Schulinterner Lehrplan
zum Kernlehrplan für die gymnasiale Oberstufe

Qualifikationsphase

Fach Informatik
Grundkurs

Inhalt:

1. Übersicht über die Inhaltsfelder und Kompetenzen

Übersicht über die Inhaltsfelder (allgemein)

I. Daten und ihre Strukturierung
II. Algorithmen
III. Formale Sprachen und Automaten
IV. Informatiksysteme
V. Informatik, Mensch und Gesellschaft

Übersicht der konkretisierten Kompetenzerwartungen

ARGUMENTIEREN (A)

Die Schülerinnen und Schüler

- erläutern und begründen methodische Vorgehensweisen, Entwurfs- und Implementationsentscheidungen sowie Aussagen über Informatiksysteme,
- zeigen im Problemlösungsprozess Alternativen auf und begründen ihre Auswahlentscheidungen,
- analysieren und erläutern informatische Modelle,
- analysieren und erläutern Computerprogramme,
- beurteilen die Angemessenheit von Modellierungen und Implementationen,
- erläutern und beurteilen informatische Modelle und Informatiksysteme hinsichtlich ihrer Möglichkeiten, Grenzen und Auswirkungen.

MODELLIEREN (M)

Die Schülerinnen und Schüler

- konstruieren zu kontextbezogenen Problemstellungen informatische Modelle,
- modifizieren und erweitern informatische Modelle,
- wenden im Modellierungsprozess geeignete Lösungsstrategien an.

IMPLEMENTIEREN (I)

Die Schülerinnen und Schüler

- implementieren auf der Grundlage von Modellen oder Modellausschnitten Computerprogramme,
- modifizieren und erweitern Computerprogramme,
- testen und korrigieren Computerprogramme systematisch.

DARSTELLEN UND INTERPRETIEREN (D)

Die Schülerinnen und Schüler

- interpretieren Daten und erläutern Beziehungen und Abläufe, die in Form von textuellen, grafischen oder formalen Darstellungen gegeben sind,
- überführen gegebene textuelle, grafische oder formale Darstellungen informatischer Zusammenhänge in eine der anderen Darstellungsformen,
- stellen informatische Modelle und Abläufe in Texten, Tabellen, Diagrammen, Grafiken und Formalismen dar.

KOMMUNIZIEREN UND KOOPERIEREN (K)

Schülerinnen und Schüler

- verwenden die Fachsprache bei der Kommunikation über informatische Sachverhalte,
- organisieren und koordinieren kooperatives und eigenverantwortliches Arbeiten,
- strukturieren den Arbeitsprozess, vereinbaren Schnittstellen und führen Ergebnisse zusammen,
- beurteilen Arbeitsorganisation, Arbeitsabläufe und Ergebnisse,
- präsentieren Arbeitsabläufe und -ergebnisse adressatengerecht

2. Entscheidungen zum Unterricht

Die nachfolgend dargestellte Umsetzung der verbindlichen Kompetenzerwartungen des Kernlehrplans findet auf zwei Ebenen statt: der Übersichts- und der Konkretisierungsebene. Sie orientiert sich damit an den Vorschlägen des Schulministeriums.

Im „Übersichtsraster Unterrichtsvorhaben“ (Kapitel 2.1) 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 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. 80 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 2.2) 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.

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

Laut Beschluss der Fachkonferenz erfolgt die Umsetzung der Anwendungskontexte in der **Programmiersprache Java** gemäß den Vorgaben des Schulministeriums. Als Standardentwicklungsumgebungen werden dabei BlueJ und Greenfoot verwendet. Die unterrichtende Lehrkraft kann jedoch weitere Entwicklungsumgebungen für Java kontextbezogen einzuführen und verwenden.

Der angegebene Zeitbedarf basiert auf einer Unterrichtsstunde mit der Dauer von 60 Minuten.

2.1 Übersichtsraster Unterrichtsvorhaben

Qualifikationsphase 1	
Unterrichtsvorhaben Q1-I **Thema:** *Wiederholung der objektorientierten Modellierung und Programmierung anhand einer kontextbezogenen Problemstellung mit Neueinführung von abstrakten Klassen und Polymorphie.* **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**: ca. 8 Stunden	***Unterrichtsvorhaben Q1-II*** **Thema:** *Suchen und Sortieren auf linearen Datenstrukturen beginnend mit iterative und vertiefend an Hand der Rekursion.* **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**: ca. 15 Stunden

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

Thema:
Modellierung und Implementierung von Anwendungen mit dynamischen, linearen Datenstrukturen (Vertiefung der Rekursion)

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: ca. 12 Stunden</td>
<td><u>Unterrichtsvorhaben Q1-IV</u>

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: ca. 15 Stunden</td>
</tr>
</table>

