Anwendungssituationen Entitäten, zugehörige Attribute, Relationen und Kardinalitäten und stellen diese in Entity-Relationship-Modellen dar. Entity-Relationship-Modelle werden interpretiert und erläutert, modifiziert und in Datenbankschemata überführt. Mit Hilfe von SQL-Anweisungen können anschließend im Kontext relevante Informationen aus der Datenbank extrahiert werden.

Ein Entity-Relationship-Diagramm kann auch verwendet werden, um die Entitäten inklusive ihrer Attribute und Relationen in einem vor-gegebenen Datenbankschema darzustellen.

An einem Beispiel wird verdeutlicht, dass in Datenbanken Redundanzen unerwünscht sind und Konsistenz gewährleistet sein sollte. Die 1. bis 3. Normalform wird als Gütekriterium für Datenbankentwürfe eingeführt. Datenbankschemata werden hinsichtlich der 1. bis 3. Normalform untersucht und (soweit nötig) normalisiert.

Zu entwickelnde Kompetenzen: Die Schüler:innen
- erläutern die Eigenschaften und den Aufbau von Datenbanksystemen unter dem Aspekt der sicheren Nutzung (A),
- analysieren und erläutern die Syntax und Semantik einer Datenbankabfrage (A),
- analysieren und erläutern eine Datenbankmodellierung (A),
- erläutern die Eigenschaften normalisierter Datenbankschemata (A),
- bestimmen Primär- und Sekundärschlüssel (M),
- ermitteln für anwendungsbezogene Problemstellungen Entitäten, zugehörige Attribute, Relationen und Kardinalitäten (M),
- modifizieren eine Datenbankmodellierung (M),
- modellieren zu einem Entity-Relationship-Diagramm ein relationales Datenbankschema (M),
- bestimmen Primär- und Sekundärschlüssel (M),
- überführen Datenbankschemata in vorgegebene Normalformen (M),
- verwenden die Syntax und Semantik einer Datenbankabfragesprache, um Informationen aus einen Datenbanksystem zu extrahieren (I),
- ermitteln Ergebnisse von Datenbankabfragen über mehrere verknüpfte Tabellen (D),
- stellen Entitäten mit ihren Attributen und die Beziehungen zwischen Entitäten in einem Entity-Relationship-Diagramm grafisch dar (D),
- überprüfen Datenbankschemata auf vorgegebene Normalisierungseigenschaften (D).

Unterrichtssequenzen	*Beispiele, Medien, Materialien*

1. Nutzung von relationalen Datenbanken ▪ Aufbau von Datenbanken und Grundbegriffe o Entwicklung von Fragestellungen zur vorhandenen Datenbank o Analyse der Struktur der vorgegebenen Datenbank und Erarbeitung der Begriffe Tabelle, Attribut, Datensatz, Datentyp, Primärschlüssel, Fremdschlüssel, Datenbankschema ▪ SQL-Abfragen o Analyse vorgegebener SQL-Abfragen und Erarbeitung der Sprachelemente von SQL (SELECT (DISTINCT) …FROM, WHERE, AND, OR, NOT) auf einer Tabelle o Analyse und Erarbeitung von SQL-Abfragen auf einer und mehrerer Tabelle zur Beantwortung der Fragestellungen (JOIN, UNION, AS, GROUP BY,ORDER BY, ASC, DESC, COUNT, MAX, MIN, SUM, Arithmetische Operatoren: +, -, *, /, (…), Vergleichsoperatoren: =, <>, >, <, >=, <=, LIKE, BETWEEN, IN, IS NULL) ▪ Vertiefung an einem weiteren Datenbankbeispiel	*Beispiel:* VideoCenter VideoCenter ist die Simulation einer Online-Videothek für den Informatik-Unterricht mit Webfrontends zur Verwaltung der Kunden, der Videos und der Ausleihe. Außerdem ist es möglich direkt SQL-Abfragen einzugeben. Es ist auch möglich, die Datenbank herunter zu laden und lokal zu installieren. *Beispiel:* Schulbuchausleihe Es wird eine Datenbank zur Verfügung gestellt, die Daten einer Schulbuch-Ausleihe enthält (über 1000 Entleiher, 200 Bücher mit mehreren tausend Exemplaren und viele Ausleihvorgänge). Die Datenbank kann in OpenOffice eingebunden werden.
2. Modellierung von relationalen Datenbanken ▪ Entity-Relationship-Diagramm o Ermittlung von Entitäten, zugehörigen Attributen, Relationen und Kardinalitäten in Anwendungssituationen und Modellierung eines Datenbankentwurfs in Form eines Entity-Relationship-Diagramms o Erläuterung und Modifizierung einer Datenbankmodellierung ▪ Entwicklung einer Datenbank aus einem Datenbankentwurf o Modellierung eines relationalen Datenbankschemas zu einem Entity-Relationship-Diagramm inklusive der Bestimmung von Primär- und Sekundärschlüsseln ▪ Redundanz, Konsistenz und Normalformen o Untersuchung einer Datenbank hinsichtlich Konsistenz und Redundanz in einer Anwendungssituation o Überprüfung von Datenbankschemata hinsichtlich der 1. bis 3. Normalform und Normalisierung (um Redundanzen zu vermeiden und Konsistenz zu gewährleisten)	*Beispiel:* Fahrradverleih Der Fahrradverleih BTR (BikesToRent) verleiht unterschiedliche Typen von Fahrrädern diverser Firmen an seine Kunden. Die Kunden sind bei BTR registriert (Name, Adresse, Telefon). BTR kennt von den Fahrradfirmen den Namen und die Telefonnummer. Kunden von BTR können CityBikes, Treckingräder und Mountainbikes ausleihen. *Beispiel:* Reederei Die Datenverwaltung einer Reederei soll in einem Datenbanksystem umgesetzt werden. Ausgehend von der Modellierung soll mit Hilfe eines ER-Modells und eines Datenbankschemas dieser erste Entwurf normalisiert und in einem Datenbanksystem umgesetzt werden. Es schließen sich diverse SQL-Abfragen an, wobei auf die Relationenalgebra eingegangen wird. *Beispiel:* Buchungssystem In dem Online-Buchungssystem einer Schule können die Lehrer Medienräume, Beamer, Laptops, Kameras, usw. für einen bestimmten Zeit-punkt buchen, der durch Datum und die Schul-stunde festgelegt ist. Dazu ist die Datenbank zu modellieren, ggf. zu normalisieren und im Datenbanksystem umzusetzen.

	Weiter sollen sinnvolle Abfragen entwickelt werden. Unter http://mrbs.sourceforge.net (abgerufen: 30.03. 2014) findet man ein freies Online-Buchungssystem inklusive Demo, anhand derer man erläutern kann, worum es in dem Projekt geht. *Beispiel:* Schulverwaltung In einer Software werden die Schulhalbjahre, Jahrgangsstufen, Kurse, Klassen, Schüler, Lehrer und Noten einer Schule verwaltet. Man kann dann ablesen, dass z.B. Schüler X von Lehrer Y im 2. Halbjahr des Schuljahrs 2011/2012 in der Jahrgangsstufe 9 im Differenzierungsbereich im Fach Informatik die Note „sehr gut“ erhalten hat. Dazu ist die Datenbank zu modellieren, ggf. zu normalisieren und im Datenbanksystem umzusetzen. Weiter sollen sinnvolle Abfragen entwickelt werden und das Thema Datenschutz besprochen werden.

UV Q1-V: Sicherheit und Datenschutz in Netzstrukturen	
Leitfragen: ▪ Wie werden Daten in Netzwerken übermittelt? ▪ Was sollte man in Bezug auf die Sicherheit beachten?	
Vorhabenbezogene Konkretisierungen: Anschließend an das vorhergehende Unterrichtsvorhaben zum Thema Datenbanken werden der Datenbankzugriff aus dem Netz, Topologien von Netzwerken, eine Client-Server-Struktur, das TCP/IP-Schichtenmodell sowie Sicherheitsaspekte beim Zugriff auf Datenbanken und verschiedene symmetrische und asymmetrische kryptografische Verfahren analysiert und erläutert. Fallbeispiele zur Datenschutzproblematik und zum Urheberrecht runden das Unterrichtsvorhaben ab.	
Zu entwickelnde Kompetenzen: Die Schüler:innen ▪ beschreiben und erläutern Topologien, die Client-Server-Struktur und Protokolle sowie ein Schichtenmodell in Netzwerken (A), ▪ analysieren und erläutern Eigenschaften und Einsatzbereiche symmetrischer und asymmetrischer Verschlüsselungsverfahren (A), ▪ untersuchen und bewerten anhand von Fallbeispielen die Auswirkungen des Einsatzes von Informatiksystemen, die Sicherheit von Informatiksystemen sowie die Einhaltung der Datenschutzbestimmungen und des Urheberrechts (A), ▪ untersuchen und bewerten Problemlagen, die sich aus dem Einsatz von Informatiksystemen ergeben, hinsichtlich rechtlicher Vorgaben, ethischer Aspekte und gesellschaftlicher Werte unter Berücksichtigung unterschiedlicher Interessenlagen (A), ▪ nutzen bereitgestellte Informatiksysteme und das Internet reflektiert zum Erschließen, zur Aufbereitung und Präsentation fachlicher Inhalte (D).	
Unterrichtssequenzen	*Beispiele, Medien, Materialien*
1. Daten in Netzwerken und Sicherheitsaspekte in Netzen sowie beim Zugriff auf Datenbanken ▪ Beschreibung eines Datenbankzugriffs im Netz anhand eines Anwendungskontextes und einer Client-Server-Struktur zur Klärung der Funktionsweise eines Datenbankzugriffs	*Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben, Verschlüsselung Q1.5 - Zu-griff auf Daten in Netzwerken (Download Q1-V.1)

▪ Netztopologien als Grundlage von Client-Server-Strukturen und TCP/IP-Schichtenmodell als Beispiel für eine Paketübermittlung in einem Netz ▪ Vertraulichkeit, Integrität, Authentizität in Netzwerken sowie symmetrische und asymmetrische kryptografische Verfahren (Cäsar-, Vigenère-, RSA-Verfahren) als Methoden Daten im Netz verschlüsselt zu übertragen	
2. Fallbeispiele zur Datenschutzproblematik und zum Urheberrecht	*Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1 5 - Datenschutz beim Videocenter, Materialblatt-Datenschutzgesetz (Download Q1-V.2)

UV Q2-I: Modellierung und Implementierung von Anwendungen mit dynamischen, nichtlinearen Datenstrukturen

Leitfragen:

- Wie können Daten im Anwendungskontext mit Hilfe binärer Baumstrukturen verwaltet werden?
- Wie kann dabei der rekursive Aufbau der Baumstruktur genutzt werden?
- Welche Vor- und Nachteile haben Suchbäume für die geordnete Verwaltung von Daten?

Vorhabenbezogene Konkretisierungen:

Anhand von Beispielen für Baumstrukturen werden grundlegende Begriffe eingeführt und der rekursive Aufbau binärer Bäume dargestellt.

Anschließend werden für eine Problemstellung in einem der Anwendungskontexte Klassen modelliert und implementiert. Dabei werden die Operationen der Datenstruktur Binärbaum thematisiert und die entsprechende Klasse BinaryTree (der Materialien für das Zentralabitur in NRW) der Vorgaben für das Zentralabitur NRW verwendet. Klassen und ihre Beziehungen werden in Entwurfs- und Implementationsdiagrammen dargestellt. Die Funktionsweise von Methoden wird anhand grafischer Darstellungen von Binärbäumen erläutert.

Unter anderem sollen die verschiedenen Baumtraversierungen (Pre-, Post- und Inorder) implementiert werden. Unterschiede bezüglich der Möglichkeit, den Baum anhand der Ausgabe der Bauminhalte via Pre-, In- oder Postorder-Traversierung zu rekonstruieren, werden dabei ebenfalls angesprochen, indem die fehlende Umkehrbarkeit der Zuordnung Binärbaum → Inorder-Ausgabe an einem Beispiel verdeutlicht wird.

Eine Tiefensuche wird verwendet, um einen in der Baumstruktur gespeicherten Inhalt zu suchen.

Zu einer Problemstellung in einem entsprechenden Anwendungskontext werden die Operationen der Datenstruktur Suchbaum thematisiert und unter der Verwendung der Klasse BinarySearchTree (der Materialien für das Zentralabitur in NRW) weitere Klassen oder Methoden in diesem Anwendungskontext modelliert und implementiert. Auch in diesem Kontext werden grafische Darstellungen der Bäume verwendet.

Die Verwendung von binären Bäumen und Suchbäumen wird anhand weiterer Problemstellungen oder anderen Kontexten weiter geübt.

Zu entwickelnde Kompetenzen: Die Schüler:innen

- erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A),
- analysieren und erläutern Algorithmen und Programme (A),
- beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A),
- ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen

und ihre Beziehungen (M),
- ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu (M),
- modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M),
- verwenden bei der Modellierung geeigneter Problemstellungen die Möglichkeiten der Polymorphie (M),
- entwickeln iterative und rekursive Algorithmen unter Nutzung der Konstruktionsstrategien „Modularisierung“ und „Teilen und Herrschen“ (M),
- implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I),
- modifizieren Algorithmen und Programme (I),
- nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I),
- interpretieren Fehlermeldungen und korrigieren den Quellcode (I),
- testen Programme systematisch anhand von Beispielen (I),
- stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D),
- stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D).

