Schulinterner Lehrplan
zum Kernlehrplan für die gymnasiale Oberstufe

des Johannes-Kepler-Gymnasiums Ibbenbüren

Informatik

(Stand: 08.03.2018)

Inhalt

1 Die Fachgruppe Informatik des Johannes-Kepler-Gymnasiums Ibbenbüren

Das Johannes-Kepler-Gymnasium (im Folgenden mit JKG abgekürzt) bildet zusammen mit der vor kurzem gegründeten Gesamtschule Ibbenbüren (vormals Hauptschule am Aasee) das „Schulzentrum Ost“. Zur Zeit besuchen ca. 1300 Schülerinnen und Schüler das Gymnasium, die von etwa 100 Lehrkräften unterrichtet werden. In der Sekundarstufe I gibt es in der Regel pro Jahrgang 5 Klassen. Das Einzugsgebiet der Schule umfasst die Stadt Ibbenbüren mit all ihren Vororten, aber auch aus weiter entfernt liegenden Nachbargemeinden kommen immer wieder Schülerinnen und Schülern ans JKG. Seit Jahren besonders großen Zulauf erhalt das JKG in der Oberstufe; in der Einführungsphase kann das JKG meistens drei, mindestens aber zwei eigene Klassen nur mit ehemaligen Haupt- und Realschülerinnen und –schülern einrichten, welche dann während des gesamtes Schuljahres in den Hauptfächern eine besondere Förderung erhalten. Als einen weiteren Grund für den großen Zuspruch in der Oberstufe kann das gute Fächerangebot auch und gerade im Bereich der Leistungskurse angesehen werden: aufgrund der zur Zeit sehr großen Schüleranzahl in den Stufen bleiben kaum Fächer-Wünsche der Schülerinnen und Schüler offen. In Kooperation mit dem Goethe-Gymnasium konnte in den letzten 15 Jahren fast ausnahmslos jährlich ein Informatik-Leistungskurs eingerichtet werden, welcher stets von einer Lehrkraft des JKGs geleitet wurde. Aber auch in den Grundkursen der Oberstufe haben die Lehrkräfte im Blick, dass es Schülerinnen und Schülern gibt, die in der Sekundarstufe I keinen Informatikunterricht besucht haben: es wird speziell in den Informatik-Kursen der Einführungsphase besonderer Wert darauf gelegt, dass keine Vorkenntnisse aus der Sekundarstufe I zum erfolgreichen Durchlaufen des Kurses erforderlich sind.

Das Fach Informatik wird am JKG ab der Jahrgangsstufe 8 im Wahlpflichtbereich II (WP II) unterrichtet; meistens können/müssen pro Jahrgang gleich zwei Informatik-Kurse eingerichtet werden. In der zweijährigen Laufzeit dieser Kurse wird in altersstufengerechter Weise unter anderem auf Grundlagen der Algorithmik am Beispiel der didaktischen Lernumgebung „mit Stiften und Mäusen (mSuM)“ in der objektorientierten Programmiersprache Delphi eingegangen. Hiermit lässt sich auch bereits in diesen Jahrgangsstufen das Thema Rekursion grafisch visualieren. Ferner werden die Schülerinnen und Schüler in weitere Themengebiete der Informatik eingeführt, die dann in ausgewählten Teilen auch „mSuM“ implementiert werden.

Der Unterricht der Sekundarstufe II wird mit Hilfe der Programmiersprache Java durchgeführt. In der Einführungsphase kommt dabei zusätzlich eine didaktische Bibliothek zum Einsatz, welche das Erstellen von grafischen Programmen erleichtert.

Durch projektartiges Vorgehen, offene Aufgaben und Möglichkeiten, Problemlösungen zu verfeinern oder zu optimieren, entspricht der Informatikun-

terricht der Oberstufe in besonderem Maße den Erziehungszielen, Leistungsbereitschaft zu fördern, ohne zu überfordern.

Die gemeinsame Entwicklung von Materialien und Unterrichtsvorhaben, die Evaluation von Lehr- und Lernprozessen sowie die stetige Überprüfung und eventuelle Modifikation des schulinternen Curriculums durch die Fachkonferenz Informatik stellen einen wichtigen Beitrag zur Qualitätssicherung und -entwicklung des Unterrichts dar.

Zurzeit besteht die Fachschaft Informatik des JKGs aus vier Lehrkräften, denen drei Computerräume mit jeweils 16 Computerarbeitsplätzen, ein Pool-Raum mit 12 Plätzen sowie Selbstlernzentrum mit 10 Plätzen zur Verfügung stehen. Alle Arbeitsplätze sind an das schulinterne Rechnernetz angeschlossen, so dass Schülerinnen und Schüler über einen individuell gestaltbaren Zugang zum zentralen Server der Schule alle Arbeitsplätze der genannten Bereiche zum Zugriff auf ihre eigenen Daten, zur Recherche im Internet oder zur Bearbeitung schulischer Aufgaben verwenden können.

Der Unterricht erfolgt im 60-Minuten-Takt. Grundkurse werden stets in Einzelstunden unterrichtet, der Leistungskurs hat daneben pro Woche eine Doppelstunde.

2 Entscheidungen zum Unterricht

2.1 Unterrichtsvorhaben

Die Darstellung der Unterrichtsvorhaben im schulinternen Lehrplan besitzt den Anspruch, sämtliche im Kernlehrplan angeführten Kompetenzen abzudecken. Dies entspricht der Verpflichtung jeder Lehrkraft, Schülerinnen und Schülern Lerngelegenheiten zu ermöglichen, so dass alle Kompetenzerwartungen des Kernlehrplans von ihnen erfüllt werden können.
Die entsprechende Umsetzung erfolgt auf zwei Ebenen: der Übersichts- und der Konkretisierungsebene.

Im „Übersichtsraster Unterrichtsvorhaben“ (Kapitel 2.1.1) wird die für alle Lehrerinnen und Lehrer gemäß Fachkonferenzbeschluss verbindliche Verteilung der Unterrichtsvorhaben dargestellt. Das Übersichtsraster dient dazu, den Kolleginnen und Kollegen einen schnellen Überblick über die Zuordnung der Unterrichtsvorhaben zu den einzelnen Jahrgangsstufen sowie den im Kernlehrplan genannten Kompetenzen, Inhaltsfeldern und inhaltlichen Schwerpunkten zu verschaffen. Der ausgewiesene Zeitbedarf versteht sich als grobe Orientierungsgröße, die nach Bedarf über- oder unterschritten werden kann. Um Freiraum für Vertiefungen, besondere Schülerinteressen, aktuelle Themen bzw. die Erfordernisse anderer besonderer Ereignisse (z.B. Praktika, Kursfahrten o.ä.) zu erhalten, wurden im Rahmen dieses schulinternen Lehrplans ca. 75 Prozent der Bruttounterrichtszeit verplant.

Während der Fachkonferenzbeschluss zum „Übersichtsraster Unterrichtsvorhaben“ zur Gewährleistung vergleichbarer Standards sowie zur Absicherung von Lerngruppenübertritten und Lehrkraftwechseln für alle Mitglieder der Fachkonferenz Bindekraft entfalten soll, beinhaltet die Ausweisung „konkretisierter Unterrichtsvorhaben“ (Kapitel 2.1.2) Beispiele und Materialien, die empfehlenden Charakter haben. Referendarinnen und Referendaren sowie neuen Kolleginnen und Kollegen dienen diese vor allem zur standardbezogenen Orientierung in der neuen Schule, aber auch zur Verdeutlichung von unterrichtsbezogenen fachgruppeninternen Absprachen zu didaktisch-methodischen Zugängen, fächerübergreifenden Kooperationen, Lernmitteln und -orten sowie vorgesehenen Leistungsüberprüfungen, die im Einzelnen auch den Kapiteln 2.2 bis 2.3 zu entnehmen sind.