Unterrichtssequenzen	*Beispiele, Medien, Materialien*
1. Analyse von Baumstrukturen in verschiedenen Kontexten ▪ Grundlegende Begriffe (Grad, Tiefe, Höhe, Blatt, Inhalt, Teilbaum, Ebene, Vollständigkeit) ▪ Aufbau und Darstellung von binären Bäumen anhand von Baumstrukturen in verschiedenen Kontexten	*Beispiel:* Termbaum Der Aufbau von Termen wird mit Hilfe von binären Baumstrukturen verdeutlicht. oder *Beispiel:* Ahnenbaum Die binäre Baumstruktur ergibt sich daraus, dass jede Person genau einen Vater und eine Mutter hat. Weitere Beispiele für Anwendungskontexte für binäre Bäume: *Beispiel:* Suchbäume (zur sortierten Speicherung von Daten) Alle Inhalte, die nach einer Ordnung vor dem Inhalt im aktuellen Teilbaum stehen, sind in dessen linkem Teilbaum, alle die nach dem Inhalt im aktuellen Teilbaum stehen, sind in dessen rechtem Teilbaum. (Dies gilt für alle Teilbäume.) oder *Beispiel:* Entscheidungsbäume Um eine Entscheidung zu treffen, werden mehrere Fragen mit ja oder nein beantwortet. Die Fragen, die möglich sind, wenn die Antwort auf eine Frage mit „ja“ beantwortet wird, befinden sich im linken Teilbaum, die Fragen, die möglich sind, wenn die Antwort „nein“ lautet, stehen im rechten Teilbaum. oder *Beispiel:* Codierungsbäume für Codierungen, deren

	Alphabet aus genau zwei Zeichen besteht Morse hat Buchstaben als Folge von Punkten und Strichen codiert. Diese Codierungen können in einem Binärbaum dargestellt werden, so dass ein Übergang zum linken Teilbaum einem Punkt und ein Übergang zum rechten Teilbaum einem Strich entspricht. Wenn man im Gesamtbaum startet und durch Übergänge zu linken oder rechten Teilbäumen einen Pfad zum gewünschten Buchstaben sucht, erhält man die Morsecodierung des Buchstabens. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.1 – Binärbaum (Download Q2-I.1)
2. Die Datenstruktur Binärbaum im Anwendungskontext unter Nutzung der Klasse BinaryTree ▪ Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen im Anwendungskontext ▪ Modellierung eines Entwurfsdiagramms und Entwicklung eines Implementationsdiagramms ▪ Erarbeitung der Klasse BinaryTree und beispielhafte Anwendung der Operationen ▪ Implementierung der Anwendung oder von Teilen der Anwendung ▪ Traversierung eines Binärbaums im Pre-, In- und Postorderdurchlauf	*Beispiel:* Informatikerbaum als binärer Baum In einem binären Baum werden die Namen und die Geburtsdaten von Informatikern lexikographisch geordnet abgespeichert. Alle Namen, die nach dieser Ordnung vor dem Namen im aktuellen Teilbaum stehen, sind in dessen linkem Teil-baum, alle die nach dem Namen im aktuellen Teilbaum stehen, sind in dessen rechtem Teil-baum. (Dies gilt für alle Teilbäume.) Folgende Funktionalitäten werden benötigt: ▪ Einfügen der Informatiker-Daten in den Baum ▪ Suchen nach einem Informatiker über den Schlüssel Name ▪ Ausgabe des kompletten Datenbestands in nach Namen sortierter Reihenfolge *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.1 – Binärbaum (Download Q2-I.2)
3. Die Datenstruktur binärer Suchbaum im Anwendungskontext unter Verwendung der Klasse BinarySearchTree ▪ Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen ▪ Modellierung eines Entwurfsdiagramms und Entwicklung eines Implementationsdiagramm, ▪ grafische Darstellung eines binären Suchbaums und Erarbeitung der Struktureigenschaften ▪ Erarbeitung der Klasse BinarySearchTree und Einführung des Interface Item zur Realisierung einer geeigneten Ordnungsrelation ▪ Implementierung der Anwendung oder von	*Beispiel:* Informatikerbaum als Suchbaum In einem binären Suchbaum werden die Namen und die Geburtsdaten von Informatikern lexikographisch geordnet abgespeichert. Alle Namen, die nach dieser Ordnung vor dem Namen im aktuellen Teilbaum stehen, sind in dessen linkem Teilbaum, alle die nach dem Namen im aktuellen Teilbaum stehen, sind in dessen rechtem Teil-baum. (Dies gilt für alle Teilbäume.) Folgende Funktionalitäten werden benötigt: ▪ Einfügen der Informatiker-Daten in den Baum ▪ Suchen nach einem Informatiker über den Schlüssel Name ▪ Ausgabe des kompletten Datenbestands in

Teilen der Anwendung inklusive einer sortierten Ausgabe des Baums	nach Namen sortierter Reihenfolge *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.1 – Binärer Suchbaum (Download Q2-I.3)
4. Übung und Vertiefungen der Verwendung von Binärbäumen oder binären Suchbäumen anhand weiterer Problemstellungen	*Beispiel:* Codierungsbäume (s.o.) oder Huffman-Codierung oder *Beispiel:* Buchindex Es soll eine Anwendung entwickelt werden, die anhand von Stichworten und zugehörigen Seitenzahlen ein Stichwortregister erstellt. Da die Stichwörter bei der Analyse des Buches häufig gesucht werden müssen, werden sie in der Klasse Buchindex als Suchbaum (Objekt der Klasse BinarySearchTree) verwaltet. Alle Inhalte, die nach einer Ordnung vor dem Inhalt im aktuellen Teilbaum stehen, sind in dessen linkem Teilbaum, alle die nach dem Inhalt im aktuellen Teilbaum stehen, sind in dessen rechtem Teilbaum. (Dies gilt für alle Teilbäume.) oder *Beispiel:* Entscheidungsbäume (s.o.) oder *Beispiel:* Termbaum (s.o.) oder *Beispiel:* Ahnenbaum (s.o.) *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.1 – Anwendung Binär-baum (Download Q2-I.4)

UV Q2-II: Endliche Automaten und formale Sprachen
Leitfragen: ▪ Wie kann man (endliche) Automaten genau beschreiben? ▪ Wie können endliche Automaten (in alltäglichen Kontexten oder zu informatischen Problemstellungen) modelliert werden? ▪ Wie können Sprachen durch Grammatiken beschrieben werden? ▪ Welche Zusammenhänge gibt es zwischen formalen Sprachen, endlichen Automaten und regulären Grammatiken?
Vorhabenbezogene Konkretisierungen:

Anhand kontextbezogener Beispiele werden endliche Automaten entwickelt, untersucht und modifiziert. Dabei werden verschiedene Darstellungsformen für endliche Automaten ineinander überführt und die akzeptierten Sprachen endlicher Automaten ermittelt. An einem Beispiel wird ein nichtdeterministischer Akzeptor eingeführt als Alternative gegenüber einem entsprechenden deterministischen Akzeptor.

Anhand kontextbezogener Beispiele werden Grammatiken regulärer Sprachen entwickelt, untersucht und modifiziert. Der Zusammen-hang zwischen regulären Grammatiken und endlichen Automaten wird verdeutlicht durch die Entwicklung von allgemeinen Verfahren zur Erstellung einer regulären Grammatik für die Sprache eines gegebenen endlichen Automaten bzw. zur Entwicklung eines endlichen Automaten, der genau die Sprache einer gegebenen regulären Grammatik akzeptiert.

Auch andere Grammatiken werden untersucht, entwickelt oder modifiziert. An einem Beispiel werden die Grenzen endlicher Automaten ausgelotet.

Zu entwickelnde Kompetenzen: Die Schüler:innen

- analysieren und erläutern die Eigenschaften endlicher Automaten einschließlich ihres Verhaltens auf bestimmte Eingaben (A),
- analysieren und erläutern Grammatiken regulärer Sprachen (A),
- zeigen die Grenzen endlicher Automaten und regulärer Grammatiken im Anwendungszusammenhang auf (A),
- ermitteln die formale Sprache, die durch eine Grammatik erzeugt wird (A),
- entwickeln und modifizieren zu einer Problemstellung endliche Automaten (M),
- entwickeln und modifizieren zu einer Problemstellung endliche Automaten (M),
- entwickeln zur akzeptierten Sprache eines Automaten die zugehörige Grammatik (M),
- entwickeln zur Grammatik einer regulären Sprache einen zugehörigen endlichen Automaten (M),
- modifizieren Grammatiken regulärer Sprachen (M),
- entwickeln zu einer regulären Sprache eine Grammatik, die die Sprache erzeugt (M),
- stellen endliche Automaten in Tabellen oder Graphen dar und überführen sie in die jeweils andere Darstellungsform (D),
- ermitteln die Sprache, die ein endlicher Automat akzeptiert (D).
- beschreiben an Beispielen den Zusammenhang zwischen Automaten und Grammatiken (D).

Unterrichtssequenzen	*Beispiele, Medien, Materialien*
1. Endliche Automaten ▪ Vom Automaten in den Schüler:innen bekannten Kontexten zur formalen Beschreibung eines endlichen Automaten ▪ Untersuchung, Darstellung und Entwicklung endlicher Automaten	*Beispiele:* Cola-Automat, Geldspielautomat, Roboter, Zustandsänderung eines Objekts „Auto“, Akzeptor für bestimmte Zahlen, Akzeptor für Teilwörter in längeren Zeichenketten, Akzeptor für Terme *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.2 – Endliche Automaten, Formale Sprachen (Download Q2-II.1)
2. Untersuchung und Entwicklung von Grammatiken regulärer Sprachen ▪ Erarbeitung der formalen Darstellung regulärer Grammatiken ▪ Untersuchung, Modifikation und Entwicklung von Grammatiken ▪ Entwicklung von endlichen Automaten zum Erkennen regulärer Sprachen die durch Grammatiken gegeben werden ▪ Entwicklung regulärer Grammatiken zu endlichen Automaten	*Beispiele:* reguläre Grammatik für Wörter mit ungerader Parität, Grammatik für Wörter, die bestimmte Zahlen repräsentieren, Satzgliederungsgrammatik *Materialien:* (s.o.)

3. Grenzen endlicher Automaten	*Beispiele:* Klammerausdrücke, $a^n b^n$ im Vergleich zu $(ab)^n$

UV Q2-III: Prinzipielle Arbeitsweise eines Computers und Grenzen der Automatisierbarkeit	
Leitfragen: ▪ Was sind die strukturellen Hauptbestandteile eines Computers und wie kann man sich die Ausführung eines maschinenahen Programms mit diesen Komponenten vorstellen? ▪ Welche Möglichkeiten bieten Informatiksysteme und wo liegen ihre Grenzen?	
Vorhabenbezogene Konkretisierungen: Anhand einer von-Neumann-Architektur und einem maschinennahen Programm wird die prinzipielle Arbeitsweise von Computern verdeutlicht. Ausgehend von den prinzipiellen Grenzen endlicher Automaten liegt die Frage nach den Grenzen von Computern bzw. nach Grenzen der Automatisierbarkeit nahe. Mit Hilfe einer entsprechenden Java-Methode wird plausibel, dass es unmöglich ist, ein Informatiksystem zu entwickeln, dass für jedes beliebige Computerprogramm und jede beliebige Eingabe entscheidet, ob das Programm mit der Eingabe terminiert oder nicht (Halteproblem). Anschließend werden Vor- und Nachteile der Grenzen der Automatisierbarkeit angesprochen und der Einsatz von Informatiksystemen hinsichtlich prinzipieller Möglichkeiten und prinzipieller Grenzen beurteilt.	
Zu entwickelnde Kompetenzen: Die Schüler:innen ▪ erläutern die Ausführung eines einfachen maschinennahen Programms sowie die Datenspeicherung auf einer „Von-Neumann-Architektur“ (A), ▪ untersuchen und beurteilen Grenzen des Problemlösens mit Informatiksystemen (A).	
Unterrichtssequenzen	*Beispiele, Medien, Materialien*
1. Von-Neumann-Architektur und die Ausführung maschinennaher Programme ▪ prinzipieller Aufbau einer von Neumann-Architektur mit CPU, Rechenwerk, Steuerwerk, Register und Hauptspeicher ▪ einige maschinennahe Befehlen und ihre Repräsentation in einem Binär-Code, der in einem Register gespeichert werden kann ▪ Analyse und Erläuterung der Funktionsweise eines einfachen maschinennahen Programms	*Beispiel:* Addition von 4 zu einer eingegeben Zahl mit einem Rechnermodell *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.3 –Von-Neumann-Architektur und maschinennahe Programmierung (Download Q2-III.1)
2. Grenzen der Automatisierbarkeit ▪ Vorstellung des Halteproblems ▪ Unlösbarkeit des Halteproblems ▪ Beurteilung des Einsatzes von Informatiksystemen hinsichtlich prinzipieller Möglichkeiten und prinzipieller Grenzen	*Beispiel:* Halteproblem *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.3 - Halteproblem (Download Q2-III.2)

2.1.2.3 Qualifikationsphase (Leistungskurs)

Die folgenden Kompetenzen aus dem Bereich Kommunizieren und Kooperieren werden in allen Unterrichtsvorhaben der Qualifikations-phase vertieft und sollen aus Gründen der Lesbarkeit nicht in jedem Unterrichtsvorhaben separat aufgeführt werden:

Die Schüler:innen

- verwenden die Fachsprache bei der Kommunikation über informatische Sachverhalte (K),
- nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung von Dateien unter Berücksichtigung der Rechteverwaltung (K),
- organisieren und koordinieren kooperatives und eigenverantwortliches Arbeiten (K),
- strukturieren den Arbeitsprozess, vereinbaren Schnittstellen und führen Ergebnisse zusammen (K),
- beurteilen Arbeitsorganisation, Arbeitsabläufe und Ergebnisse (K),
- präsentieren Arbeitsabläufe und -ergebnisse adressatengerecht (K).

UV Q1-I: Wiederholung der objektorientierten Modellierung und Programmierung auch unter Berücksichtigung der Gestaltung einer Benutzungsoberfläche
Leitfragen: ▪ Wie modelliert und implementiert man zu einer Problemstellung in einem geeigneten Anwendungskontext Java-Klassen inklusive ihrer Attribute, Methoden und Beziehungen? ▪ Wie kann man die Modellierung und die Funktionsweise der Anwendung grafisch darstellen? ▪ Wie kann ein Softwareprojekt mit angemessener Benutzungsoberfläche arbeitsteilig entwickelt und getestet werden?
Vorhabenbezogene Konkretisierungen: Zu einer Problemstellung in einem Anwendungskontext soll eine Java-Anwendung entwickelt werden. Diese Anwendung soll aus einer Benutzungsoberfläche und einer davon getrennten Verarbeitung der mit der Oberfläche eingegebenen Daten bestehen. Auf diese Weise werden nicht nur Grundideen der Objektorientierung wiederholt, sondern auch erste Schritte zur Thematisierung des MVC-Prinzips realisiert. Die Schüler:innen analysieren zu Beginn die gegebene Problemstellung und formulieren Anforderungen an die zu entwickelnde Java-Anwendung. Anschließend wird eine Benutzungsoberfläche nach softwareergonomischen Kriterien konzipiert und mit einem GUI-Builder umgesetzt. Mit Blick auf die fertige Benutzungsoberfläche wird eine Objektorientierte Modellierung entwickelt und in Form eines Implementationsdia-gramms fixiert. Wesentliche Klassen werden von Schüler:innen dokumentiert. Der Nachrichtenaustausch zwischen verschiedenen Objekten wird verdeutlicht, indem die Kommunikation zwischen zwei ausgewählten Objekten grafisch dargestellt wird. In diesem Zusammenhang wird das Nachrichtenkonzept der Objektorientierten Programmierung wiederholt. Eine Implementierung der Java-Anwendung kann arbeitsteilig erfolgen. Zum Schluss wird das fertige Projekt mit Hilfe einer Testanwendung getestet.
Zu entwickelnde Kompetenzen: Die Schüler:innen ▪ analysieren und erläutern objektorientierte Modellierungen (A), ▪ beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), ▪ modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (M), ▪ ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M), ▪ entwickeln mit didaktisch orientierten Entwicklungsumgebungen einfache Benutzungsoberflächen zur Kommunikation mit einem Informatiksystem (M), ▪ implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I), ▪ nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I),

- wenden eine didaktisch orientierte Entwicklungsumgebung zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I),
- interpretieren Fehlermeldungen und korrigieren den Quellcode (I),
- testen Programme systematisch anhand von Beispielen und mithilfe von Testanwendungen(I),
- stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D),
- dokumentieren Klassen (D),
- stellen die Kommunikation zwischen Objekten grafisch dar (D),
- nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung von Daten, zur Organisation von Arbeitsabläufen sowie zur Verteilung und Zusammenführung von Arbeitsanteilen (K).

Unterrichtssequenzen	*Beispiele, Medien, Materialien*
1. Entwicklung eines Anforderungskatalogs Analyse der Problemstellung ▪ Analyse der Problemstellung ▪ Entwicklung von Anwendungsfällen (use cases) ▪ Formulierung eines Anforderungskatalogs	*Beispiel*: Taschenrechner Schüler:innen entwickeln eine Taschenrechneranwendung. Dabei kann es sich um einen Rechner mit einfachen Grundrechenarten bis hin zu einem wissenschaftlichen Taschenrechner mit umfangreicheren Funktionen handeln. Die Anforderungen sind von den Schüler:innen selbst zu definieren.
2. Entwurf einer grafischen Benutzungsoberfläche ▪ Erarbeitung softwareergonomischer Prinzipien für Benutzungsoberflächen ▪ Entwurf von Prototypen für eine Benutzungsoberfläche unter Berücksichtigung des Anforderungskatalogs ▪ Einarbeitung in die Funktionsweise eines GUI-Builders ▪ Realisierung einer Benutzungsoberfläche mit einem GUI-Builder	*Materialien*: Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.I (Download LK-Q1-I.1)
3. Klassenmodellierung und Visualisierung von Objektkommunikation ▪ Entwurf eines Implementationsdiagramms unter Berücksichtigung des Anforderungskatalogs und der Benutzungsoberfläche ▪ Grafische Darstellung der Kommunikation zwischen mindestens zwei Objekten ▪ Dokumentation von Klassen	
4. Projektentwicklung und Test (ggf. arbeitsteilig) ▪ Implementierung der Anwendung mit grafischer Benutzungsoberfläche ▪ Test einzelner Klassen der Gesamtanwendung mit Hilfe von Testanwendungen	*Materialien*: Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.I (Download LK-Q1-I.2)

UV Q1-II: Modellierung und Implementierung von dynamischen linearen Datenstrukturen sowie von Anwendungen unter Verwendung dynamischer, lineare Datenstrukturen

Leitfrage:
- Wie können beliebig viele linear angeordnete Daten in unterschiedlichen Anwendungskontexten problemgerecht verwaltet werden?

Vorhabenbezogene Konkretisierungen:

Nach Analyse einer Problemstellung in einem geeigneten Anwendungskontext, in dem Daten nach dem First-In-First-Out-Prinzip verwaltet werden, wird der Aufbau von einer geeigneten Datenstruktur entwickelt und implementiert. Es werden unterschiedliche Implementationsmöglichkeiten verglichen. Anschließend werden die für die Anwendung notwendigen Klassen modelliert und implementiert. Die Klasse Queue wird als Alternative zur selbstentwickelten Datenstruktur vorgestellt. Anschließend wird die Anwendung modifiziert, um den Umgang mit der Datenstruktur zu üben.

Anhand einer Anwendung, in der Daten nach dem Last-In-First-Out-Prinzip verwaltet werden, werden Unterschiede zwischen den Datenstrukturen Schlange und Stapel erarbeitet. Eine der beiden Klassen zu den linearen Strukturen sollte vollständig modelliert, implementiert und dokumentiert werden.

Um einfacher an Objekte zu gelangen, die zwischen anderen gespeichert sind, wird ein Anwendungskontext vorgestellt, bei dem die Anwendung der Klassen Queue und Stack nicht problemgerecht ist. Die Klasse List, bei der das Einfügen an beliebiger Stelle einer sequenziell angeordneten Struktur möglich ist, wird eingeführt und in einem Anwendungskontext verwendet.

In mindestens einem weiteren Anwendungskontext wird die Verwaltung von Daten in Schlangen, Stapeln oder Listen vertieft. Modellierungen werden dabei in Entwurfs- und Implementationsdiagrammen dargestellt.

Zu entwickelnde Kompetenzen: Die Schüler:innen

- modellieren Klassen mit ihren Attributen, Methoden und Assoziationsbeziehungen unter Angabe von Multiplizitäten (M),
- implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I),
- erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A),
- analysieren und erläutern Algorithmen und Programme (A),
- beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A),
- ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare und nichtlineare Datensammlungen zu (M),
- ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M),
- modifizieren Algorithmen und Programme (I),
- implementieren Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (I),
- implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I),
- nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I),
- interpretieren Fehlermeldungen und korrigieren den Quellcode (I),
- testen Programme systematisch anhand von Beispielen und mit Hilfe von Testanwendungen (I),
- stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D).

Unterrichtssequenzen	*Beispiele, Medien, Materialien*
1. Modellierung und Implementation einer Datenstruktur Schlange, die die Daten nach dem FIFO-Prinzip im Anwendungskontext verwaltet.	*Beispiel*: (Schlange) Patientenwarteschlange Sobald eine Patientin oder ein Patient in einer Arztpraxis eintrifft, werden Name und Krankenkasse erfasst. Die Verwaltung der Patientenwarteschlange geschieht über eine Klasse, die hier als Wartezimmer bezeichnet wird. Wesentliche Operationen sind das „Hinzufügen“ von Patientinnen und Patienten sowie das „Entfernen“ von Patientinnen und Patienten, nach-dem sie zur Behandlung gerufen wurden. Die Simulationsanwendung stellt eine GUI zur Verfügung, legt ein Wartezimmer an und steuert die Abläufe.

	Erster Lösungsansatz: Die FIFO-Datenstruktur wird problembezogen modelliert und implementiert. *Materialien*: Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.II (Download LK-Q1-II.1)
2. Die Datenstruktur Schlange im Anwendungskontext unter Nutzung der Klasse Queue ▪ Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen ▪ Erarbeitung der Funktionalität der Klasse Queue und des Konzepts der parametrisierten Klassen ▪ Modellierung und Implementierung der Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse Queue	Zweiter Lösungsansatz: Bei der Modellierung und Implementation wird der generische Daten-typ Queue verwendet. *Materialien*: Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.II (Download LK-Q1-II.2)
3. Die Datenstruktur Stapel im Anwendungskontext unter Nutzung der Klasse Stack ▪ Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen ▪ Erarbeitung der Funktionalität der Klasse Stack ▪ Modellierung und Implementierung der Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse Stack	*Beispiel*: (Stapel) Handdechiffrierer mit Undo-Operation. In einem Text sind Buchstaben vertauscht worden, er wird so zu einem Geheimtext. Dieser Text wird in ein Softwareprodukt kopiert und schrittweise dechiffriert. Es können für jedes Zeichen Ersetzungsvorschläge gemacht wer-den und die Software setzt diese Ersetzung im gesamten Text um. Am Kontrolltext bemerkt man, ob die Ersetzungen sinnvoll waren. Falls man einen Irrweg gegangen ist, kann man die Ersetzungen bis zu dem gewünschten Punkt schrittweise rückgängig machen. Die Modellierung und die Implementation er-folgt mit dem generischen Datentyp Stack. *Materialien*: Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.II (Download LK-Q1-II.3)
4. Die Datenstruktur Lineare Liste im Anwendungskontext unter Nutzung der Klasse List ▪ Erarbeitung der Vorteile der Klasse List im Gegensatz zu den bereits bekannten linearen Strukturen ▪ Modellierung und Implementierung einer kontextbezogenen Anwendung unter Verwendung der Klasse List	*Beispiel*: (Liste) Todo-Liste Es werden Aufgaben verwaltet, die in eine vom Benutzer gewünschte Reihenfolge angeordnet werden. Der Benutzer kann in der Todo-Liste navigieren, bestehende Einträge ändern oder löschen, sowie vor oder hinter einem Eintrag neue Aufgaben einfügen. Ein Test, in dem alle Aufgaben aufgelistet sind, kann ebenfalls abgerufen werden. Die Modellierung und die Implementation er-folgt mit dem generischen Datentyp List. *Materialien*: Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.II (Download LK-Q1-II.4)

5. Vertiefung – Anwendungen von Listen, Stapeln oder Schlangen in mindestens einem weiteren Kontext ▪ Modellierung und Implementierung einer weiteren Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse Queue ▪ Modellierung und Implementierung einer weiteren Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse Stack ▪ Modellierung und Implementierung einer weiteren kontextbezogenen Anwendung unter Verwendung der Klasse List	*Beispiele für Vertiefungen* *Beispiel*: (Schlange) Auslastungssimulation einer Arztpraxis. Bei einer Arztpraxis wird die Auslastung des Arztes während der Sprechstunde simuliert. Eingegeben werden die Öffnungszeiten und die voraussichtliche Anzahl der zu erwartenden Patienten. Die Behandlungsdauern der Patientinnen und Patienten werden zufällig gesetzt. Ergebnis der Simulation sind die Zeiten, die die Arztpraxis nicht ausgelastet ist und die Zeit, die am Ende der Sprechstunde notwendig ist, um alle noch wartenden Patienten zu behandeln. *Beispiel*: (Stapel) Terme in Postfix-Notation Die sog. UPN (umgekehrte polnische Notation) bzw. Postfix-Notation eines Terms setzt den Operator hinter die Operanden. Um den Wert des Terms zu berechnen, kann ein Stack verwendet werden. *Beispiel*: (Liste) Verwaltung einer Todo-Liste mit Prioritäten (Todo-Listenmanager) Es werden Aufgaben verwaltet, die mit einer gewissen Priorität erledigt werden sollen. Es werden drei Prioritätsstufen unterschieden und die Aufgaben so angeordnet, dass sie in passender Reihenfolge stehen.

UV Q1-III: Suchen und Sortieren auf linearen Datenstrukturen sowie Modellierung, Implementierung und Komplexitätsanalyse von Such- und Sortieralgorithmen

Leitfrage:

- Wie kann man gespeicherte Informationen günstig (wieder-)finden?

Vorhabenbezogene Konkretisierungen:

In einem Anwendungskontext werden zunächst Informationen in einer linearen Liste bzw. einem Feld gesucht. Hierzu werden Verfahren entwickelt und implementiert bzw. analysiert und erläutert, wobei neben einem iterativen auch ein rekursives Verfahren thematisiert wird und mindestens ein Verfahren selbst entwickelt und implementiert wird. Die verschiedenen Verfahren werden hinsichtlich Speicherbedarfes und Zahl der Vergleichsoperationen miteinander verglichen.

Anschließend werden Sortierverfahren entwickelt und implementiert (ebenfalls für lineare Listen und Felder). Hierbei soll auch ein rekursives Sortierverfahren entwickelt werden. Die Implementationen von Quicksort sowie dem Sortieren durch Einfügen werden analysiert und erläutert. Falls diese Verfahren vorher schon entdeckt wurden, sollen sie hier wiedererkannt werden. Die rekursive Abarbeitung eines Methodenaufrufs von Quicksort wird grafisch dargestellt.

Abschließend werden verschiedene Sortierverfahren hinsichtlich der Anzahl der benötigten Vergleichsoperationen und des Speicherbedarfs beurteilt.

Zu entwickelnde Kompetenzen: Die Schüler:innen

- analysieren und erläutern Algorithmen und Programme (A),
- beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A),
- beurteilen die Effizienz von Algorithmen unter Berücksichtigung des Speicherbedarfs und der Zahl der Operationen (A),

▪ entwickeln iterative und rekursive Algorithmen unter Nutzung der Strategien „Modularisierung“ , „Teilen und Herrschen“ und „Backtracking“(M), ▪ modifizieren Algorithmen und Programme (I), ▪ implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), ▪ implementieren und erläutern iterative und rekursive Such- und Sortierverfahren unterschiedlicher Komplexitätsklassen (Speicherbedarf und Laufzeitverhalten)(I), ▪ nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), ▪ interpretieren Fehlermeldungen und korrigieren den Quellcode (I), ▪ testen Programme systematisch anhand von Beispielen und mit Hilfe von Testanwendungen (I), ▪ stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D).	
Unterrichtssequenzen	*Beispiele, Medien, Materialien*
1. Suchen von Daten in Listen und Arrays ▪ Lineare Suche in Listen und in Arrays ▪ Binäre Suche in Arrays als Beispiel für rekursives Problemlösen ▪ Untersuchung der beiden Suchverfahren hinsichtlich ihrer Effizienz (Speicherbedarf, Anzahl der Vergleiche)	*Beispiel*: Benutzerverwaltung für einen Fotokopierer Benutzerinnen und Benutzer des Kopierers geben eine vierstellige Ziffernkombination an, um sich zu legitimieren. Über diesen PIN-Code soll auch die Benutzerinnen und Benutzer des Kopierers identifizierbar sein, um auf sein Konto den Kopierverbrauch zu buchen. Die Benutzungsdaten werden in linearen Strukturen verwaltet, die nach den PIN-Nummern sortiert vorliegen. *Materialien*: Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.III - Suchen und Sortieren (Download Q1-III.1)
2. Sortieren in Listen und Arrays - Entwicklung und Implementierung von iterativen und rekursiven Sortierverfahren ▪ Entwicklung und Implementierung eines einfachen Sortierverfahrens für eine Liste ▪ Implementierung eines einfachen Sortierverfahrens für ein Feld ▪ Entwicklung eines rekursiven Sortierverfahren für ein Feld (z.B. Sortieren durch Mischen)	Beispiel: Benutzerverwaltung für einen Fotokopierer (s.o.) Die Benutzungsdaten sollen nach den Kriterien Kopierverbrauch, Namen und PIN-Nummern geordnet ausgegeben werden können. *Materialien*: Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.III - Suchen und Sortieren (Download Q1-III.2)
3. Untersuchung der Effizienz der Sortierverfahren „Sortieren durch direktes Einfügen“ und „Quicksort“ auf linearen Listen ▪ Grafische Veranschaulichung der Sortierverfahren ▪ Untersuchung der Anzahl der Vergleichsoperationen bei beiden Sortierverfahren ▪ Beurteilung der Effizienz der beiden Sortierverfahren, untere Schranke für die Laufzeit von Sortieralgorithmen	*Beispiel*: Test- und Analyseumgebung für Sortieralgorithmen Listen mit unterschiedlich vielen Listenelementen und unterschiedlicher Vorsortierung werden bezüglich der Anzahl der Vergleiche von Listenelementen analysiert. Die untere Schranke für die Laufzeit wird bestimmt. *Materialien*: Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1.III - Suchen und Sortieren (Download Q1-III.3)

UV Q1-IV.1: Modellierung, inkl. Implementierung, und Nutzung relationaler Datenbanken in Anwendungskontexten

Leitfragen:

- Wie können Fragestellungen mit Hilfe einer Datenbank beantwortet werden?
- Wie entwickelt man selbst eine Datenbank für einen Anwendungskontext?

Vorhabenbezogene Konkretisierungen:

Ausgehend von einer vorhandenen Datenbank entwickeln Schüler:innen für sie relevante Fragestellungen, die mit dem vorhandenen Datenbestand beantwortet werden sollen. Zur Beantwortung dieser Fragestellungen wird die vorgegebene Datenbank von den Schüler:innen analysiert, und die notwendigen Grundbegriffe für Datenbanksysteme sowie die erforderlichen SQL-Abfragen werden erarbeitet.

In anderen Anwendungskontexten müssen Datenbanken erst noch entwickelt werden, um Daten zu speichern und Informationen für die Beantwortung von möglicherweise auftretenden Fragen zur Verfügung zu stellen. Dafür ermitteln Schüler:innen in den Anwendungssituationen Entitätstypen, Attribute der zugehörigen Entitäten, Beziehungstypen und Kardinalitäten und stellen diese in Entity-Relationship-Diagramm dar. Entity-Relationship-Modelle werden interpretiert und erläutert, modifiziert und in Datenbankschemata über-führt. Mit Hilfe von SQL-Anweisungen können anschließend im Kontext relevante Informationen aus der Datenbank extrahiert werden.