Da in den folgenden Unterrichtsvorhaben Inhalte in der Regel anhand von Problemstellungen in Anwendungskontexten bearbeitet werden, werden in einigen Unterrichtsvorhaben jeweils mehrere Inhaltsfelder angesprochen.

2.1.1 Übersichtsraster Unterrichtsvorhaben

2.1.1.1 Einführungsphase

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

Thema:
Grundlegende Begrifflichkeiten und Funktionsweise einer funktionalen Programmiersprache (Prolog)

Zentrale Kompetenzen:
• Modellieren
• Implementieren
• Kommunizieren und Kooperieren
• Darstellen und Interpretieren

Inhaltsfelder:
• Informatiksysteme
• Formale Sprachen

Inhaltliche Schwerpunkte:
• Dateisystem
• Syntax und Semantik einer Programmiersprache

Zeitbedarf: ca. 20 Stunden</td>
<td><u>Unterrichtsvorhaben E-II</u>

Thema:
Grundlagen der objektorientierten Analyse, Modellierung und Implementierung sowie Algorithmik anhand von Greenfoot-Projekten

Zentrale Kompetenzen:
• Modellieren
• Implementieren
• Darstellen und Interpretieren
• Kommunizieren und Kooperieren

Inhaltsfelder:
• Daten und ihre Strukturierung
• Algorithmen
• Formale Sprachen

Inhaltliche Schwerpunkte:
• Objekte und Klassen
• Syntax und Semantik einer Programmiersprache
• Analyse, Entwurf und Implementierung einfacher Algorithmen

Zeitbedarf: 20 Stunden</td>
</tr>
</table>

Einführungsphase	
Unterrichtsvorhaben E-III **Thema**: *Codierungen* **Zentrale Kompetenzen:** • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Informatiksysteme **Inhaltliche Schwerpunkte:** • Digitalisierung **Zeitbedarf**: 4 Stunden	*Unterrichtsvorhaben E-IV* **Thema**: *Modellierung und Implementierung von Klassen- und Objektbeziehungen anhand von Simulationen in BlueJ* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen **Inhaltliche Schwerpunkte:** • Objekte und Klassen • Syntax und Semantik einer Programmiersprache • Analyse, Entwurf und Implementierung einfacher Algorithmen **Zeitbedarf**: 18 Stunden

<table>
<tr><th colspan="2">Einführungsphase</th></tr>
<tr><td><u>Unterrichtsvorhaben E-V</u>

Thema:
Lineare Datensammlung sowie Such- und Sortieralgorithmen anhand kontextbezogener Beispiele

Zentrale Kompetenzen:
• Argumentieren
• Modellieren
• Darstellen und Interpretieren
• Kommunizieren und Kooperieren

Inhaltsfelder:
• Algorithmen

Inhaltliche Schwerpunkte:
• Statische lineare Datenstruktur zur effektiven Verwaltung großer Datenmengen
• Algorithmen zum Suchen und Sortieren
• Analyse, Entwurf und Implementierung einfacher Algorithmen

Zeitbedarf: 12 Stunden</td><td><u>Unterrichtsvorhaben E-VI</u>

Thema:
Geschichte der digitalen Datenverarbeitung und die Grundlagen des Datenschutzes

Zentrale Kompetenzen:
• Argumentieren
• Darstellen und Interpretieren
• Kommunizieren und Kooperieren

Inhaltsfelder:
• Informatik, Mensch und Gesellschaft
• Informatiksysteme

Inhaltliche Schwerpunkte:
• Wirkungen der Automatisierung
• Geschichte der automatischen Datenverarbeitung
• Einzelrechner

Zeitbedarf: 6 Stunden</td></tr>
<tr><td colspan="2">Summe Einführungsphase: 80</td></tr>
</table>

2.1.1.2 Qualifikationsphase 1 (Grundkurs und Leistungskurs)

<table>
<tr><th colspan="2">Qualifikationsphase 1</th></tr>
<tr>
<td><u>Unterrichtsvorhaben Q1-I</u>

Thema:
Wiederholung der objektorientierten Modellierung und Programmierung anhand einer kontextbezogenen Problemstellung

Zentrale Kompetenzen:
• Argumentieren
• Modellieren
• Implementieren
• Darstellen und Interpretieren
• Kommunizieren und Kooperieren

Inhaltsfelder:
• Daten und ihre Strukturierung
• Algorithmen
• Formale Sprachen und Automaten
• Informatiksysteme

Inhaltliche Schwerpunkte:
• Objekte und Klassen
• Analyse, Entwurf und Implementierung von Algorithmen
• Syntax und Semantik einer Programmiersprache
• Nutzung von Informatiksystemen

Zeitbedarf: 8 Stunden</td>
<td><u>Unterrichtsvorhaben Q1-II</u>

Thema:
Modellierung und Implementierung von Anwendungen mit dynamischen, linearen Datenstrukturen

Zentrale Kompetenzen:
• Argumentieren
• Modellieren
• Implementieren
• Darstellen und Interpretieren
• Kommunizieren und Kooperieren

Inhaltsfelder:
• Daten und ihre Strukturierung
• Algorithmen
• Formale Sprachen und Automaten

Inhaltliche Schwerpunkte:
• Objekte und Klassen
• Analyse, Entwurf und Implementierung von Algorithmen
• Algorithmen in ausgewählten informatischen Kontexten
• Syntax und Semantik einer Programmiersprache

Zeitbedarf: 20 Stunden</td>
</tr>
</table>

Qualifikationsphase 1	
Unterrichtsvorhaben Q1-III **Thema:** *Suchen und Sortieren auf linearen Datenstrukturen* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Algorithmen • Formale Sprachen und Automaten **Inhaltliche Schwerpunkte:** • Analyse, Entwurf und Implementierung von Algorithmen • Algorithmen in ausgewählten informatischen Kontexten • Syntax und Semantik einer Programmiersprache **Zeitbedarf**: 16 Stunden	*Unterrichtsvorhaben Q1-IV* **Thema:** *Modellierung und Nutzung von relationalen Datenbanken in Anwendungskontexten* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen und Automaten • Informatik, Mensch und Gesellschaft **Inhaltliche Schwerpunkte:** • Datenbanken • Algorithmen in ausgewählten informatischen Kontexten • Syntax und Semantik einer Programmiersprache • Sicherheit **Zeitbedarf**: 20 Stunden

Qualifikationsphase 1	
Unterrichtsvorhaben Q1-V **Thema:** *Sicherheit und Datenschutz in Netzstrukturen* **Zentrale Kompetenzen:** • Argumentieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Informatiksysteme • Informatik, Mensch und Gesellschaft **Inhaltliche Schwerpunkte:** • Einzelrechner und Rechnernetzwerke • Sicherheit • Nutzung von Informatiksystemen, Wirkungen der Automatisierung **Zeitbedarf**: 10 Stunden	
Summe Qualifikationsphase 1: 74 Stunden	