Ein Entity-Relationship-Diagramm kann auch verwendet werden, um die Entitätstypen inklusive der Attribute ihrer Entitäten und die Be-ziehungstypen in einem vorgegebenen Datenbankschema darzustellen.

An einem Beispiel wird verdeutlicht, dass in Datenbanken Redundanzen unerwünscht sind und Konsistenz gewährleistet sein sollte. Die 1. bis 3. Normalform wird als Gütekriterium für Datenbankentwürfe eingeführt. Datenbankschemata werden hinsichtlich der 1. bis 3. Normalform untersucht und (soweit nötig) normalisiert.

Zu entwickelnde Kompetenzen: Die Schüler:innen

- ermitteln für anwendungsbezogene Problemstellungen Entitäten, zugehörige Attribute, Relationen und Kardinalitäten (M),
- stellen Entitäten mit ihren Attributen und die Beziehungen zwischen Entitäten mit Kardinalitäten in einem Entity-Relationship-Diagramm grafisch dar (D),
- modifizieren eine Datenbankmodellierung (M),
- modellieren zu einem Entity-Relationship-Diagramm ein relationales Datenbankschema (M),
- bestimmen Primär- und Sekundärschlüssel (M),
- implementieren ein relationales Datenbankschema als Datenbank (I),
- analysieren und erläutern eine Datenbankmodellierung (A),
- erläutern die Eigenschaften normalisierter Datenbankschemata (A),
- überprüfen Datenbankschemata auf vorgegebene Normalisierungseigenschaften (D),
- überfuhren Datenbankschemata in die 1. bis 3. Normalform (M),
- ermitteln Ergebnisse von Datenbankabfragen über mehrere verknüpfte Tabellen (D),
- analysieren und erläutern die Syntax und Semantik einer Datenbankabfrage (A),
- verwenden die Syntax und Semantik einer Datenbankabfragesprache, um Informationen aus einem Datenbanksystem zu extrahieren (I),
- erläutern Eigenschaften und Aufbau von Datenbanksystemen unter dem Aspekt der sicheren Nutzung (A).

Unterrichtssequenzen	*Beispiele, Medien, Materialien*
4. Nutzung von relationalen Datenbanken ▪ Aufbau von Datenbanken und Grundbegriffe o Entwicklung von Fragestellungen zur vorhandenen Datenbank o Analyse der Struktur der vorgegebenen Datenbank und Erarbeitung der Begriffe Tabelle, Attribut, Datensatz, Daten-	*Beispiel*: VideoCenter VideoCenter ist die Simulation einer Online-Videothek für den Informatik-Unterricht mit Web Frontend zur Verwaltung der Kunden, der Videos und der Ausleihe. Außerdem ist es möglich direkt SQL-Abfragen einzugeben. Es ist auch möglich, die Datenbank herunterzuladen und lokal zu installie-

typ, Primärschlüssel, Fremdschlüssel, Datenbankschema ▪ SQL-Abfragen o Analyse vorgegebener SQL-Abfragen und Erarbeitung der Sprachelemente von SQL (SELECT (DISTINCT) …FROM, WHERE, AND, OR, NOT) auf einer Tabelle o Analyse und Erarbeitung von SQL-Abfragen auf einer und mehrerer Tabelle zur Beantwortung der Fragestellungen (JOIN, UNION, AS, GROUP BY,ORDER BY, ASC, DESC, COUNT, MAX, MIN, SUM, Arithmetische Operatoren: +, -, *, /, (…), Vergleichsoperatoren: =, <>, >, <, >=, <=, LIKE, BETWEEN, IN, IS NULL) ▪ Vertiefung an einem weiteren Datenbankbeispiel	ren. Unter http://dokumentation.videocenter.schule.de/old/video/index.html (abgerufen: 01. 02. 2016) findet man den Link zu dem VideoCenter-System sowie nähere Informationen. Lesenswert ist auch die dort verlinkte „Dokumentation der Fallstudie" mit didaktischem Material, welches alternativ bzw. ergänzend zu der im Folgenden beschriebenen Durchführung verwendet werden kann. *Beispiel*: Schulbuchausleihe Unter www.brd.nrw.de/lerntreffs/informatik/structure/material/sek2/datenbanken.php (abgerufen: 01. 02. 2016) wird eine Datenbank zur Verfügung gestellt, die Daten einer Schulbuch-Ausleihe enthält (über 1000 Entleiher, 200 Bücher mit mehreren tausend Exemplaren und viele Ausleihvorgänge). Die Datenbank kann in OpenOffice eingebunden werden.
5. Modellierung von relationalen Datenbanken ▪ Entity-Relationship-Diagramm o Ermittlung von Entitäten, zugehörigen Attributen, Relationen und Kardinalitäten in Anwendungssituationen und Modellierung eines Datenbankentwurfs in Form eines Entity-Relationship-Diagramms o Erläuterung und Modifizierung einer Datenbankmodellierung ▪ Entwicklung einer Datenbank aus einem Datenbankentwurf o Modellierung eines relationalen Datenbankschemas zu einem Entity-Relationship-Diagramm inklusive der Bestimmung von Primär- und Sekundärschlüsseln ▪ Redundanz, Konsistenz und Normalformen o Untersuchung einer Datenbank hinsichtlich Konsistenz und Redundanz in einer Anwendungssituation o Überprüfung von Datenbankschemata hinsichtlich der 1. bis 3. Normalform und Normalisierung (um Redundanzen zu vermeiden und Konsistenz zu gewährleisten)	*Beispiel:* Fahrradverleih Der Fahrradverleih BTR (BikesToRent) verleiht unterschiedliche Typen von Fahrrädern diverser Firmen an seine Kunden. Die Kunden sind bei BTR registriert (Name, Adresse, Telefon). BTR kennt von den Fahrradfirmen den Namen und die Telefonnummer. Kunden von BTR können CityBikes, Treckingräder und Mountainbikes ausleihen. *Beispiel:* Reederei Die Datenverwaltung einer Reederei soll in einem Datenbanksystem umgesetzt werden. Ausgehend von der Modellierung soll mit Hilfe eines ER-Modells und eines Datenbankschemas dieser erste Entwurf normalisiert und in einem Datenbanksystem umgesetzt werden. Es schließen sich diverse SQL-Abfragen an, wobei auf die Relationenalgebra eingegangen wird. *Beispiel:* Buchungssystem In dem Online-Buchungssystem einer Schule können die Lehrer Medienräume, Beamer, Laptops, Kameras, usw. für einen bestimmten Zeit-punkt buchen, der durch Datum und die Schul-stunde festgelegt ist. Dazu ist die Datenbank zu modellieren, ggf. zu normalisieren und im Datenbanksystem umzusetzen. Weiter sollen sinnvolle Abfragen entwickelt werden. Unter http://mrbs.sourceforge.net (abgerufen: 30.03. 2014) findet man ein freies Online-Buchungssystem inklusive Demo, anhand derer man erläutern kann, worum es in dem Projekt geht.

	Beispiel: Schulverwaltung In einer Software werden die Schulhalbjahre, Jahrgangsstufen, Kurse, Klassen, Schüler, Lehrer und Noten einer Schule verwaltet. Man kann dann ablesen, dass z.B. Schüler X von Lehrer Y im 2. Halbjahr des Schuljahrs 2011/2012 in der Jahrgangsstufe 9 im Differenzierungsbereich im Fach Informatik die Note „sehr gut“ erhalten hat. Dazu ist die Datenbank zu modellieren, ggf. zu normalisieren und im Datenbanksystem umzusetzen. Weiter sollen sinnvolle Abfragen entwickelt werden und das Thema Datenschutz besprochen werden.

UV Q1-IV.2: Projektorientierte Softwareentwicklung am Beispiel einer Anwendung mit Datenbankanbindung

Leitfrage:

- Wie kann ein komplexeres Java-Projekt unter Verwendung einer Datenbank realisiert werden?

Vorhabenbezogene Konkretisierungen:

Der Schwerpunkt des vorliegenden Unterrichtsvorhabens liegt in der schülerorientierten Erarbeitung eines Java-Projekts, das einer Ver-knüpfung zwischen der Modellierung und Abfrage von Datenbanken sowie der Entwicklung von Problemlösungen mit Hilfe dynamischer Datenstrukturen wie zum Beispiel der linearen Liste oder dem Graphen herstellt. Dabei soll ein möglichst vollständiger Softwareentwicklungszyklus durchlaufen und sowohl die Arbeit mit Datenbanken als auch mit dynamischen Datenstrukturen vertieft bzw. geübt werden.

Dazu wird zunächst durch die Schüler:innen eine lebensweltnahe Problemstellung entwickelt, die sich auch direkt aus den Anforderungen des Schulalltags ergeben und zu einem real einsetzbaren Softwareprodukt führen kann. Dabei könnte es sich zum Beispiel um ein Ausleihsystem für die Schülerbibliothek, eine Datenbank für Fehlstunden oder die Verwaltung von Ergebnissen vom Sport-fest handeln. Da es nicht immer einfach ist, eine so praxisorientierte Problemstellung zu finden, sind aber auch andere Projekte denkbar.

Die Wahl der Problemstellung sollte je nach Lerngruppe einen Schwerpunkt auf die Datenbankmodellierung und Abfrage oder aber auf die Arbeit mit dynamischen Datenstrukturen legen. Soll sich das Projekt auf Datenbanken konzentrieren, ist z.B. ein Quizspiel denkbar, das schulweit Fragen aus einer zentralen Datenbank abruft, Ergebnisse und Ranglisten aller Spielerinnen und Spieler verwaltet und ggf. diese auch gegeneinander antreten lässt, indem ihnen die gleichen Fragen gestellt werden und sie somit in einen direkten Vergleich treten. Will man den Schwerpunkt des Projekts auf dynamische Datenstrukturen legen, wäre anknüpfend an das Unterrichtsvorhaben „Q1-II“ die Entwicklung eines Routenplaners, zum Beispiel basierend auf einer Datenbank mit echten Daten des deutschen Autobahnnetzes, ein geeignetes Projekt.

Der Aufbau dieses Projekts orientiert sich an einer vereinfachten Version des Wasserfallmodells der Softwareentwicklung, bestehend aus Analyse, Modellierung, Implementierung und Test (und ggf. auch Installation und Schulung). Insbesondere die Modellierung und Implementierung beziehen sich dabei gegebenenfalls auf die Entwicklung einer geeigneten Datenbank und deren Abfrage und Manipulation mit SQL und auf die Entwicklung eines entsprechenden Java-Programms, das die Datenbank nutzt. Bei Projekten, die Datenbanken mit einer großen Anzahl von Datensätzen erfordern, sollten die Datensätze in geeigneter elektronischer Form vorgegeben werden. Zur Ab-frage und Manipulation der Daten kommen didaktisch vereinfachte Klassen zur Einbindung einer Datenbank in ein Java-Programm zum Einsatz (siehe Abiturklassen zu Datenbanken).

Um den normalerweise hohen Zeitbedarf für Projektarbeiten möglichst gering zu halten, sollte arbeitsteilig vorgegangen werden und auch auf das Prinzip des Prototypings, d.h. die Vervollständigung eines vom Lehrenden vorgegebenen Teilprogramms, zurückgegriffen werden. So sollte zum Beispiel die zeitaufwändige, aber wenig ergiebige Implementation einer grafischen Benutzungsoberfläche – nicht jedoch deren

Design – den Schüler:innen abgenommen werden.

Zu entwickelnde Kompetenzen: Die Schüler:innen

- ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M),
- modellieren Klassen mit ihren Attributen, Methoden und Assoziationsbeziehungen unter Angabe von Multiplizitäten (M),
- ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare und nichtlineare Datensammlungen zu (M),
- ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M),
- stellen die Kommunikation zwischen Objekten grafisch dar (D),
- stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D),
- dokumentieren Klassen (D),
- analysieren und erläutern objektorientierte Modellierungen (A),
- implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I),
- ermitteln für anwendungsbezogene Problemstellungen Entitäten, zugehörige Attribute, Relationen und Kardinalitäten (M),
- stellen Entitäten mit ihren Attributen und die Beziehungen zwischen Entitäten mit Kardinalitäten in einem Entity-Relationship-Diagramm grafisch dar (D),
- modellieren zu einem Entity-Relationship-Diagramm ein relationales Datenbankschema (M),
- bestimmen Primär- und Sekundärschlüssel (M),
- implementieren ein relationales Datenbankschema als Datenbank (I),
- analysieren und erläutern eine Datenbankmodellierung (A),
- erläutern die Eigenschaften normalisierter Datenbankschemata (A),
- überprüfen Datenbankschemata auf vorgegebene Normalisierungseigenschaften (D),
- überführen Datenbankschemata in die 1. bis 3. Normalform (M),
- analysieren und erläutern Algorithmen und Programme (A),
- modifizieren Algorithmen und Programme (I),
- testen Programme systematisch anhand von Beispielen und mit Hilfe von Testanwendungen (I),
- ermitteln Ergebnisse von Datenbankabfragen über mehrere verknüpfte Tabellen (D),
- nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I),
- beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A),
- interpretieren Fehlermeldungen und korrigieren den Quellcode (I),
- analysieren und erläutern die Syntax und Semantik einer Datenbankabfrage (A),
- verwenden die Syntax und Semantik einer Datenbankabfragesprache, um Informationen aus einem Datenbanksystem zu extrahieren (I),nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung von Daten, zur Organisation von Arbeitsabläufen sowie zur Verteilung und Zusammenführung von Arbeitsanteilen (K),
- wenden didaktisch orientierte Entwicklungsumgebungen zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I),
- entwickeln mit didaktisch orientierten Entwicklungsumgebungen einfache Benutzungsoberflächen zur Kommunikation mit einem Informatiksystem (M),
- erläutern Eigenschaften und Aufbau von Datenbanksystemen unter dem Aspekt der sicheren Nutzung (A),
- untersuchen und bewerten anhand von Fallbeispielen Auswirkungen des Einsatzes von Informatiksystemen sowie Aspekte der Sicherheit von Informatiksystemen, des Datenschutzes und des Urheberrechts (A).

Unterrichtssequenzen	*Beispiele, Medien, Materialien*
1. Analyse einer lebensweltnahen Problemstellung im Hinblick auf die Entwicklung eines Java-Programms mit Datenbankanbindung	*Beispiel*: Quizspiel Verwaltung von Quizfragen, Antworten und Ranglisten

▪ Entwicklung einer Programmidee ▪ Analyse des Problembereichs ▪ Entwicklung eines Anforderungskatalogs für das zu entwickelnde Programm	*Beispiel*: Navigationssystem Ermittlung von kürzesten Wegen im deutschen Autobahnnetz *Beispiel*: Verwaltung der Schülerbücherei Verwaltungsprogramm für das Einpflegen und Ausleihe von Büchern der Schülerbibliothek *Beispiel*: Fehlstundenverwaltung Verwaltungsprogramm für die Fehlstunden von Schüler:innen *Beispiel*: Sportfestverwaltung Verwaltung von Aufgaben, Sportereignissen und Ergebnissen des Schulsportfestes *Beispiel*: Materialverwaltung Materialien für Vertretungsstunden sollen verwaltet werden.
2. Modellierung einer datenbankgestützten Problemlösung unter Berücksichtigung des MVC-Prinzips ▪ Strukturierung nach dem MVC-Prinzip ▪ Modellierung der Datenbank (ER-Diagramm, Datenbankschema) ▪ Modellierung der Kontrollklassen und Entwicklung von SQL-Anweisungen entsprechend des Anforderungskatalogs ▪ Modellierung einer grafischen Benutzungsoberfläche	*Materialien*: Ergänzungsmaterialien zum Lehrplan-navigator – Dokumentation der Abitur-klassen zur Datenbank-anbindung (Download LK-Q1-VI.1) Ergänzungsmaterialien zum Lehrplan-navigator – Projekt Navigationssystem (Download LK-Q1-VI.2) Ergänzungsmaterialien zum Lehrplan-navigator – Projekt Quizspiel (Download LK-Q1-VI.3)
3. Umsetzung des Softwareprojektes Grafische Veranschaulichung der Sortierverfahren ▪ Umsetzung der Datenbank in einem Datenbankmanagementsystem ▪ Implementierung der Kontrollklassen mit Anbindung an die Datenbank unter Verwendung didaktischer Klassen ▪ Integration in den Prototypen der Benutzeroberfläche	*Materialien*: Ergänzungsmaterialien zum Lehrplan-navigator – Dokumentation der Abitur-klassen zur Datenbank-anbindung (Download LK-Q1-VI.1) Ergänzungsmaterialien zum Lehrplan-navigator – Projekt Navigationssystem (Download LK-Q1-VI.2) Ergänzungsmaterialien zum Lehrplan-navigator – Projekt Quizspiel (Download LK-Q1-VI.3)
4. *Optional*: Einführung und Evaluation ▪ Installation und Test des Endprodukts im konkreten Anwendungskontext ▪ Schulung von Anwendern an der neuen Software ▪ Evaluation des Projekts	

UV Q1-V: Grundlagen der Netzwerkkommunikation und Sicherheit und Datenschutz in Netzwerken sowie Modellierung und Implementierung von Client-Server-Anwendungen in kontextbezogenen Problemstellungen

Leitfragen:

- Wie werden Daten in Netzwerken übermittelt?
- Was sollte man in Bezug auf die Sicherheit beachten?
- Wie entwickelt man ein Client-Server-System im Anwendungskontext?
- Welche Algorithmen sind zu implementieren?

Vorhabenbezogene Konkretisierungen:

Zunächst werden die Grundlagen von Datenübertragung in Netzwerken erarbeitet. Den Einstieg bildet ein Vergleich der Kommunikation in Netzen mit der physikalischen Zustellung von Sendungen durch Postunternehmen, der zu einem Schichtenmodell als Strukturierungsprinzip für Netzwerkkommunikation führt. Im Anschluss werden von den Schüler:innen Antworten auf grundsätzliche Herausforderungen im Bereich Netzwerkkommunikation erarbeitet: die Wahl einer geeigneten Codierung, Vor- und Nachteile verschiedener Topologien, Adressierung/Routing in IP-Netzen sowie die Gestaltung von Protokollen für die Anwendungsebene.

In einer zweiten Phase werden zunächst Clients für vorhandene Server-Dienste entwickelt. Darauf aufbauend können anschließend eigene Server modelliert und implementiert werden.

In einer dritten Phase modellieren und implementieren die die Schüler:innen schließlich ein Client-Server-System. Dieses macht u.a. ein Verständnis von Nebenläufigkeit notwendig, da ein Server parallel Nachrichten von mehreren Clients empfangen und verarbeiten können muss.

Zum Abschluss der Reihe wird auch die Frage der menschlichen Kommunikation in Netzwerken aufgenommen; es werden die Themen Datenschutz, Urheberrecht und moralische Verantwortung systematisiert und vertieft. Im Bereich Datenschutz werden grundlegende Begriffe (z. B. personenbezogene Daten, informationelle Selbstbestimmung, Datensparsamkeit usw.) eingeführt und an weiteren Fallbeispielen verdeutlicht. Im Bereich Urheberrecht sollte mindestens ein verbreitetes Lizenzsystem thematisiert werden (z. B. Creative-Commons-Lizenzen) und anhand von Beispielen verdeutlicht werden. Die Erarbeitung der Schwerpunkte Datenschutz, Urheberrecht und moralische Verantwortung kann dabei sequenziell mit der gesamten Lerngruppe oder parallel in zieldifferenten Teilgruppen erfolgen. In beiden Fällen müssen die Ergebnisse zusammengefasst in der gesamten Lerngruppe gesichert werden.

Zu entwickelnde Kompetenzen: Die Schüler:innen

- beschreiben und erläutern Netzwerk-Topologien, die Client-Server-Struktur und Protokolle sowie ein Schichtenmodell in Netzwerken (A),
- analysieren und erläutern Algorithmen und Programme (A),
- analysieren und erläutern Protokolle zur Kommunikation in einem Client-Server-Netzwerk (A),
- entwickeln und erweitern Protokolle zur Kommunikation in einem Client-Server-Netzwerk (M),
- modifizieren Algorithmen und Programme (I)
- ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und Beziehungen (M),
- modellieren Klassen mit ihren Attributen, Methoden und Assoziationsbeziehungen unter Angabe von Multiplizitäten (M),
- ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare und nichtlineare Datensammlungen zu (M),
- analysieren und erläutern objektorientierte Modellierungen (A),
- stellen Klassen und ihre Beziehungen grafisch dar (D),
- implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I).

Unterrichtssequenzen	*Beispiele, Medien, Materialien*
1. Grundlagen der Datenübertragung in Netzwerken ▪ Schichtenmodell (Erarbeitung eines standardisierten Schichtenmodells für Netzwerkkommunikation) ▪ Grundlagen der Codierung (Erarbeitung einer eigenen, vereinfachten Codierung für die Bitübertragung) ▪ Topologien (Erarbeitung und Vergleich ausgewählter Netzwerktopologien) ▪ Routing (Analyse von Grundlagen der Adressierung in IP-Netzwerken) ▪ Protokolle (Erarbeitung des beispielhaften Aufbaus eines Protokolls auf der Anwendungsschicht)	*Material*: Aufgabensammlung Anhand einer Sammlung von Aufgabenblättern (teils inkl. implementierter Begleitwerk-zeuge) erarbeiten die Schüler:innen im Anwendungskontext Grundlagen der Inhaltspunkte „OSI-Referenzmodell“, „Codierung“, „Topologien“, „Routing“ und „Protokolle“
2. Analyse, Modellierung und Implementierung von Netzwerkanwendungen in Client-Server-Struktur ▪ Nutzung einfacher Server-Dienste mittels Client o Modellierung und Implementierung von Clients für einfach Serverdienste ▪ Anbieten von Diensten mittels Server o Analyse vorgegebener Implementationen einfacher Server o Modellierung und Implementierung eigener Server	*Beispiel*: Echo- bzw. Daytime-Clients und –Server sowie eigene Erweiterungen Anhand der Echo- und Daytime-Dienste (z.B. lokal im Schulnetz durch den Lehren-den zur Verfügung gestellt) erarbeiten die Schüler:innen zunächst den die Funktionsweise bzw. den Aufbau einfacher Clients und verwenden dabei zunächst die Klasse Connection, später die Klasse Client. In einem zweiten Schritt implementieren die Schüler:innen Daytime- und Echo-Client bzw. Erweiterungen/Abwandlungen derselben (ggf. mit Steigerung des Interaktionsgrades) selbst.
3. Entwicklung eines vollständigen Client-Server-Systems ▪ Protokollentwurf, Dialogorientierung ▪ Modellierung mittels Entwurfs- und Implementationsdiagramm ▪ Bedeutung von Nebenläufigkeit ▪ Implementierung	*Beispiel*: Messenger-Dienst Abschließend entwickeln die Schüler:innen ein Client-Server-System zum Versenden von Nachrichten zwischen einzelnen Rechnern (einfacher Messenger), basierend auf selbst gewählten „Nicknames“.
4. Erarbeitung grundlegender Positionen (ggf. in zieldifferenten Gruppen) ▪ Erarbeitung von Grundideen des Datenschutzes (z.B. personenbezogene Daten, informationelle Selbstbestimmung, Datensparsamkeit usw.) ▪ Erarbeitung von Grundideen des Urheberrechts anhand eines verbreiteten Lizenzsystems ▪ Erarbeitung eines einfachen moralischen Bewertungsmaßstabes ▪ Anwendung auf Fallbeispiele	*Materialien*: Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.I – Grundlagen „Informatik, Mensch und Gesellschaft“ (Download LK-Q2-I.2) Beispiel: Creative-Commons-Lizenzen URL: http://de.creativecommons.org (Abgerufen am 19.06.2016)

UV Q2-I: Modellierung und Implementierung von dynamischen, nicht-linearen Datenstrukturen sowie

von Anwendungen mit dynamischen, nichtlinearen Datenstrukturen

Leitfragen:

- Wie können Daten im Anwendungskontext mit Hilfe von Graphen und binärer Baumstrukturen verwaltet werden?
- Wie kann in einem Graphen ein beliebiger, alle oder der kürzeste Weg zwischen zwei Knoten effizient gefunden werden?
- Welche Kanten müssen in einem zusammenhängenden Graphen mindestens verbleiben, sodass dieser bei minimaler Summe der Kantengewichte weiterhin zusammenhängend ist?
- Wie kann ich mit einem Binärbaum Daten sortiert verwalten?

Vorhabenbezogene Konkretisierungen:

Anhand von Beispielen für Graphen werden grundlegende Begriffe eingeführt und die beiden Darstellungsformen Adjazenzmatrix und Adjazenzliste erarbeitet. Die Funktionalität der Methoden der Klassen Graph, Vertex und Edge aus den Vorgaben für das Zentralabitur NRW wird erarbeitet. Die Fragestellung nach der Suche eines oder aller Wege zwischen zwei Knoten in einem Graphen motiviert die Erarbeitung von Algorithmen zur Tiefen- und Breitensuche, mit denen Graphen systematisch durchsucht werden können. Anschließend werden anhand von Anwendungskontexten Algorithmen für die Bestimmung kürzester Wege in einem Graphen sowie zur Konstruktion von minimalen Spannbäumen modelliert und zum Teil auch implementiert.

Die Datenstruktur Binärbaum mit den notwendigen Begrifflichkeiten wird als Spezialfall eines Graphen anhand von Anwendungskontexten erarbeitet. Die entsprechende Klasse BinaryTree<ContentType> der Vorgaben für das Zentralabitur NRW wird zur Modellierung und Implementierung verschiedener Problemstellungen verwendet. Unter anderem sollen die verschiedenen Baumtraversierungen (Pre-, Post- und In-Order) implementiert werden.

Mithilfe einer anwendungsorientierten Problemstellung werden die Operationen der Datenstruktur Suchbaum thematisiert und die Klasse BinarySearchTree<ContentType> (der Materialien für das Zentralabitur in NRW) modelliert und zumindest partiell implementiert.

Zu entwickelnde Kompetenzen: Die Schüler:innen

- erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A),
- analysieren und erläutern Algorithmen und Programme (A),
- beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A),
- beurteilen die Effizienz von Algorithmen unter Berücksichtigung des Speicherbedarfs und der Zahl der Operationen (A),
- ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M),
- ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare und nichtlineare Datensammlungen zu (M),
- modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M),
- verwenden bei der Modellierung geeigneter Problemstellungen die Möglichkeiten der Polymorphie (M),
- entwickeln iterative und rekursive Algorithmen unter Nutzung der Konstruktionsstrategien „Modularisierung“ und „Teilen und Herrschen“ und „Backtracking“(M),
- entwickeln mit didaktisch orientierten Entwicklungsumgebungen einfache Benutzungsoberflächen zur Kommunikation mit einem Informatiksystem (M),
- implementieren Operationen dynamischer (linearer oder nichtlinearer) Datenstrukturen (I),
- implementieren und erläutern iterative und rekursive Such- und Sortierverfahren unterschiedlicher Komplexitätsklassen (Speicherbedarf und Laufzeitverhalten) (I),
- nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I),
- interpretieren Fehlermeldungen und korrigieren den Quellcode (I),
- testen Programme systematisch anhand von Beispielen und mithilfe von Testanwendungen(I),

- wenden didaktisch orientierte Entwicklungsumgebungen zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I),
- stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D),
- stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D),
- nutzen bereitgestellte Informatiksysteme und das Internet reflektiert zur Erschließung, Aufbereitung und Präsentation fachlicher Inhalte (D),
- nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung von Daten, zur Organisation von Arbeitsabläufen sowie zur Verteilung und Zusammenführung von Arbeitsanteilen (K).

Unterrichtssequenzen	*Beispiele, Medien, Materialien*
1. Analyse von Graphen in verschiedenen Kontexten ▪ Grundlegende Begriffe (Graph, gerichtet – ungerichtet, Knoten, Kanten, Kantengewicht) ▪ Aufbau und Darstellung von Graphen anhand von Graphenstrukturen in verschiedenen Kontexten (Adjazenzmatrix, Adjazenzliste)	*Beispiel*: „Das Haus vom Nikolaus“ Das Haus vom Nikolaus ist das bekannteste Beispiel für einen Graphen, für den ein Eulerweg, aber kein Eulerkreis existiert. An-hand dieses Beispiels werden die Grundbegriffe der Graphentheorie sowie die Darstellung eines Graphen als Adjazenzmatrix ein-geführt. *Beispiel:* Soziale Netzwerke Da es in dem Graph eines sozialen Netz-werks im Verhältnis zu den Knoten nur wenige Kanten gibt, bietet sich dieses Beispiel zur Einführung der Darstellung eines Graphen in Form von Adjazenzlisten an. *Materialien*: Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben LK-Q1.V (Download LK-Q1.V.1)
2. Die Datenstruktur Graph im Anwendungskontext unter Nutzung der Klassen Graph, Vertex und Edge. ▪ Erarbeitung der Klassen Graph, Vertex und Edge und beispielhafte Anwendung der Operationen ▪ Bestimmung von Wegen in Graphen im Anwendungskontext (Tiefensuche, Breitensuche) ▪ Bestimmung von kürzesten Wegen in Graphen im Anwendungskontext (Backtracking, Dijkstra). ▪ Bestimmung von minimalen Spannbäumen eines Graphen im Anwendungskontext.	*Beispiel*: Soziale Netzwerke Ausgehend von dem Problem der Berechnung der Dichte eines sozialen Netzwerkes werden die Funktionalitäten der Methoden der Klassen Graph, Vertex und Edge erarbeitet und erste Beispiele modelliert und implementiert: Konstruktion eines Beispielgraphen Anzahl der Knoten Summe der Kantengewichte Anzahl der Nachbarn eines Knotens *Beispiel*: Wegsuche Ausgehend von dem Problem der Suche eines beliebigen Weges zwischen zwei Knoten in einem Graphen wird der Backtracking-Algorithmus zur Tiefensuche erarbeitet. Durch Wegfall der Abbruchbedingung „Zielknoten gefunden“ lassen sich mit dem Algorithmus alle Wege zwischen Start und Zielknoten finden. Als Alternative wird der Algorithmus zur Breitensuche erarbeitet, der als Ergebnis eine Liste aller Knoten, die auf dem Weg vom Start- zum Zielknoten gefunden wurde, zurückgibt. Ausgehend vom Zielknoten kann durch Vorgängersuch in dieser Liste ein