2.1.1.3 Qualifikationsphase 2 (Grundkurs und Leistungskurs)

<table>
<tr><th colspan="2">Qualifikationsphase 2</th></tr>
<tr>
<td><u>Unterrichtsvorhaben Q2-I</u>

Thema:
Modellierung und Implementierung von Anwendungen mit dynamischen, nichtlinearen Datenstrukturen

Zentrale Kompetenzen:
• Argumentieren
• Modellieren
• Implementieren
• Darstellen und Interpretieren
• Kommunizieren und Kooperieren

Inhaltsfelder:
• Daten und ihre Strukturierung
• Algorithmen
• Formale Sprachen und Automaten

Inhaltliche Schwerpunkte:
• Objekte und Klassen
• Analyse, Entwurf und Implementierung von Algorithmen
• Algorithmen in ausgewählten informatischen Kontexten
• Syntax und Semantik einer Programmiersprache

Zeitbedarf: 24 Stunden</td>
<td><u>Unterrichtsvorhaben Q2-II</u>

Thema:
Endliche Automaten und formale Sprachen

Zentrale Kompetenzen:
• Argumentieren
• Modellieren
• Darstellen und Interpretieren
• Kommunizieren und Kooperieren

Inhaltsfelder:
• Endliche Automaten und formale Sprachen

Inhaltliche Schwerpunkte:
• Endliche Automaten
• Grammatiken regulärer Sprachen
• Möglichkeiten und Grenzen von Automaten und formalen Sprachen

Zeitbedarf: 20 Stunden</td>
</tr>
</table>

Qualifikationsphase 2	
Unterrichtsvorhaben Q2-III **Thema:** *Prinzipielle Arbeitsweise eines Computers und Grenzen der Automatisierbarkeit* **Zentrale Kompetenzen:** • Argumentieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Informatiksysteme • Informatik, Mensch und Gesellschaft **Inhaltliche Schwerpunkte:** • Einzelrechner und Rechnernetzwerke • Grenzen der Automatisierung **Zeitbedarf:** 12 Stunden	*Unterrichtsvorhaben Q2-IV (nur LK)* **Thema:** *Modellierung und Implementierung von Anwendungen mit dynamischen, nichtrekursiven Datenstrukturen (Graphen)* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen und Automaten **Inhaltliche Schwerpunkte:** • Objekte und Klassen • Analyse, Entwurf und Implementierung von Algorithmen • Algorithmen in ausgewählten informatischen Kontexten • Syntax und Semantik einer Programmiersprache **Zeitbedarf**: 18 Stunden

Qualifikationsphase 2	
Unterrichtsvorhaben Q2-V **Thema:** *Wiederholung und Vorbereitung Abitur* **Zeitbedarf:** 12 Stunden	
Summe Qualifikationsphase 2: 56 (LK: 74) Stunden	

2.1.2 Konkretisierte Unterrichtsvorhaben

Im Folgenden sollen die im *Unterkapitel 2.1.1* aufgeführten Unterrichtsvorhaben konkretisiert werden.

Hinweis:

Verbindliche Festlegungen der Fachkonferenz:
Die Fachkonferenz des Kepler-Gymnasiums hat Themen, Leitfragen und die Ausführungen unter der Überschrift *Vorhabenbezogene Konkretisierung* verbindlich vereinbart, ebenso die Sequenzierung der Unterrichtsvorhaben (erste Tabellenspalte) und die ausgewiesenen Kompetenzen (zweite Tabellenspalte). Alle Mitglieder der Fachkonferenz haben sich darauf verständigt, in ihrem Unterricht Lerngelegenheiten anzubieten, so dass Schülerinnen und Schüler diese Kompetenzen im Rahmen der festgelegten Unterrichtssequenzen erwerben oder vertiefen können.

Unterrichtliche Anregungen:
Die angeführten Beispiele, Medien und Materialien sind dagegen Vorschläge bzw. Hilfen für die Lehrkräfte des Kepler-Gymnasiums. In diesen Bereichen sind Abweichungen von den vorgeschlagenen Vorgehensweisen möglich.

Die folgenden Installationspakete und Dokumentationen stehen den Schülern im Unterricht, aber auch zum freien Download im Internet zur Verfügung:

- SWI-Prolog
- Greenfoot
- BlueJ
- Filius
- Dokumentationen der Programme

In der Qualifikationsphase werden die Unterrichtsvorhaben unter Berücksichtigung der Vorgaben für das Zentralabitur Informatik in NRW konkretisiert. Diese sind zu beziehen unter der Adresse

http://www.standardsicherung.schulministerium.nrw.de/abitur-gost/fach.php?fach=15 (abgerufen: 30. 04. 2014)

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 Schülerinnen und Schüler

- 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).

Unterrichtsvorhaben EF-I

Thema: Grundlegende Begrifflichkeiten und Funktionsweise einer funktionalen Programmiersprache (Prolog)

Vorhabenbezogene Konkretisierung:
Das erste Unterrichtsvorhaben stellt eine allgemeine Einführung in das Fach Informatik dar. Dabei ist zu berücksichtigen, dass für manche Schülerinnen und Schüler in der Einführungsphase der erste Kontakt mit dem Unterrichtsfach Informatik stattfindet, so dass zu Beginn eine grundlegende Einführung in das Schulnetzsystem des Kepler-Gymnasiums stattfinden muss.

Der Einstieg mit der funktionalen Programmiersprache Prolog gewährleistet einen Einstieg in die Programmierung, bei der die SuS keinerlei Vorkenntnisse mitbringen müssen. Dies ist sinnvoll, da das JKG einen hohen Anteil an ehemaligen Haupt- und Realschülern in der EP vorweist.

Bei der Beschäftigung mit wissensbasierter Modellierung lernen die SuS ein anderes Programmierparadigma als die weit verbreitete OOP-Sichtweise kennen. Sie beinhaltet neben den grundlegenden Konzepten Fakten, Anfragen, Regeln einen ersten Zugang zur Rekursion.

Zeitbedarf: ca. 20 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Informations- und Datenübermittlung in Netzen (a) Informatische Kommunikation in Rechnernetzen am Beispiel des Schulnetzwerks (z.B. Benutzeranmeldung, Netzwerkordner, Zugriffsrechte) (b) Möglichkeiten zur Datenspeicherung und zum Datenaustausch auf den Schulrechnern (z.B. Ordnerstruktur, Dateibezeichner usw.)	Die Schülerinnen und Schüler • nutzen die im Unterricht eingesetzten Informatiksysteme selbstständig, sicher, zielführend und verantwortungsbewusst (D), • nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung und gemeinsamen Verwendung von Daten unter Berücksichtigung der Rechteverwaltung (K).	
2. Grundlagen wissensbasierter Programmierung (a) Idee der wissensbasierten Modellierung/Programmierung (b) Wissensbasen erstellen: Fakten (c) Anfragen an Wissensbasen (d) Regeln für Wissensbasen formulieren	• konstruieren zu kontextbezogenen Problemstellungen informatische Modelle, modifizieren und erweitern diese (A+I), • implementieren einfache Algorithmen unter Beachtung der Syntax und Semantik einer Programmiersprache (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I).	
3. Rekursion (a) Idee der Rekursion an einfachen Beispielen (Fibonacci, Fakultät, Türme von Hanoi) (b) Formulierung rekursiver Regeln in Prolog, Rekursionsanker (c) Darstellungsformen rekursiver Abläu-		

fe (Aufrufbaum, Traversierung, Rekursionstiefe)		

Unterrichtsvorhaben EF-II

Thema: Grundlagen der objektorientierten Analyse, Modellierung und Implementierung sowie Algorithmik anhand von Greenfoot-Projekten

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

Dazu werden zunächst konkrete Gegenstandsbereiche aus der Lebenswelt der Schülerinnen und Schüler 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 und Modifizierung erster Projekte mit Hilfe der didaktischen Programmierumgebung Greenfoot begonnen. Die von der Bibliothek vorgegebenen Klassen werden von Schülerinnen und Schülern in Teilen analysiert und entsprechende Objekte anhand einfacher Problemstellungen erprobt. Dazu muss der grundlegende Aufbau einer Java-Klasse thematisiert und zwischen Attributen und Methoden sowie Deklaration, Initialisierung und Methodenaufrufen unterschieden werden.