	Weg vom Start- zum Zielknoten gefunden werden. Damit wird das Verfahren beim Dijkstra-Algorithmus vorbereitet. *Beispiel*: Kürzeste Wege Ausgehend vom Backtracking-Algorithmus zur Bestimmung aller Wege von einem Start- zu einem Zielknoten in einem Graphen wird ein Algorithmus zur Bestimmung des kürzesten Weges erarbeitet. Aufwandbetrachtungen führen zu der Frage nach einem effizienteren Algorithmus. Der Dijkstra-Algorithmus kann durch geschickte Aufgabenstellung, ggf. unter Einbeziehung des in den Materialien enthaltenen Programms GraphTool von der Lerngruppe selbstständig erarbeitet und auf mehrere Beispiele angewandt werden. Die Implementierung erfolgt in der Lerngruppe arbeitsteilig unter Vorgabe einer Benutzungsoberfläche. Der Vergleich der beiden Algorithmen unter Effizienzaspekten ist Bestandteil des Unterrichts. *Beispiel*: Versorgungsnetz Die Problemstellung, Verbraucher eines Dorfes möglichst kostengünstig an ein Versorgungsnetz (Kabel, Gas, Telefon) anzuschließen, motiviert die Behandlung des minimalen Spannbaumes eines Graphen. Die Definition eines Baumes und eines Spannbaumes als Spezialfälle von Graphen bereiten die nächste Unterrichtssequenz vor. *Materialien*: Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben LK-Q1.V (Download LK-Q1.V.2) (Download Q2-I.2)
3. Die Datenstruktur Binärbaum als Spezialfall eines Graphen im Anwendungskontext unter Nutzung der Klasse BinaryTree<ContentType> ▪ Definition eines Binärbaums und grundlegende Begriffe ▪ Erarbeitung der Klasse BinaryTree<ContentType> und beispielhafte Anwendung der Operationen ▪ Implementierung der Traversierung eines Binärbaums im Pre-, In- und Postorderdurchlauf ▪ Modellierung und Implementierung einer Anwendung unter Verwendung der Datenstruktur Binärbaum	*Beispiel*: MorseBaum Morse hat Buchstaben als Folge von Punkten und Strichen codiert. Diese Codierungen können in einem Binärbaum dargestellt werden, sodass ein Übergang zum linken Teilbaum einem Punkt und ein Übergang zum rechten Teilbaum einem Strich entspricht. Anhand dieses Beispiels werden die Definition eines Binärbaums als Spezialfall eines Graphen und eine rekursive Definition erarbeitet. Die Methoden der generischen Klasse BinaryTree<ContentType> eingeführt und zur Implementation des Morsecodierers und -dekodierers genutzt. Wenn man bei der Wurzel startet und durch Übergänge zu linken oder rechten Teilbäumen einen Pfad zum gewünschten Buchstaben sucht, erhält man den Morsecode des Buchstabens. Im Leistungskurs wird auch ein rekursiver Algorithmus zur Dekodie-

	rung von Morsezeichen entwickelt. *Beispiel*: Termbaum Arithmetische Ausdrücke werden in einem Binärbaum dargestellt. Mit den Traversierungsalgorithmen lassen sich die Terme in Pre-, Post- und In-Order-Darstellung ausgeben. Gegebenenfalls kann ein Interpreter für einen arithmetischen Term mit den vier Grundrechenarten entwickelt werden. Im Unterrichtsvorhaben (Q2-III) kann das Beispiel unter dem Thema „Parsen eines einfachen Terms und die Erzeugung eines Termbaums“ wieder aufgegriffen werden. *Beispiel*: Infomatikerbaum als Binärbaum In einem binären Baum werden die Namen und die Geburtsdaten von Informatikern lexikographisch geordnet abgespeichert. Alle Namen, die nach dieser Ordnung vor dem Namen im aktuellen Teilbaum stehen, sind in dessen linkem Teilbaum, alle die nach dem Namen im aktuellen Teilbaum stehen, sind in dessen rechtem Teilbaum. (Dies gilt für alle Teilbäume.) Folgende Funktionalitäten werden benötigt: Einfügen der Informatiker-Daten in den Baum Suchen nach einem Informatiker über den Schlüssel Name Ausgabe des kompletten Datenbestandes in nach Namen sortierter Reihenfolge Anhand des Beispiels werden die Eigenschaften und die Methoden eines binären Suchbaums entwickelt und implementiert. Materialien: Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben LK-Q1.V (Download LK-Q1-V.3)
4. Erarbeitung, Implementierung und Verwendung der Datenstruktur binärer Suchbaum im Anwendungskontext ▪ Erarbeitung der Eigenschaften eines binären Suchbaums im Anwendungskontext ▪ Erarbeitung der Attribute und Methoden der generischen Klasse BinarySearchTree<ContentType> und des Interfaces ComparableContent ▪ Implementierung des Konstruktors und der Methode search der Klasse BinarySearchTree<ContentType> ▪ Implementierung eines Anwendungsbeispiels einschließlich der sortierten Ausgabe eines binären Suchbaumes	*Beispiel*: Informatikerbaum binärer Such-baum Das Beispiel wird wieder aufgegriffen und diesmal mit der Klasse BinarySearchTree<ContentType> implementiert. Durch Modifikation der implementierten Methoden des Interfaces ComparableContent wird der Suchbaum nach den Geburtsdaten der Informatiker sortiert. *Beispiel*: Buchindex Als weiteres Anwendungsbeispiel, das mehrere dynamische Datenstrukturen miteinander verknüpft, soll ein Programm modelliert und implementiert werden, das das Stich-wortregister eines Buches verwaltet. Die Wörter werden in einem binären Suchbaum verwaltet, die zugehörigen Seitenzahlen als lineare Listen.

	Materialien: Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben LK-Q1.VI (Download LK-Q1.V.4)

UV Q2-II: Endliche Automaten, Kellerautomaten und formale Sprachen sowie Modellierung und Implementierung eines Parsers zu einer formalen Sprache

Leitfragen:

- Wie kann man (endliche) Automaten genau beschreiben?
- Wie können endliche Automaten und Kellerautomaten (in alltäglichen Kontexten oder zu informatischen Problemstellungen) modelliert werden?
- Wie können Sprachen durch Grammatiken beschrieben werden?
- Welche Zusammenhänge gibt es zwischen formalen Sprachen, Automaten und Grammatiken?
- Wie kann ein Parser für eine einfache formale Sprache entwickelt werden?

Vorhabenbezogene Konkretisierungen:

Anhand kontextbezogener Beispiele werden endliche Automaten entwickelt, untersucht und modifiziert. Dabei werden verschiedene Darstellungsformen für endliche Automaten ineinander überführt und die akzeptierten Sprachen endlicher Automaten ermittelt. An einem Beispiel wird ein nichtdeterministischer Akzeptor als Alternative zu einem entsprechenden deterministischen Akzeptor eingeführt. Auch die Umwandlung eines nichtdeterministischen Automaten in einen deterministischen Automaten wird thematisiert.

Anhand kontextbezogener Beispiele werden Grammatiken regulärer Sprachen entwickelt, untersucht und modifiziert. Der Zusammen-hang zwischen regulären Grammatiken und endlichen Automaten wird verdeutlicht durch die Entwicklung von allgemeinen Verfahren zur Erstellung einer regulären Grammatik für die Sprache eines gegebenen endlichen Automaten bzw. zur Entwicklung eines endlichen Automaten, der genau die Sprache einer gegebenen regulären Grammatik akzeptiert. Zu einer einfachen regulären Sprache wird ein Parser in Form eines Java-Programms entwickelt.

Auch nicht-reguläre Grammatiken werden untersucht, entwickelt oder modifiziert. An einem Beispiel werden die Grenzen endlicher Au-tomaten ausgelotet. Mit Blick auf diese Einschränkungen endlicher Automaten wird die Idee eines Automaten mit Speicher thematisiert und zu einem Kellerautomaten weiterentwickelt.

Zu entwickelnde Kompetenzen: Die Schüler:innen

- analysieren und erläutern die Eigenschaften endlicher Automaten und Kellerautomaten einschließlich ihres Verhaltens auf bestimmte Eingaben (A),
- analysieren und erläutern Grammatiken regulärer und kontextfreier Sprachen (A),
- erläutern die Grenzen endlicher Automaten und regulärer Sprachen im Anwendungszusammenhang (A),
- ermitteln die formale Sprache, die durch eine Grammatik erzeugt wird (A),
- entwickeln und modifizieren zu einer Problemstellung endliche Automaten oder Kellerautomaten (M),
- entwickeln zur akzeptierten Sprache eines Automaten die zugehörige Grammatik (M),
- entwickeln zur Grammatik einer regulären oder kontextfreien Sprache einen zugehörigen endlichen Automaten oder einen Kellerautomaten (M),
- modifizieren Grammatiken regulärer und kontextfreier Sprachen (M),
- entwickeln zu einer regulären oder kontextfreien Sprache eine Grammatik, die die Sprache erzeugt (M),
- stellen endliche Automaten in Tabellen oder Graphen dar und überführen sie in die jeweils andere Darstellungsform (D),
- ermitteln die Sprache, die ein endlicher Automat oder ein Kellerautomat akzeptiert (D),
- beschreiben an Beispielen den Zusammenhang zwischen Automaten und Grammatiken (D),

▪ entwickeln iterative und rekursive Algorithmen unter Nutzung der Strategien „Modularisierung“, „Teilen und Herrschen“ und „Backtracking“ (M).	
Unterrichtssequenzen	*Beispiele, Medien, Materialien*
1. Endliche Automaten ▪ Erarbeitung der formalen Beschreibung eines endlichen Automaten auf der Grundlage von Automaten in bekannten Kontexten ▪ Untersuchung, Darstellung und Entwicklung endlicher Automaten ▪ Umwandlung nichtdeterministischer endlicher Automaten in deterministische endliche Automaten	*Beispiele*: Cola-Automat, Geldspielautomat, Roboter, Zustandsänderung eines Objekts „Auto“, Akzeptor für bestimmte Zahlen, Akzeptor für Teilwörter in längeren Zeichenketten, Akzeptor für Terme *Materialien*: Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.III (Download LK-Q2-III.1)
2. Untersuchung und Entwicklung von Grammatiken regulärer Sprachen ▪ Erarbeitung der formalen Darstellung regulärer Grammatiken ▪ Untersuchung, Modifikation und Entwicklung von Grammatiken ▪ Entwicklung von endlichen Automaten zum Erkennen regulärer Sprachen die durch Grammatiken gegeben werden ▪ Entwicklung regulärer Grammatiken zu endlichen Automaten ▪ Entwicklung eines Parsers für eine einfache reguläre Sprache	*Beispiele*: reguläre Grammatik für Wörter mit ungerader Parität, Grammatik für Wörter, die bestimmte Zahlen repräsentieren, Satzgliederungsgrammatik *Materialien*: Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.III (Download LK-Q2-III.2)
3. Grenzen endlicher Automaten	*Beispiele*: Klammerausdrücke, a^nb^n im Vergleich zu $(ab)^n$ Materialien: Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.III (Download LK-Q2-III.3)
4. Entwicklung eines Kellerautomaten als Antwort auf die Grenzen endlicher Automaten ▪ Erweiterung eines DEA um eine einzel-ne Speichervariable zum Zählen von Eingabezeichen (z.B. Klammern) und Problematisierung dieses Ansatzes ▪ Entwicklung eines Automaten mit Kellerspeicher ▪ Anwendung eines Kellerautomaten zur Syntaxüberprüfung auf Grundlage von nichtregulären Grammatiken ▪ Implementierung eines Kellerautomaten zur Syntaxüberprüfung (Backtracking)	*Materialien*: Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.III (Download LK-Q2-III.4)

UV Q2-III: Prinzipielle Arbeitsweise eines Computers inkl. Modellierung und Implementierung eines Scanners, Parsers und Interpreters für eine einfache maschinennahe Sprache sowie Grenzen der Auto-

matisierung

Leitfragen:

- Was sind die strukturellen Hauptbestandteile eines Computers und wie kann man sich die Ausführung eines maschinenahen Programms mit diesen Komponenten vorstellen?
- Wie werden Programme aus höheren Programmiersprachen für den Computer verständlich und wie werden sie in eine tiefere Sprachebene übersetzt und interpretiert?

Vorhabenbezogene Konkretisierungen:

Anhand einer von-Neumann-Architektur und einem maschinennahen Programm wird die prinzipielle Arbeitsweise von Computern verdeutlicht. Grundlegenden Begrifflichkeiten bei der maschinellen Übersetzung von einer Hochsprache in eine maschinenverständliche Sprache werden definiert, veranschaulicht und zum Vorwissen aus dem Unterrichtsvorhaben Q2-III in Beziehung gesetzt.

Ausgehend von einer einfachen formalen Sprache (z. B. eine Konstruktionssprache für geometrische Figuren) werden die Bestandteile eines Compilers dargestellt:

Der Scanner eines Compilers wird in Form eines endlichen Automaten modelliert und implementiert. Die Begriffe Symboltabelle und Tokenliste werden inhaltlich gefüllt. Die dem Parser des Compilers zugrunde liegende Grammatik wird in Form einer regulären oder kontextfreien Grammatik definiert und zugehörige Parser-Methoden werden implementiert. Zum Abschluss wird ein Interpreter-Modul entwickelt, welches die einfache formale Sprache in eine andere Sprachebene übersetzt.