Zu Beginn wird bewusst auf die Verwendung von Kontrollstrukturen verzichtet wird und der Quellcode aus rein linearen Sequenzen zusammengesetzt. So ist eine Fokussierung auf die Grundlagen der Objektorientierung möglich, ohne dass algorithmische Probleme ablenken.
Anschließend steht die Verwendung von Kontrollstrukturen (Verzweigung, Schleifen) zur Umsetzung und Implementierung einfacher Algorithmen im Mittelpunkt. Außerdem werden logische Verknüpfungen (boolsche Operatoren) behandelt.

Zeitbedarf: ca. 20 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Identifikation von Objekten (a) Am Beispiel eines lebensweltnahen Beispiels werden Objekte im Sinne der Objektorientierten Modellierung eingeführt. (b) Objekte werden mit Objektkarten visualisiert und mit sinnvollen Attributen und „Fähigkeiten“, d.h. Methoden versehen. (c) Unterscheidung von Klasse und Objekt, Fähigkeiten und Attributen (d) Vereinfachte Klassen- und Objektdiagramme (e) Einfache Modellierungen (Entwurf von Klassendiagrammen für bestimmte Anwendungsfälle)	Die Schülerinnen und Schüler • 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), • modellieren Klassen unter Verwendung von Vererbung (M), ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen (M) zu, stellen den Zustand eines Objekts dar (D), • stellen Klassen, Assoziations- und Vererbungsbeziehungen in Diagrammen grafisch dar (D), • analysieren und erläutern eine objektorientierte Modellierung (A), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I).	
2. Analyse und Modifikation von Klassen didaktischer Lernumgebungen (a) Objektorientierte Programmierung als modularisiertes Vorgehen (Entwicklung von Problemlösungen auf Grundlage vorhandener Klassen) (b) Teilanalyse der Klassen der didaktischen Lernumgebungen Greenfoot (c) Grundaufbau einer Java-Klasse (d) Implementierung eigener Methoden unter Verwendung von internen Me-		

thodenaufrufen auch mit Parameterübergabe		
3. Algorithmik (a) Einführung der Kontrollstrukturen: - bedingte Anweisung - Mehrfachverzweigung - vorprüfende Schleife - nachprüfende Schleife - Zählschleife anhand ausgewählter Problemstellungen in Greenfoot-Projekten (b) Anwendung der Kontrollstrukturen beim Entwurf und der Implementierung einfacher Algorithmen. (c) Dekonstruktion vorgegebener Algorithmen	Die Schülerinnen und Schüler • analysieren und erläutern einfache Algorithmen und Programme (A), • modifizieren einfache Algorithmen und Programme (I), • entwerfen einfache Algorithmen und stellen sie umgangssprachlich und grafisch dar (M), • implementieren Algorithmen unter Verwendung von Kontrollstrukturen sowie Methodenaufrufen (I), • testen Programme schrittweise anhand von Beispielen (I).	

Unterrichtsvorhaben EF-III

Thema: Codierung

Vorhabenbezogene Konkretisierung:
Zur Vorbereitung auf die folgenden Unterrichtssequenzen (insbesondere Daten und Datentypen) wird in diesem Unterrichtsvorhaben die digitale Repräsentation von Zahlen und Zeichen thematisiert. Insbesondere wird dabei auf das Binärsystem sowie den ASCII eingegangen.
Zeitbedarf: ca. 4 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
Binärsystem (a) Grundlagen: Bit als kleinste Informationseinheit zur Repräsentation zweier Zustände, Verkettung von Bits zur Repräsentation mehrerer Zustände. (b) Umwandlung zwischen Binär- und Dezimalsystem (c) Zusammenhang zwischen Java-Datentypen (byte, int, char, double, float) und der Länge ihrer Binärcodierung **Codierung** (a) Eindeutigkeit von Codierungen (b) ASCII-Code als Beispiel zur Zeichencodierung, Codierung und Decodierung von Text.	Die Schülerinnen und Schüler • stellen ganze Zahlen und Zeichen in Binärcodes dar (D), • interpretieren Binärcodes als Zahlen und Zeichen (D),	

Unterrichtsvorhaben EF-IV

Thema: Modellierung und Implementierung von Klassen- und Objektbeziehungen anhand von Simulationen in BlueJ

Vorhabenbezogene Konkretisierung:
Der Schwerpunkt dieses Unterrichtsvorhabens liegt auf der Entwicklung und Modifizierung von Simulationen aus dem Alltagsgebrauch der SuS. Zunächst wird ein Projekt bearbeitet, anhand dessen bereits erlernte Grundlagen (Attribute, Methoden, lokale Variablen, Ausgaben, Konstruktoren, Methodenaufrufe, Parameter) vertieft werden. Dabei werden auch noch einmal Kontrollstrukturen in Form von Schleifen und Verzweigungen benötigt und aufgegriffen.

Das zweite Projekt besteht aus mehreren Klassen, die miteinander interagieren. Dabei sollen Klassenbeziehungen, externe Methodenaufrufe und Objekte als Attribute thematisiert werden. In der Herangehensweise wird zusätzlich die informatische Idee der Abstraktion und Modularisierung behandelt.

Zeitbedarf: ca. 18 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Vertiefung/Wiederholung grundlegender objektorientierter Konzepte (a) Attribute und Methoden (b) Methodentypen (mit Parameter) (c) Ausgaben (d) lokale Variablen (e) Konstruktoren	Die Schülerinnen und Schüler • 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 oder Objekttypen 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).	*Beispiel*: Ticketautomat Die Schülerinnen und Schüler realisieren und modifizieren einen einfachen Ticketautomaten und erweitern diesen im Hinblick auf eine realitätsnahe Simulation *Beispiel*: Bibliothek
2. Objektinteraktion, Systeme mit mehreren Klassen (a) Assoziation (b) externe Methodenaufrufe (c) Abstraktion und Modularisierung (d) Objekte als Attribute		*Beispiel:* Zug Die Schülerinnen und Schüler realisieren einen graphischen Zug, der sich über den Bildschirm bewegen kann. Der Zug setzt sich zusammen aus elmentaren, geometrischen Flächen.

Unterrichtsvorhaben EF-V

Thema: Lineare Datensammlung sowie Such- und Sortieralgorithmen anhand kontextbezogener Beispiele

Vorhabenbezogene Konkretisierung:
Dieses Unterrichtsvorhaben beschäftigt sich zunächst mit der Einführung von Feldern als Möglichkeit der effektiven Verwaltung von vielen Objekten gleichen Typs. Daran schließt sich die Erarbeitung von Such- und Sortieralgorithmen an. Der Schwerpunkt des Vorhabens liegt dabei auf den Algorithmen selbst; nicht alle, sondern nur ausgewählte Verfahren sollen auch in Java implementiert werden.

Zunächst diskutieren die Schülerinnen und Schüler mögliche Einsatzszenarien für Such- und Sortieralgorithmen, um sich der Bedeutung einer effizienten Lösung dieser Probleme bewusst zu werden. In einem ersten Schritt erschließen sich die SuS das lineare und binäre Suchverfahren. Anschließend werden Strategien zur Sortierung mit Hilfe eines explorativen Spiels von den Schülerinnen und Schülern selbst erarbeitet. Nach der algorithmischen Formulierung der Verfahren folgt abschließend der Vergleich hinsichtlich Zeitaufwand und Effizienz.

Die Schülerinnen und Schüler sollen auf diese Weise das *Sortieren durch Vertauschen*, das *Sortieren durch Auswählen* und mindestens einen weiteren Sortieralgorithmus, kennen lernen. Die Implementation ein rekursiven Such- oder Sortierverfahrens (z.B. Quicksort) ist nicht vorgesehen.

Zeitbedarf: 12 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Einführung von Feldern (a) Notwendigkeit von linearen Datensammlungen (b) Grundlegende Struktur und implementierung eines Arrays in Java	Die Schülerinnen und Schüler • beurteilen die Effizienz von Algorithmen am Beispiel von Sortierverfahren hin-	

2. Lineare und binäre Suche (a) Suchaufgaben im Alltag und im Kontext informatischer Systeme (b) Evtl. Simulationsspiel zum effizienten Suchen mit binärer Suche (c) Effizienzbetrachtungen zur binären Suche im Vergleich zur linearen Suche	sichtlich Zeit (A), • entwerfen einen weiteren Algorithmus zum Sortieren (M), • analysieren Such- und Sortieralgorithmen und wenden sie auf Beispiele an (D).	
3. Explorative Erarbeitung von Sortierverfahren (a) Sortierprobleme im Kontext informatischer Systeme und im Alltag (z.B. Dateisortierung, Tabellenkalkulation, Telefonbuch, Bundesligatabelle, usw.) (b) Vergleich zweier Elemente als Grundlage eines Sortieralgorithmus (Dreieckstausch) (c) Erarbeitung eines Sortieralgorithmus durch die Schülerinnen und Schüler (d) Formulierung eines ausgewählten Verfahrens in Pseudocode und Implementierung in Java		
4. Systematisierung von Algorithmen und Effizienzbetrachtungen (a) Anwendung von Sortieralgorithmen auf verschiedene Beispiele (b) Bewertung von Algorithmen anhand der Anzahl der nötigen Vergleiche (c) Effizienzbetrachtungen an einem konkreten Beispiel bezüglich der Rechenzeit.		

Unterrichtsvorhaben EF-VI

Thema: Geschichte der digitalen Datenverarbeitung und die Grundlagen des Datenschutzes

Vorhabenbezogene Konkretisierung:
Das folgende Unterrichtsvorhaben stellt den Abschluss der Einführungsphase dar. Schülerinnen und Schüler sollen selbstständig in Kleingruppen informatische Themenbereiche aus dem Kontext der Geschichte der Datenverarbeitung bearbeiten und vorstellen.

Anschließend wird verstärkt auf den Aspekt des Datenschutzes eingegangen. Dazu kann ggf. ein Rollenspiel durchgeführt werden. Das Bundesdatenschutzgesetz wird 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.

Zeitbedarf: 6 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Selbstständige Erarbeitung von Themen durch die Schülerinnen und Schüler (a) Mögliche Themen zur Erarbeitung in Kleingruppen: • „Eine kleine Geschichte der Digitalisierung: vom Morsen zum modernen Digitalcomputer“ • „Eine kleine Geschichte der Kryptographie: von Caesar zur Enigma“ • „Auswirkungen der Digitalisierung:	Die Schülerinnen und Schüler • bewerten anhand von Fallbeispielen die Auswirkungen des Einsatzes von Informatiksystemen (A), • erläutern wesentliche Grundlagen der Geschichte der digitalen Datenverarbeitung (A), • nutzen das Internet zur Recherche, zum Datenaustausch und zur Kommunikation.	

Veränderungen der Arbeitswelt und Datenschutz“ (b) Vorstellung und Diskussion durch Schülerinnen und Schüler	(K).	
2. Vertiefung des Themas Datenschutz (a) Planspiel: „Datenschutz in vernetzten Informationssystemen“ (b) Erarbeitung grundlegender Begriffe des Datenschutzes (c) Problematisierung und Anknüpfung an die Lebenswelt der Schülerinnen und Schüler (d) Diskussion und Bewertung von Fallbeispielen aus dem Themenbereich „Datenschutz“		*Beispiel:* Fallbeispiele aus dem aktuellen Tagesgeschehen Die Schülerinnen und Schüler bearbeiten Fallbeispiele aus ihrer eigenen Erfahrungswelt oder der aktuellen Medienberichterstattung. *Materialien:* Unterlagen Planspiel im Dropbox-Ordner

2.1.2.2 Qualifikationsphase 1 (Grundkurs und Leistungskurs)

Die zusätzlichen Inhalte und Kompetenzen für den Leistungskurs werden kursiv hervorgehoben.

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

Die Schülerinnen und Schüler

- 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).

Unterrichtsvorhaben Q1-I:

Thema: 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?*

Vorhabenbezogenen Konkretisierung:
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 Schülerinnen und Schülern 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.

Zeitbedarf: 8 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. Wiederholung und Erweiterung der objektorientierten Modellierung und Programmierung durch Analyse und Erweiterung eines kontextbezogenen Beispiels (a) Analyse der Problemstellung (b) Analyse der Modellierung (Implementationsdiagramm) (c) Erweiterung der Modellierung im Implementationsdiagramm (Vererbung, abstrakte Klasse) (d) Kommunikation zwischen mindestens zwei Objekten (grafische Darstellung) (e) Dokumentation von Klassen (f) Implementierung der Anwendung oder von Teilen der Anwendung	Die Schülerinnen und Schüler • 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),	*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 Un-

	• 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).	terrichtsvorhaben Q1.1-Wiederholung (Download Q1-I.1)

Unterrichtsvorhaben Q1-II:

Thema: Modellierung und Implementierung von Anwendungen mit dynamischen, linearen Datenstrukturen

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

Vorhabenbezogene Konkretisierung:
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 Klassen 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.