Zu entwickelnde Kompetenzen: Die Schüler:innen

- erläutern die Ausführung eines einfachen maschinennahen Programms sowie die Datenspeicherung auf einer „Von-Neumann-Architektur“ (A),
- ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M),
- modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (M),
- modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M),
- verwenden bei der Modellierung geeigneter Problemstellungen Möglichkeiten der Polymorphie (M),
- ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M),
- stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D),
- analysieren und erläutern objektorientierte Modellierungen (A),
- implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I),
- analysieren und erläutern Algorithmen und Programme (A),
- modifizieren Algorithmen und Programme (I),
- entwickeln iterative und rekursive Algorithmen unter Nutzung der Strategien „Modularisierung“ und „Teilen und Herrschen“ und „Backtracking“ (M),
- implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I),
- testen Programme systematisch anhand von Beispielen und mit Hilfe von Testanwendungen (I),
- analysieren und erläutern die Eigenschaften endlicher Automaten und Kellerautomaten einschließlich ihres Verhaltens bei bestimmten Eingaben (A),
- ermitteln die Sprache, die ein endlicher Automat oder ein Kellerautomat akzeptiert (D),
- entwickeln und modifizieren zu einer Problemstellung endliche Automaten oder Kellerautomaten (M),
- analysieren und erläutern Grammatiken regulärer und kontextfreier Sprachen (A),
- modifizieren Grammatiken regulärer und kontextfreier Sprachen (M),
- ermitteln die formale Sprache, die durch eine Grammatik erzeugt wird (A),
- entwickeln zu einer regulären oder kontextfreien Sprache eine Grammatik, die die Sprache erzeugt (M),

▪ modellieren und implementieren Scanner, Parser und Interpreter zu einer gegebenen regulären Sprache (I), ▪ nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung von Daten, zur Organisation von Arbeitsabläufen sowie zur Verteilung und Zusammenführung von Arbeitsanteilen (K), ▪ untersuchen und beurteilen Grenzen des Problemlösens mit Informatiksystemen (A).	
Unterrichtssequenzen	*Beispiele, Medien, Materialien*
1. Von-Neumann-Architektur und die Ausführung maschinennaher Programme ▪ prinzipieller Aufbau einer von Neumann-Architektur mit CPU, Rechenwerk, Steuerwerk, Register und Hauptspeicher ▪ einige maschinennahe Befehlen und ihre Repräsentation in einem Binär-Code, der in einem Register gespeichert werden kann ▪ Analyse und Erläuterung der Funktionsweise eines einfachen maschinennahen Programms	*Beispiel*: Addition von 4 zu einer eingegeben Zahl mit einem Rechnermodell *Materialien*: Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.IV (Download LK-Q2-IV.1)
2. Simulation der Phasen eines Compilers ▪ Prozesse beim Compiler: o scannen o parsen o übersetzen/interpretieren ▪ Arten von Fehlern: o lexikalischer Fehler o syntaktischer Fehler o semantischer Fehler ▪ Einordnung der neuen Begriffe in den Gesamtkontext der formalen Sprachen o Automaten o Grammatiken o Sprachen	Beispiel: Scanner, Parser und Interpreter einer Konstruktionssprache für geometrische Figuren Anhand einer einführenden Folienpräsentation werden die Begrifflichkeiten definiert. Ein kleiner Ausschnitt einer Pseudo-Programmiersprache zur Konstruktion von Zeichnungen wird betrachtet. Anfänglich besteht der Sprachumfang aus Programmen mit lediglich einer einzigen Zuweisung. Die grundlegenden Schritte eines Compilers werden am Beispiel dieser Grammatik nachvollzogen. Materialien: Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.IV (Download LK-Q2-IV.2)
3. Die Schritte eines Compilers ▪ Scanner: o endlicher Automat als Grundlage o Vorgabe von Symboltabelle und Tokenliste zur Verwaltung und Erkennung des Quelltextes o Erweiterung des terminalen Alphabets der zu übersetzenden formalen Sprache o Implementierung als endlicher Automat ▪ Parser: o reguläre (oder wahlweise kontextfreie) Grammatik als Grundlage o Vorgabe einer Grundversion des Parsers o Erweiterung des Sprachumfangs o Implementierung der Parsermethoden für die Produktionsregeln der kontextfreien Grammatik	*Beispiel*: Der Scanner-Automat zur Erkennung der einzelnen Symbole wird als endlicher Automat realisiert. Mithilfe der Symboltabelle wird die Vereinfachung des Automaten deutlich gemacht und der vereinfachte Scanner-Automat wird schrittweise erweitert und implementiert. In der zweiten Phase wird die der Pseudoprogrammiersprache zugrunde liegende Grammatik analysiert. Die Überprüfung der syntaktischen Korrektheit wird in Form eines Parsers modelliert und implementiert. Dabei wird der Sprachumfang der Pseudo-Programmiersprache schrittweise erweitert. In der dritten Phase wird ein zu Teilen bereits vorbereiteter Interpreter analysiert und erweitert, welcher Programme der Pseudo-Programmiersprache zur Konstruktion von Zeichnungen in eine Grafik übersetzt. *Materialien*:

▪ Interpreter o Vorgabe einer Grundversion des Interpreters o Erweiterung des Sprachumfangs o Implementierung	Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.IV (Download LK-Q2-IV.3)
4. Grenzen der Automatisierbarkeit ▪ Vorstellung des Halteproblems ▪ Unlösbarkeit des Halteproblems ▪ Beurteilung des Einsatzes von Informatiksystemen hinsichtlich prinzipieller Möglichkeiten und prinzipieller Grenzen	*Beispiel:* Halteproblem *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.3 - Halteproblem (Download Q2-III.2)

UV Q2-IV: Entwicklung eines Netzwerkspiels mit Durchführung eines vollständigen Softwareentwicklungszyklus

Leitfragen:
- Wie ist ein Softwareentwicklungszyklus aufgebaut?
- Welches Protokoll ist für die vorgegebene Funktionalität angemessen bzw. garantiert eine fehlerfreie Kommunikation?
- Welche Reaktionen auf Kommunikationsereignisse sind server- und clientseitig zu entwickeln?
- Wie können verwendete Daten mit Hilfe von Graphen verwaltet werden?

Vorhabenbezogene Konkretisierungen:
Anwendungskontext ist ein von den Schüler:innen zu entwickelndes Zweipersonen-Netzwerkspiel. Anhand des Spiels sollen die grundlegenden Prinzipien und Begrifflichkeiten der Kommunikation in Netzwerken, dem Aufbau einer Client-Server-Struktur und der Datenorganisation mit Hilfe von Graphen aus den Unterrichtsvorhaben Q1-V und Q2-II festigend wiederholt werden. Grundlage für das Softwareprojekt sind die Klassen zur Netzwerkkommunikation Connection, Client und Server. Der Begriff des Softwareentwicklungszyklus wird thematisiert und es werden alle Phasen der Entwicklung eines Softwareprojekts über Analyse, Modellierung und Implementierung anhand des Netzwerkspiels vollzogen. Das Projekt läuft in folgenden groben Phasen ab:

1. Projekteinstieg / -planung (Spielauswahl, Zeitmanagement)
2. Projektumsetzung (informatische Analyse, Modellierung, Implementierung, Test ➔ ggf. in Zyklen)
   2.1 Analyse des Spiels mit dem Ziel einer späteren informatischen Umsetzung als Netzwerkspiel
   2.2 Modellierung und Implementierung des Spiels als Netzwerkanwendung
   2.3 Reflexion des Softwareprodukts
3. Projektreflexion (Reflexion der Projektplanung, Präsentation)

Zu entwickelnde Kompetenzen: Die Schüler:innen
- ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M),
- modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (M),
- modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M),
- verwenden bei der Modellierung geeigneter Problemstellungen Möglichkeiten der Polymorphie (M),
- ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M),
- stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D),
- analysieren und erläutern objektorientierte Modellierungen (A),
- implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I),
- analysieren und erläutern Algorithmen und Programme (A),

▪ modifizieren Algorithmen und Programme (I), ▪ entwickeln iterative und rekursive Algorithmen unter Nutzung der Strategien „Modularisierung“ und „Teilen und Herrschen“ und „Backtracking“ (M), ▪ implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), ▪ testen Programme systematisch anhand von Beispielen und mit Hilfe von Testanwendungen (I). ▪ analysieren und erläutern Algorithmen und Methoden zur Client-Server-Kommunikation (A), ▪ entwickeln und implementieren Algorithmen und Methoden zur Client-Server-Kommunikation (I). ▪ beschreiben und erläutern Netzwerk-Topologien, die Client-Server-Struktur und Protokolle sowie ein Schichtenmodell in Netzwerken (A), ▪ analysieren und erläutern Protokolle zur Kommunikation in einem Client-Server-Netzwerk (A), ▪ entwickeln und erweitern Protokolle zur Kommunikation in einem Client-Server-Netzwerk (M).	
Unterrichtssequenzen	*Beispiele, Medien, Materialien*

1. Projekteinstieg / -planung (Spielauswahl, Zeitmanagement) **2. Projektumsetzung (informatische Analyse, Modellierung, Implementierung, Test ➔ ggf. in Zyklen)** *2.1. Analyse des Spiels mit dem Ziel einer späteren informatischen Umsetzung als Netzwerkspiel* (a) Spielen des Spiels in mehreren Kleingruppen in Form eines Rollenspiels nach vereinbarten Regeln: i. Ein Schüler oder eine Schülerin übernehmen die Rolle des Spielleiters oder der Spielleiterin. ii. Ein weiteres Gruppenmitglied protokolliert präzise die Kommunikation zwischen der Spielleiterin bzw. dem Spielleiter und den Spielerinnen und Spielern. (b) Formalisierung des Spielablaufs *2.2. Modellierung und Implementierung des Spiels als Netzwerkanwendung* (a) Entwurf eines Protokolls zur Kommunikation zwischen Spielserver und -client basierend auf den erarbeiteten Spielregeln (b) Entwicklung von Entwurfs- und Implementationsdiagrammen für den Spielserver (c) Implementation und Test der Spielserver-Klasse und von dieser benötigter Fachklassen (d) Modellierung, Implementierung und Test des Spielclients (e) Test der Zusammenarbeit von Spielserver und Spielclient *2.3. Reflexion des Softwareprodukts* (a) Identifikation der Stufen eines Softwareentwicklungszyklus (b) Wiederholende Darstellung der Entwicklungsschritte zum fertigen Produkt (c) Mögliche Erweiterungen: i. ggf. Implementierung einer grafischen Benutzeroberfläche (GUI) für den Client ii. ggf. Analyse, Modellierung und Implementierung alternativer Spielregeln iii. ggf. Implementierung einer KI für einen Computer-Gegenspieler **3. Projektreflexion (Reflexion der Projektplanung, Präsentation)**	*Beispiel*: GraphColoringGame Zwei-Personen-Spiel, bei dem die beiden Personen abwechselnd am Zug sind. Das Spielfeld ist ein ungerichteter Graph ohne Mehrfachkanten. Die Knoten des Graphen können mit vorgegebenen Farben markiert werden. Die beiden Spielerinnen bzw. Spieler färben abwechselnd einen noch ungefärbten Knoten. Dabei muss beachtet werden, dass kein adjazenter Knoten mit derselben Farbe markiert wurde. Es verliert derjenige Spielerinn bzw. derjenige Spieler, die bzw. der unter diesen Bedingungen keinen Zug mehr machen kann. Sind alle Knoten korrekt gefärbt, endet das Spiel unentschieden. *Materialien*: Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2-V (LK) (Download LK-Q2-V.1)

2.2 Grundsätze der fachmethodischen und fachdidaktischen Arbeiten

Die Fachkonferenz Informatik hat sich unter Berücksichtigung des Schulprogramms auf folgende Grundsätze fachmethodischer und fachdidaktischer Arbeit geeinigt:

(1) Der Unterricht orientiert sich im aktuellen Stand der Informatik.

(2) Der Unterricht folgt dem Prinzip der Exemplarität und soll ermöglichen, informatische Strukturen und Gesetzmäßigkeiten in den ausgewählten Problemen und Projekten zu erkennen.

(3) Der Unterricht ist problemorientiert und knüpft an Interessen und Erfahrungen der Schüler:innen an.

(4) Der Unterricht ist anschaulich sowie gegenwarts- und zukunftsorientiert. Dazu beschäftigen sich die Schüler:innen auch mit aktuellen Informatiksystemen und deren weiterer Entwicklung.

(5) Der Unterricht ist handlungsorientiert, d. h. projekt- und produktorientiert angelegt, und fördert und fordert das selbstständige und eigenverantwortliche Arbeiten der Schüler:innen.

(6) Der Unterricht ist kooperativ, d. h. er fördert das gemeinsame und gemeinschaftliche Arbeiten und Problemlösen.

(7) Der Unterricht betont und berücksichtigt die individuellen Lernwege der Schüler:innen. Der Unterricht fördert die Kinder individuell und differenziert, wo dies erforderlich ist, und regt Schüler:innen mit besonderen Begabungen an, diese weiterzuentwickeln.

(8) Der Unterricht fördert Gendergerechtigkeit und Diversität, indem er die persönlichen Interessen der Lernenden im Informatikunterricht spezifisch berücksichtigt und stereotype Vorstellungen der Informatik überwindet.

(9) Der Unterricht fördert eine offene und positive Kommunikationskultur, in der Schüler:innen ermutigt sind, auch fehlerhafte oder unvollständige Lösungen zur Diskussion zu stellen sowie wertschätzende und konstruktive Kritik zu geben und zu erhalten.

(10) Der Unterricht leistet einen wichtigen Beitrag zur Vorbereitung auf Ausbildung und Beruf und zeigt informatikaffine Berufsfelder auf.

2.3 Grundsätze der Leistungsbewertung und Leistungsrückmeldung

Auf der Grundlage von § 48 SchulG und §13 - §16 APO-GOSt hat die Fachkonferenz im Einklang mit dem entsprechenden schulbezogenen Konzept die nachfolgenden Grundsätze zur Leistungsbewertung und Leistungsrückmeldung beschlossen.

Die nachfolgenden Absprachen stellen die Minimalforderungen an das lerngruppenübergreifende gemeinsame Handeln der Fachgruppenmitglieder dar. Bezogen auf die einzelne Lerngruppe kommen ergänzend weitere der in den Folgeabschnitten genannten Instrumente der Leistungsüberprüfung zum Einsatz.

2.3.1 Verbindliche Absprachen im Beurteilungsbereich der „Schriftlichen Leistungen“

2.3.1.1 Grundsätzliches

Die Schriftlichen Arbeiten („Klausuren“) dienen der Überprüfung der Lernergebnisse nach einer Unterrichtssequenz. Sie geben darüber Aufschluss, inwieweit die Schüler:innen in der Lage sind, die Aufgaben mit den im Unterricht erworbenen Kompetenzen zu lösen. Klausuren sind deshalb grundsätzlich in den aktuellen Unterrichtszusammenhang zu integrieren. Gleichwohl können die Arbeiten nach entsprechender Wiederholung im Unterricht auch Aufgabenteile enthalten, die Kompetenzen aus weiter zurückliegenden Unterrichtsvorhaben oder übergreifende prozessbezogene Kompetenzen erfordern.

Rückschlüsse aus den Klausuren werden dabei auch als Grundlage für die weitere Unterrichtsplanung sowie als Diagnoseinstrument für die individuelle Förderung genutzt.

2.3.1.2 Verbindliche Absprachen

Die Fachkonferenz Informatik hat folgende verbindliche Absprachen bezüglich der Schriftlichen Arbeiten getroffen:

(1) Die Anzahl der Klausuren im Fach Informatik der Sekundarstufe II ist im Rahmen der Vorgaben der APO-GOSt für den Wahlpflichtbereich wie folgt festgelegt:

Jahrgangsstufe	Klausuren pro Halbjahr	Dauer
EF	1	90 Min.
Q1	2	135 Min. (GK) bzw. 180 Min. (LK)
Q2.1	2	135 Min. (GK) bzw. 180 Min. (LK)
Q2.2	1 (sog. Abiturvorklausur; nur falls Abiturfach)	wie im Abitur

Anmerkung: Ab dem Abiturjahrgang 2024 liegt die Klausurzeit für die Abiturklausur bei 225 Minuten im Grundkurs und 270 Minuten im Leistungskurs.

(2) Anstelle einer Klausur kann in der Q1.2 eine Facharbeit geschrieben werden.

(3) Die Kompetenzbereiche (Darstellen und Interpretieren, Modellieren, Argumentieren, Implementieren, Kommunizieren und Kooperieren) werden in den Klausuren in angemessenem Umfang eingefordert.

(4) Die Formulierungen der Aufgabenstellungen orientieren sich an der vom Schulministerium des Landes Nordrhein-Westfalen herausgegebenen Operatorenübersicht für das Fach Informatik.

(5) Alle drei Anforderungsbereiche (AFB I: Reproduzieren, AFB II: Zusammenhänge herstellen, AFB III: Verallgemeinern und Reflektieren) sind bei der Erstellung der Kursarbeiten angemessen zu berücksichtigen, wobei der Anforderungsbereich II den Schwerpunkt bildet.