Zeitbedarf: 20 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. Die Datenstruktur Schlange im Anwendungskontext unter Nutzung der Klasse `Queue` (a) Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen (b) Erarbeitung der Funktionalität der Klasse `Queue` (c) Modellierung und Implementierung der Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse `Queue`	Die Schülerinnen und Schüler • erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A), • analysieren und erläutern Algorithmen und Programme (A), • *implementieren Operationen dynamischer (linearer und nichtlinearer) Datenstrukturen (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	*Beispiel:* Patientenwarteschlange (jeder kennt seinen Nachfolger bzw. alternativ: seinen Vorgänger) Sobald ein Patient in einer Arztpraxis eintrifft, werden sein Name und seine Krankenkasse erfasst. 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)

	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).	
2. Die Datenstruktur Stapel im Anwendungskontext unter Nutzung der Klasse `Stack` (a) Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen (b) Erarbeitung der Funktionalität der Klasse `Stack` (c) Modellierung und Implementierung der Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse `Stack`		*Beispiel:* Heftstapel 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` (a) Erarbeitung der Vorteile der Klasse `List` im Gegensatz zu den bereits bekannten linearen Strukturen (b) 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 Abfahrer 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 ein-

		geordnet. 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, senkrecht von unten nach oben verlaufenden Straßenstück, das von Autos nur einspurig in eine Richtung befahren werden kann, ist ein Haltepunkt markiert, an dem die Ampel steht. Bei einem Klick auf eine Schaltfläche mit der Aufschrift „Heranfahren“ soll ein neues Auto an den Haltepunkt heranfahren bzw. bis an das letzte Auto, das vor dem Haltepunkt wartet. Grünphasen der Ampel werden durch einen Klick auf eine Schaltfläche 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)

Unterrichtsvorhaben Q1-III:

Thema: Suchen und Sortieren auf linearen Datenstrukturen

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

Vorhabenbezogene Konkretisierung:
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.

Zeitbedarf: 16 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. Suchen von Daten in Listen und Arrays (a) Lineare Suche in Listen und in Arrays (b) Binäre Suche in Arrays als Beispiel für rekursives Problemlösen (c) Untersuchung der beiden Suchverfahren hinsichtlich ihrer Effizienz (Laufzeitverhalten, Speicherbedarf)	Die Schülerinnen und Schüler • 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	*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:*

	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). • *entwickeln mit didaktisch orientierten Entwicklungsumgebungen einfache Benutzeroberflächen zur Kommunikation mit einem Informatiksystem (M)*	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 (a) Entwicklung und Implementierung eines einfachen Sortierverfahrens für eine Liste (b) Implementierung eines einfachen Sortierverfahrens für ein Feld (c) 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 (a) Grafische Veranschaulichung der Sortierverfahren (b) Untersuchung der Anzahl der Vergleichsoperationen und des Speicherbedarf bei beiden Sortierverfahren (c) Beurteilung der Effizienz der beiden Sortierverfahren		*Beispiel:* Karteiverwaltung (s.o.) oder *Beispiel*: Bundesjugendspiele (s.o.) *Materialien:* (s.o.)

Unterrichtsvorhaben Q1-IV:

Thema: 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 Konkretisierung:
Ausgehend von einer vorhandenen Datenbank entwickeln Schülerinnen und Schüler für sie relevante Fragestellungen, die mit dem vorhandenen Datenbestand beantwortet werden sollen. Zur Beantwortung dieser Fragestellungen wird die vorgegebene Datenbank von den Schülerinnen und Schülern 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 Schülerinnen und Schüler 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 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.

Zeitbedarf: 20 Stunden

Sequenzierung des Unterrichtsvorhabens

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Nutzung von relationalen Datenbanken (a) Aufbau von Datenbanken und Grundbegriffe • Entwicklung von Fragestellungen zur vorhandenen Datenbank • Analyse der Struktur der vorgegebenen Datenbank und Erarbeitung der Begriffe Tabelle, Attribut, Datensatz, Datentyp, Primärschlüssel, Fremdschlüssel, Datenbankschema (b) SQL-Abfragen • Analyse vorgegebener SQL-Abfragen und Erarbeitung der Sprachelemente von SQL (`SELECT (DISTINCT) …FROM, WHERE, AND, OR, NOT`) auf einer Tabelle • 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`) (c) Vertiefung an einem weiteren Datenbankbeispiel	Die Schülerinnen und Schüler • 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 Informatio-	*Beispiel:* FitnessCenter FitnessCenter ist die Simulation eines Online-Fitness-Centers für den Informatik-Unterricht mit Webfrontends zur Verwaltung der Kunden und der Kurse. Außerdem ist es möglich direkt SQL-Abfragen einzugeben. Es ist auch möglich, die Datenbank herunter zu laden und lokal zu installieren. Unter http://fitnesscenter.schule.de (abgerufen: 08. 03. 2018) findet man den Link zu dem FitnessCenter-System sowie nähere Informationen. *Beispiel*: SQLLearner Unter www.michaelsaul.de/sqllearner/sqllearner.htm (abgerufen: 08.03.2018) wird ein Selbstlernprogramm zum Kennenlernen der SQL-Datenbankabfragesprache angeboten, wobei die SQL-Abfragen interaktiv per Drag-und Drop zusammengestellt werden können. *Beispiel*: SQL-Tutorial Hessen Anhand verschiedener Datenbanken werden in 7 Lektionen und zugehörigen Übungen in einem Selbstlernkurs die Grundlagen von Datenbankabfragen mittels SQL erlernt. Adresse: www.imoodle.de/sqltutorial/index.html (abgerufen: 08.03.2018)

2. Modellierung von relationalen Datenbanken (a) Entity-Relationship-Diagramm • Ermittlung von Entitäten, zugehörigen Attributen, Relationen und Kardinalitäten in Anwendungssituationen und Modellierung eines Datenbankentwurfs in Form eines Entity-Relationship-Diagramms • Erläuterung und Modifizierung einer Datenbankmodellierung (b) Entwicklung einer Datenbank aus einem Datenbankentwurf • Modellierung eines relationalen Datenbankschematas zu einem Entity-Relationship-Diagramm inklusive der Bestimmung von Primär- und Sekundärschlüsseln (c) Redundanz, Konsistenz und Normalformen • Untersuchung einer Datenbank hinsichtlich Konsistenz und Redundanz in einer Anwendungssituation • Überprüfung von Datenbankschemata hinsichtlich der 1. bis 3. Normalform und Normalisierung (um Redundanzen zu vermeiden und Konsistenz zu gewährleisten)	nen aus einen Datenbanksystem zu extrahieren (I), • *implementieren ein relationales Datenbankschema (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).	

Unterrichtsvorhaben Q1-V:

Thema: Sicherheit und Datenschutz in Netzstrukturen

Leitfragen: *Wie werden Daten in Netzwerken übermittelt? Was sollte man in Bezug auf die Sicherheit beachten?*

Vorhabenbezogene Konkretisierung:
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.

Zeitbedarf: 10 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Daten in Netzwerken und Sicherheitsaspekte in Netzen sowie beim Zugriff auf Datenbanken (a) Beschreibung eines Datenbankzugriffs im Netz anhand eines Anwendungskontextes und einer Client-Server-Struktur zur Klärung der Funktionsweise eines Datenbankzugriffs (b) Netztopologien als Grundlage von Client-Server-Strukturen und TCP/IP-Schichtenmodell als Beispiel für eine Paketübermittlung in einem Netz (c) Vertraulichkeit, Integrität, Authentizität in Netzwerken sowie symmetrische und asymmetrische kryptografische Verfahren (Cäsar-, Vigenère-, RSA-	Die Schülerinnen und Schüler • beschreiben und erläutern 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),* • analysieren und erläutern Eigenschaften, *Funktionsweisen* und Einsatzbereiche symmetrischer und asymmetrischer Verschlüsselungsverfahren (A), • untersuchen und bewerten anhand von Fallbeispielen die Auswirkungen des Ein-	*Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben, Verschlüsselung Q1.5 - Zugriff auf Daten in Netzwerken (Download Q1-V.1) Workshop mit Filius zu den Grundlagen des Netzwerkaufbaus, Netzwerkkonfiguration, Routing. (Dropbox IF-Fachschaft)

Verfahren) als Methoden Daten im Netz verschlüsselt zu übertragen	satzes 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), • *entwickeln und erweitern Protokolle zur Kommunikation in einem Client-Server-Netzwerk (M),* • nutzen bereitgestellte Informatiksysteme und das Internet reflektiert zum Erschließen, zur Aufbereitung und Präsentation fachlicher Inhalte (D).	
2. Fallbeispiele zur Datenschutzproblematik und zum Urheberrecht		*Materialien:* *Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1 5* - Datenschutz beim Videocenter, Materialblatt-Datenschutzgesetz (Download Q1-V.2)
3. *Client-Server Netzwerk; Client-Server Anwendung*	• *analysieren und erläutern Algorithmen und Methoden zur Client-Server-Kommunikation (A),* • *erläutern das Prinzip der Nebenläufigkeit (A),* • *entwickeln und implementieren Algorithmen und Methoden zur Client-Server-Kommunikation (I),*	

2.1.2.3 Qualifikationsphase 2 (Grundkurs und Leistungskurs)

Die zusätzlichen Inhalte und Kompetenzen für den Leistungskurs werden kursiv hervorgehoben.