(6) Die Korrektur und Bewertung der Klausuren erfolgt transparent und kriteriengeleitet. Die Schüler:innen erhalten eine individualisierte, an Kompetenzen orientierte Rückmeldung, die auch als diagnostische Grundlage in Beratungsgesprächen und zur individuellen Förderung dient. Teillösungen und Ansätze sind bei der Bewertung angemessen zu berücksichtigen.

(7) Die Zuordnung der Rohpunktsumme zu den Notenstufen orientiert sich am Zuordnungsschema des Zentralabiturs (s. u.). Von diesen kann aber im Einzelfall begründet abgewichen werden, wenn sich z. B. besonders originelle Teillösungen nicht durch Rohpunkte gemäß den Kriterien des Erwartungshorizonts abbilden lassen oder eine Abwertung wegen besonders schwacher Darstellung (gem. APO-GOSt §13 (2)) angemessen erscheint.

Notenraster

Notenpunkt	ab % Rohpunkte	Notenpunkt	ab % Rohpunkte	Notenpunkt	ab % Rohpunkte
15	95	10	70	5	45
14	90	9	65	4	40
13	85	8	60	3	33
12	80	7	55	2	27
11	75	6	50	1	20

2.3.2 Verbindliche Absprachen im Beurteilungsbereich „Sonstige Leistungen im Unterricht“

2.3.2.1 Kriterien zur Bewertung der Sonstigen Mitarbeit

In die Bewertung der Sonstigen Mitarbeit fließen i. d. R. folgende Aspekte ein, die den Schüler:innen am Anfang des Schuljahres bekannt gegeben werden müssen:

- Beteiligung am Unterrichtsgespräch (in Quantität und Kontinuität)
- Methodische und inhaltliche Qualität der Beiträge
- Eingehen auf Beiträge und Argumentationen von Mitschülerinnen und Mitschülern, Unterstützung von Mitlernenden
- Umgang mit neuen Problemen und Beteiligung bei der Suche nach neuen Lösungswegen
- Selbstständigkeit im Umgang mit der Arbeit
- Gewissenhafte und vollständige Bearbeitung der Arbeitsaufträge (sowohl im Unterricht als auch bei der Erstellung der Hausaufgaben)
- Beteiligung während kooperativer Arbeitsphasen (Partner- oder Gruppenarbeit)
- Darstellungsleistung bei Kurzvorträgen und Referaten sowohl mündlich als auch bezüglich der in den Vorträgen genutzten Medien (Plakate, Folien, digitale Präsentationen etc.)
- Ergebnisse schriftlicher Übungen
- Eine dem Lernstand angemessene Verwendung der informatischen Fachsprache
- Korrekte Verwendung informatikspezifischer Darstellungsformen
- Anfertigung zusätzlicher (freiwilliger) Arbeiten, z. B. eigenständige Ausarbeitungen im Rahmen binnendifferenzierender Maßnahmen

Die Leistungsbewertung bezieht sich grundsätzlich auf die Erreichung der im schulinternen Curriculum festgelegten Kompetenzen (kriterienorientierte Bezugsnorm). Die Leistungsbewertung bezieht sich weiterhin in gewissem Rahmen auch auf die im Kurs erbrachten Leistungen der Lernenden (soziale Bezugsnorm). Die Tatsache, dass erfolgreiches Lernen kumulativ ist, wird im Beurteilungsbereich „Sonstige Leistungen“ bei der Leistungsbewertung angemessen berücksichtigt (individuelle Bezugsnorm).

2.3.2.2 Übersicht zur kriteriengeleiteten Bewertung der Leistungen in der Sonstigen Mitarbeit

Als Hilfestellung, wie Qualität und Quantität der Unterrichtsbeiträge begründet und gewichtet in die Benotung eingehen können, kann folgende Übersicht verwendet werden[3]:

Note	**Beschreibung der Leistung**
sehr gut	Regelmäßige, aktive Mitarbeit; produktiv, gesprächsfördernd und -lenkend; an Beiträge der MitschülerInnen sinnvoll anknüpfend; sachlich konzentriert in der Bearbeitung gestellter Aufgaben; störungsfreie Arbeit; eigenständige, den Unterricht tragende neue Gedanken, ggf. alternative Lösungswege; sprachlich präzise und nuanciert, durchgängig reflektierende und argumentative Beiträge; kann sich mühelos an jedem Gespräch beteiligen; fachsprachlich korrekte Diktion; verfügt über ein gutes Repertoire an idiomatischen Redemitteln
gut	Regelmäßige Mitarbeit; mehr eigenständige als reproduzierende Beiträge; sachlich konzentriert in der Bearbeitung gestellter Aufgaben, störungsfreie Arbeit; Impulse aufnehmend und gezielt verwertend; gelegentlich Fähigkeit zu Transferleistungen; manchmal Beiträge der MitschülerInnen aufgreifend; teilweise selbstständiges Urteilen; unterscheidet zwischen Wesentlichem und Unwesentlichem; sprachlich präzise und argumentativ formulierte Beiträge; flüssige und spontane Äußerungen, ohne offensichtliche Suche nach Wörtern; sachgerechte Formulierung von Ideen und Inhalten (treffender Wortschatz)
befriedigend	Häufigere, aber keine durchgängige Mitarbeit; meist rezeptiv, gelegentlich reproduktiv; auf Lenkung angewiesen, diese aber aufnehmend, selten Fähigkeiten zu Transferleistungen; auf Fragen Antworten gebend, die Einsicht in Zusammenhänge erkennen lassen; in mehreren Sätzen und in Zusammenhängen geläufig bis flüssig formulierte Beiträge; gelegentliche Suche nach treffenden Wörtern im Sachgebiet
ausreichend	Punktuelle freiwillige Mitarbeit mit geringem inhaltlichem Ertrag; weitgehend reproduktive Beiträge (Sachinformation, Unterrichtsergebnisse, Hausaufgaben); eher passive Aufmerksamkeit: bei Nachfrage nachvollziehendes Mitdenken erkennbar; in der sprachlichen Form wenig entfaltet; verfügt über einen geringen aktiven Fachwortschatz, kann aber rezeptiv dem Unterrichtsgespräch/der Diskussion folgenden
mangelhaft	Auf Nachfrage allenfalls akustische Aufnahme des Unterrichtsgespräches erkennbar; selten einzelne Äußerungen, aber ohne Ertrag; schweigendes Mitdenken? Fehlende Konzentration auf das Unterrichtsgeschehen; sprachlich unzureichend; Ein-Satz-Antworten ohne weitere Entfaltung; Schwierigkeiten, den Themenwortschatz sachgerecht anzuwenden und nachzuvollziehen und somit einer Diskussion zu folgen
ungenügend	Teilnahmslos, schweigend; auf Nachfrage kein verwertbarer Beitrag

2.3.3 Grundsätze zur Leistungsrückmeldung und Beratung

Zum Ende jedes Quartals erhalten die Schüler:innen eine Information in vorher vereinbarter Form über den individuellen Leistungsstand. Gegebenenfalls ist eine Kontaktaufnahme mit den Eltern erforderlich.

3 Zusammengestellt von B. Freyer, Hamm. Verändert (T. Fleger).

Grundsätzlich besteht die Möglichkeit zur Lernberatung an Elternsprechtagen sowie in den Sprechstunden der Fachlehrerinnen und Fachlehrer. Bei nicht ausreichenden Leistungen bietet die Lehrkraft dem Schüler bzw. der Schülerin (sowie den Erziehungsberechtigten) spezielle Beratungstermine an. Zentrale Inhalte der Beratungsgespräche werden dokumentiert. Zudem werden die Lernhinweise und die Unterstützungsangebote der Lehrkraft schriftlich festgehalten.

2.4 Lehr- und Lernmittel

In der Einführungsphase wird mit eigens von der Fachschaft Informatik entwickeltem Lehr- und Lernmaterial gearbeitet. In der Qualifikationsphase wird das Lehrwerk „Informatik – Lehrwerk für die gymnasiale Oberstufe“, Schülerband 2, des Schöningh-Verlags verwendet.

3 Entscheidungen zu fach- und unterrichtsübergreifenden Fragestellungen

3.1.1.1 Hausaufgabenkonzept

Die Informatiklehrkräfte beachten bei der Vergabe von Hausaufgaben das Hausaufgabenkonzept (Beschluss vom 4.10.2016) für die Sek. II des Gymnasium Paulinum.

Aufgrund der besonderen Natur des Informatikunterrichts werden Aufgaben häufig unter Verwendung von Informatiksystemen, z. B. Computern, die mit spezieller Software ausgestattet sind, bearbeitet. Aus diesem Grund ist von den Lehrkräften vor Erteilung einer Hausaufgabe sicherzustellen, dass allen Schüler:innen die technischen Möglichkeiten zur Verfügung stehen, die gegebene Aufgabe zu erledigen. Gegebenenfalls sind die Hausaufgaben von der Lehrkraft in solcher Form zu stellen, die die Schüler:innen sie ersatzweise passabel mit Stift und Papier bearbeiten können.

3.1.1.2 Zusammenarbeit mit anderen Fächern

Das Schulprogramm des Gymnasium Paulinum formuliert als Entwicklungsziel eine enge Zusammenarbeit der mathematisch-naturwissenschaftlichen Fächer.

In den Bereichen der theoretischen Informatik und Programmierung bietet sich vielfach eine Kooperation mit der Fachschaft Mathematik an, etwa im Bereich der Aussagenlogik oder bei der Umsetzung zahlreicher mathematischer Funktionen und Algorithmen in eine automatisierte, computerlesbare Form. In der technischen Informatik finden sich zahlreiche Anknüpfungspunkte mit der Physik, beginnend bei der Verwendung von Halbleitern sowie UND- und ODER-Schaltungen zur Implementierung zunehmend komplexerer logischer Schaltkreise, Halb- und Volladdierer.

Da im Inhaltsfeld „Informatik, Mensch und Gesellschaft“ auch gesellschaftliche und ethische Fragen im Unterricht angesprochen werden, soll eine mögliche Zusammenarbeit mit den gesellschaftswissenschaftlichen Fachgruppen, insbesondere der Philosophie ausgelotet werden.

Hinweise auf die verschiedenen Möglichkeiten der Zusammenarbeit finden sich bei den jeweiligen Unterrichtsvorhaben.

3.1.1.3 Wettbewerbe

Die Fachgruppe Informatik fördert und unterstützt Schüler:innen, die an Wettbewerben teilnehmen möchten. In der Sekundarstufe II I wird insbesondere eine regelmäßige Teilnahme am bundesweiten Wettbewerb „Informatik-Biber“ und dem Jugendwettbewerb Informatik sowie nach individueller Eignung und entsprechendem Interesse die Teilnahme einzelner Schüler:innen am Bundeswettbewerb Informatik angestrebt.

3.1.1.4 Fernunterricht und Distanzlernen

1. Curriculare und methodische Absprachen bei Fernunterricht

In der Jgst. EF ist eine Arbeit am Computer unumgänglich, um Praxis in der Programmierung mit Java zu erlangen und damit den Anforderungen des Kernlehrplans und des schulischen Curriculums zu genügen. Den Schüler:innen werden Hilfestellungen bei der Installation und Konfiguration der benötigten Software zur Verfügung gestellt.

Sofern eine Installation der Java Laufzeitumgebung und Entwicklungsumgebung nicht möglich ist, ist ggf. ein Ausweichen auf einen Online Java Compiler möglich. Eine Ausstattung der Schüler:innen mit Laptops ist vonseiten der Schule nur in geringem Umfang möglich.

In der Qualifikationsphase ist die Programmierung ebenfalls essenzieller Bestandteil der Arbeit im Informatikunterricht, darüber hinaus gestatten die curricular festgelegten Inhalte in größerem Maße die analoge Arbeit. Diese Inhalte können im Fernunterricht stärker in den Fokus genommen werden.

2. Curriculare und methodische Absprachen bei Teilpräsenzunterricht

Bei einem Präsenzunterricht mit einzelnen Absenzen von Schüler:innen kann eine Übertragung Videoübertragung des Unterrichtsgeschehens - spezifisch des Lehrers oder der Lehrerin erfolgen. Arbeitsblätter werden im PDF-Format zur Verfügung gestellt, Arbeitsaufträge mündlich oder bei Bedarf zusätzlich schriftlich gegeben. Die Plattform Microsoft Teams kann zur Bild- und Tonübertragung sowie zum Austausch von Aufgaben und Lösungen genutzt werden.

Bei einer Teilung der Lerngruppe (mit A- und B-Gruppen) wird mit der Methode *Flipped Classroom* gearbeitet: die Lehrerin oder der Lehrer stellt den Schüler:innen zur Heimarbeit Material zur Aneignung der theoretischen fachlichen Grundlagen eines Themas zur Verfügung. In den Präsenzsitzungen wird die Zeit zur effektiven Übung und Besprechung genutzt.

3. Aufarbeitung etwaiger Lerndefizite nach Phasen des Fernunterrichts

Nach Wiederbeginn des Präsenzunterrichts wird eine zeitlich begrenzte Wiederholungsphase angesetzt, in der die im Fernunterricht erworbenen Kompetenzen noch einmal überprüft und von den Schüler:innen demonstriert werden können.

4. Bewertung von Leistungen im Fernunterricht

Im Teilpräsenzunterricht werden nur die Präsenzphasen zur Bildung der Note zur sonstigen Mitarbeit herangezogen. Im Fernunterricht erhalten die Schüler:innen ein regelmäßiges Feedback bezüglich der von ihnen eingereichten Arbeitsresultate. Eine Notenbildung auf Basis dieser Resultate entfällt, sofern nicht vom Schulministerium andere Maßgaben getroffen werden.

4 Qualitätssicherung und Evaluation

Durch Diskussion der Aufgabenstellung schriftlicher Leistungsüberprüfungen und einen häufigen Austausch über die unterrichtlichen Erfahrungen mit den entwickelten und verwendeten Unterrichtsmaterialien in Fachdienstbesprechungen sowie eine regelmäßige Erörterung der Ergebnisse von Leistungsüberprüfungen wird ein hohes Maß an fachlicher Qualitätssicherung erreicht.

Das schulinterne Curriculum wird als Ergebnis dieser Fachgruppendiskussionen weiterentwickelt und neuen Erfordernissen bezüglich der Kompetenzorientierung und der aktuellen Entwicklung der Fachwissenschaft sowie der gesellschaftlich genutzten Informatiksysteme angepasst.

Insbesondere angesichts der Tatsache, dass das Fach Informatik am Gymnasium Paulinum eine junge Geschichte hat (der erste Informatikkurs in der Oberstufe startete im Schuljahr 2020/2021) und sich somit derzeit noch im Aufbau befindet, ist es der Fachgruppe Informatik ein wichtiges Anliegen, die einzelnen Unterrichtsvorhaben auch in Zukunft regelmäßig und umfangreich zu reflektieren und die konkrete Ausgestaltung der Vorhaben (vgl. Kapitel 2.1.2) bei Bedarf anzupassen und zu überarbeiten.

Von der Fachgruppe Informatik erkannte Fortbildungsnotwendigkeiten werden der Fortbildungskoordinatorin oder dem Fortbildungskoordinator benannt und eine Umsetzung beantragt. Weitergehende, insbesondere fachliche, fachdidaktische oder methodische Fortbildungen werden bedarfsgerecht von den Lehrkräften wahrgenommen und die Inhalte der Fortbildungen der Fachgruppe vorgestellt und gemeinsam zur Unterrichtsentwicklung genutzt.