Unterrichtsvorhaben Q2-I:

Thema: 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 Konkretisierung:
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 Umkehrbarbeit 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.

Zeitbedarf: 24 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Analyse von Baumstrukturen in verschiedenen Kontexten (a) Grundlegende Begriffe (Grad, Tiefe, Höhe, Blatt, Inhalt, Teilbaum, Ebene, Vollständigkeit) (b) Aufbau und Darstellung von binären Bäumen anhand von Baumstrukturen in verschiedenen Kontexten	Die Schülerinnen und Schüler • 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“	*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

	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), • *implementieren Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (I),* • stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D), • stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D).	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) Skript (Selbstlernkurs) zu binären Bäumen (Dropbox IF-Fachschaft)
2. Die Datenstruktur Binärbaum im Anwendungskontext unter Nutzung der Klasse `BinaryTree` (a) Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen im Anwendungskontext (b) Modellierung eines Entwurfsdiagramms und Entwicklung eines Im-		*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 Teilbaum, alle die nach dem Namen im aktuellen Teilbaum stehen, sind in dessen rechtem Teilbaum. (Dies gilt für alle Teilbäume.)

plementationsdiagramms (c) Erarbeitung der Klasse `BinaryTree` und beispielhafte Anwendung der Operationen (d) Implementierung der Anwendung oder von Teilen der Anwendung (e) Traversierung eines Binärbaums im Pre-, In- und Postorderdurchlauf		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) Skript (Selbstlernkurs) zu binären Bäumen (Dropbox IF-Fachschaft)
3. Die Datenstruktur binärer Suchbaum im Anwendungskontext unter Verwendung der Klasse BinarySearchTree (a) Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen (b) Modellierung eines Entwurfsdiagramms und Entwicklung eines Implementationsdiagramm, grafische Darstellung eines binären Suchbaums und Erarbeitung der Struktureigenschaften (c) Erarbeitung der Klasse BinarySearchTree und Einführung des Interface Item zur Realisierung einer geeigneten Ordnungsrelation (d) Implementierung der Anwendung oder von Teilen der Anwendung inklusive einer sortierten Ausgabe des Baums		*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 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 Datenbestands in nach Namen sortierter Reihenfolge *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.1 – Binärer Suchbaum (Download Q2-I.3)

		Skript (Selbstlernkurs) zu binären Bäumen (Dropbox IF-Fachschaft)
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.) *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.1 – Anwendung Binärbaum (Download Q2-I.4)

Unterrichtsvorhaben Q2-II:

Thema: 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? im Leistungskurs zusätzlich: wie können Programmiersprachen durch Grammatiken beschrieben werden? Welches Automatenmodell korrespondiert zu dieser Art von Grammatiken?*

Vorhabenbezogene Konkretisierung:
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 Zusammenhang 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. An einem Beispiel werden die Grenzen endlicher Automaten ausgelotet.

im Leistungskurs zusätzlich: Zur Überprüfung der Korrektheit arithmetischer Terme wird eine passende kontextfreie Grammatik analysiert. Anhand ausgewählter kontextfreier Grammatiken wird deren Entsprechung in Form von Syntaxdiagrammen und in Kellerautomaten thematisiert. Für arithmetische Terme erfolgt die Implementation eines passenden Scanners, Parsers und Interpreters (zur Berechnung des Term-Wertes).

Zeitbedarf: 20 (bzw. im Leistungskurs 30) Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien oder Materialien
1. Endliche Automaten (a) Vom Automaten in den Schülerinnen und Schülern bekannten Kontexten zur formalen Beschreibung eines endlichen Automaten (b) Untersuchung, Darstellung und Entwicklung endlicher Automaten	Die Schülerinnen und Schüler • 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), • 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 *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 Kellerautomaten)* (M), • modifizieren Grammatiken regulärer *und kontextfreier* Sprachen (M), • entwickeln zu einer regulären Spra-	*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) Selbstlernskript zur Automatentheorie (Dropbox IF-Fachschaft) Lernprogramm zur Automatenentwicklung: Exorciser (https://www.swisseduc.ch/informatik/exorciser/inhalt.html)
2. Untersuchung und Entwicklung von Grammatiken regulärer Sprachen (a) Erarbeitung der formalen Darstellung regulärer Grammatiken (b) Untersuchung, Modifikation und Entwicklung von Grammatiken (c) Entwicklung von endlichen Automaten zum Erkennen regulärer Sprachen die durch Grammatiken gegeben werden (d) Entwicklung regulärer Grammati-		*Beispiele:* reguläre Grammatik für Wörter mit ungerader Parität, Grammatik für Wörter, die bestimmte Zahlen repräsentieren, Satzgliederungsgrammatik *Materialien: (s.o.)*

ken zu endlichen Automaten	che 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). • *modellieren und implementieren Scanner, Parser und Interpreter zu einer gegebenen regulären Sprache (I)*	
3. Grenzen endlicher Automaten		*Beispiele**:*** Klammerausdrücke, a^nb^n im Vergleich zu $(ab)^n$
4. *Kontextfreie Grammatiken* (a) *Formale Darstellung* (b) *Links-/Rechtsableitung von Worten aus einer kontextfreien Grammatik; Syntaxbaum.* (c) *Syntaxdiagramme als äquivalente Darstellungsform.* (d) *Zugehöriges Automatenmodell: Kellerautomaten.* (e) *Implementation von Scanner, Parser, Interpreter für die Prüfung und Auswertung arithmetischer Terme*		***Beispiele:*** *einfache Formelsprache* *Dangling-Else-Problem* $a^nb^nc^m$ *Syntaxdiagramme: Ausdruck, Term, Plusterm, Faktor, Malfaktor*

Unterrichtsvorhaben Q2-III:

Thema: 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 Konkretisierung:
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.

Zeitbedarf: 12 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien oder Materialien
1. Von-Neumann-Architektur und die Ausführung maschinennaher Programme a) prinzipieller Aufbau einer von Neumann-Architektur mit CPU, Rechenwerk, Steuerwerk, Register und Hauptspeicher b) einige maschinennahe Befehlen und ihre Repräsentation in einem Binär-Code, der in einem Register gespeichert werden kann c) Analyse und Erläuterung der Funktionsweise eines einfachen maschinennahen Programms	Die Schülerinnen und Schüler • 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).	*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 a) Vorstellung des Halteproblems b) Unlösbarkeit des Halteproblems c) Beurteilung des Einsatzes von Informatiksystemen hinsichtlich prinzipiel-		*Beispiel:* Halteproblem *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.3 - Halteproblem

ler Möglichkeiten und prinzipieller Grenzen		(Download Q2-III.2)

Unterrichtsvorhaben Q2-IV:

Thema: Modellierung und Implementierung von Anwendungen mit dynamischen, nichtrekursiven Datenstrukturen (Graphen)

Leitfragen: *Wie funktioniert eine Navigationssoftware? (Wie) können nicht-lineare Datenstrukturen mit Hilfe von linearen Datenstrukturen verwaltet werden?*

Vorhabenbezogene Konkretisierung:
Mit Hilfe von der Fragestellung, wie ein Navigationssystem den kürzesten Weg zwischen A und B ermittelt, wird in das Themengebiet Graphentheorie eingeführt. Ausgehend vom Königsberger Brückenproblem werden zunächst Grundbegriffe und typische Fragestellungen erschlossen. Das Problem der Kommunikation in heutigen Netzwerksystemen sowie die Frage nach der Funktionsweise eines Navigationssystems motivieren die Auseinandersetzung mit der programmiertechnischen Realisierung.

Zeitbedarf: 24 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. Grundbegriffe, klassische Problemstellungen a) Knoten, Kanten (gerichtet/gewichtet), Gewichte, Graph b) Königsberger Brückenproblem, Rundreiseproblematik, Euler- und Hamiltonkreise	Die Schülerinnen und Schüler • erläutern Operationen dynamischer nicht-linearer Datenstrukturen (A), • analysieren und erläutern Algorithmen und Programme (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Program-	Königsberger-Brückenproblem (Folie) Algorithmus von Hierholzer
2. Darstellungsformen und speichertechnische Verwaltung von Graphen		Modellierung eines Kartenausschnittes zur Nut-

a) Adjazenzmatrix, Adjazenzliste b) Knotenliste, Verwaltung der Kanten als zusätzliche Liste in jedem Knoten c) Verwendung der Standardoperationen	men (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), • 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), • 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 nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D),	zung in einem Navigationssystem
3. Graphenalgorithmen a) Tiefensuche und Breitensuche b) Backtracking zur Pfadsuche zwischen zwei Knoten c) Dijkstra-Algorithmus zur Suche des kürzesten Weges zwischen zwei Knoten		BFS- und DFS- Spiel Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q2.IV - Graphen

	• stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D).	

Unterrichtsvorhaben Q2-V:

Wiederholung und Vertiefung ausgewählter Kompetenzen und Inhalte des ersten Jahrs der Qualifikationsphase

2.2 Grundsätze der fachmethodischen und fachdidaktischen Arbeit

In Absprache mit der Lehrerkonferenz sowie unter Berücksichtigung des Schulprogramms hat die Fachkonferenz Informatik des Johannes-Kepler-Gymnasiums die folgenden fachmethodischen und fachdidaktischen Grundsätze beschlossen. In diesem Zusammenhang beziehen sich die Grundsätze 1 bis 14 auf fächerübergreifende Aspekte, die auch Gegenstand der Qualitätsanalyse sind, die Grundsätze 15 bis 21 sind fachspezifisch angelegt.

Überfachliche Grundsätze:

1) Geeignete Problemstellungen zeichnen die Ziele des Unterrichts vor und bestimmen die Struktur der Lernprozesse.
2) Inhalt und Anforderungsniveau des Unterrichts entsprechen dem Leistungsvermögen der Schüler/innen.
3) Die Unterrichtsgestaltung ist auf die Ziele und Inhalte abgestimmt.
4) Medien und Arbeitsmittel sind schülernah gewählt.
5) Die Schüler/innen erreichen einen Lernzuwachs.
6) Der Unterricht fördert eine aktive Teilnahme der Schüler/innen.
7) Der Unterricht fördert die Zusammenarbeit zwischen den Schülern/innen und bietet ihnen Möglichkeiten zu eigenen Lösungen.
8) Der Unterricht berücksichtigt die individuellen Lernwege der einzelnen Schüler/innen.
9) Die Schüler/innen erhalten Gelegenheit zu selbstständiger Arbeit und werden dabei unterstützt.
10) Der Unterricht fördert strukturierte und funktionale Partner- bzw. Gruppenarbeit.
11) Der Unterricht fördert strukturierte und funktionale Arbeit im Plenum.
12) Die Lernumgebung ist vorbereitet; der Ordnungsrahmen wird eingehalten.
13) Die Lehr- und Lernzeit wird intensiv für Unterrichtszwecke genutzt.
14) Es herrscht ein positives pädagogisches Klima im Unterricht.

Fachliche Grundsätze:

15) Der Unterricht unterliegt der Wissenschaftsorientierung und ist dementsprechend eng verzahnt mit seiner Bezugswissenschaft.
16) Der Unterricht ist problemorientiert und soll von realen Problemen ausgehen und sich auf solche rückbeziehen.
17) Der Unterricht folgt dem Prinzip der Exemplarizität und soll ermöglichen, informatische Strukturen und Gesetzmäßigkeiten in den ausgewählten Problemen und Projekten zu erkennen.
18) Der Unterricht ist anschaulich sowie gegenwarts- und zukunftsorientiert und gewinnt dadurch für die Schülerinnen und Schüler an Bedeutsamkeit.
19) Der Unterricht ist handlungsorientiert, d.h. projekt- und produktorientiert angelegt.
20) Im Unterricht werden sowohl für die Schule didaktisch reduzierte als auch reale Informatiksysteme aus der Wissenschafts-, Berufs- und Lebenswelt eingesetzt.
21) Der Unterricht beinhaltet reale Begegnung mit Informatiksystemen.

3 Entscheidungen zu fach- und unterrichtsübergreifenden Fragen

wird im Schuljahr 2015/16 überarbeitet.

4 Qualitätssicherung und Evaluation

Das schulinterne Curriculum stellt keine starre Größe dar, sondern ist als „lebendes Dokument“ zu betrachten. Dementsprechend sind die Inhalte stetig zu überprüfen, um ggf. Modifikationen vornehmen zu können. Die Fachkonferenz (als professionelle Lerngemeinschaft) trägt durch diesen Prozess zur Qualitätsentwicklung und damit zur Qualitätssicherung des Faches bei.

Durch Diskussion der Aufgabenstellung von Klausuren in Fachdienstbesprechungen und eine regelmäßige Erörterung der Ergebnisse von Leistungsüberprüfungen wird ein hohes Maß an fachlicher Qualitätssicherung erreicht.

Das schulinterne Curriculum (siehe 2.1) ist zunächst bis 2017 für den ersten Durchgang durch die gymnasiale Oberstufe nach Erlass des Kernlehrplanes verbindlich. Erstmalig nach Ende der Einführungsphase im Sommer 2015, werden in einer Sitzung der Fachkonferenz Erfahrungen ausgetauscht und ggf. Änderungen für den nächsten Durchgang der Einführungsphase beschlossen, um erkannten ungünstigen Entscheidungen schnellstmöglich entgegenwirken zu können.

Nach Abschluss des Abiturs 2017 wir die Fachkonferenz Informatik auf der Grundlage ihrer Unterrichtserfahrungen eine Gesamtsicht des schulinternen Curriculums vornehmen und ggf. eine Beschlussvorlage für die erste Fachkonferenz des folgenden Schuljahres erstellen.