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

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: ca. 10 Stunden</td><td></td></tr>
<tr><td colspan="2"><u>Summe Qualifikationsphase 1 ca. 60 Stunden</u></td></tr>
</table>

<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: ca. 18 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:
• Algorithmen
• Endliche Automaten und formale Sprachen

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

Zeitbedarf: ca. 15 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**: ca. 9 Stunden??	
Summe Qualifikationsphase 2 ca. 42 Stunden	

2.2 Konkretisierte Unterrichtsvorhaben

Hinweis: Thema, Inhaltsfelder, inhaltliche Schwerpunkte und Kompetenzen hat die Fachkonferenz des Paul-Klee-Gymnasiums Overath verbindlich vereinbart. In allen anderen Bereichen sind Abweichungen von den vorgeschlagenen Vorgehensweisen bei der Konkretisierung der Unterrichtsvorhaben möglich.

Unterrichtsvorhaben Q1-I

Thema: Wiederholung der objektorientierten Modellierung und Programmierung anhand einer kontextbezogenen Problemstellung mit Neueinführung von abstrakten Klassen und Polymorphie.

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 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: *ca. 6 Stunden*

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Wiederholung der objektorientierten Modellierung und Programmierung anhand einer kontextbezogenen Problemstellung mit Neueinführung von abstrakten Klassen und Polymorphie (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), • verwenden bei der Modellierung geeigneter Problemstellungen die Möglichkeiten der Polymorphie (M), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der	Beispiel: Aufbau eines Adressbuches • Es wird zunächst die Klasse der Objekte (ein Kontakt), die verwaltet werden sollen, bearbeitet. Mit Hilfe der String-Klasse werden Vergleichsmethoden für Objekte dieser Klasse entwickelt und umgesetzt. • Mittels eines Implementationsdiagramms erhalten die SuS eine Übersicht die Klasse, die die Kontakte verwaltet. Hierbei wird der gleichartige Typ des Kontakts in einem Array verwaltet. Zunächst eine Methode zum Füllen des Arrays umgesetzte. • Die SuS entwickeln und definieren weitere Methode zum Veralten von Kontakten. Diese werden dann Umgesetzt. • Als Vorbereitung für das Sortieren wird der Löschvorgang simuliert und mit Hilfe einer Tausche-Methode umgesetzt. • Nutzen einer graphischen Oberfläche zum zur Dateneingabe und Datenausgabe.

	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).	

Unterrichtsvorhaben Q1-II

Thema: *Suchen und Sortieren auf linearen Datenstrukturen iterativ beginnend und vertiefend an Hand der Rekursion.*

Leitfrage: *Wie kann man gespeicherte Informationen effizient ablegen? Wie kann man Informationen günstig wiederfinden? Muss ein Zugriff immer iterativ erfolgen?*

Vorhabenbezogene Konkretisierung:
In einem Anwendungskontext werden zunächst Informationen in einem Feld gesucht. Hierzu werden iterative Sortier- und Suchverfahren vertieft, weiterentwickelt und implementiert bzw. analysiert und hinsichtlich ihres Speicherbedarfes und Zahl der Vergleichsoperationen miteinander verglichen.

Dabei wird die Rekursion als Möglichkeit der Effizienzsteigerung bzgl. Implementationsaufwand und Speicherbedarf thematisiert. Nach der Behandlung des allgemeinen Aufbaus einer Rekursion und der Umsetzung verschiedener allgemeiner Rekursionen wird anschließend das rekursive Sortierverfahren Quicksort analysiert und erläutert.

Zeitbedarf: ca. 15 Stunden

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Suchen von Daten in Arrays (a) Lineare Suche in Arrays (b) Binäre Suche in Arrays als Beispiel für rekursives Problemlösen (c) Entwickeln und Implementieren einfacher Sortierverfahren für ein Feld (d) 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 und rekursive Such- und Sortierverfahren (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse	Beispiel: Suchen im Adressbuch und sortieren der Kontakte • Entwicklung und Implementation eines Algorithmus von Kontaktaden zu einem Namen • Entwicklung und Implementation einer selbst entworfenen Sortierverfahrens • Grafische Darstellung und Implementation des Minsort, Maxsort und Bubblesort
2. Entwicklung und Implementierung von rekursiven Algorithmen (a) Untersuchung des rekursiven Aufbaus von Formel und Definition einer Rekursion (b) Entwicklung und Implantation von rekursiven Formeln (c) Entwicklung und Implementation von grafischen rekursiven Verfahren		Beispiel: Die Summe der ersten n-Zahlen als rekursives Verfahren Beispiel: Fibonacci-Folge Beispiel: Das Pascalsche Dreieck Beispiel grafische Rekursionen mit der Turtle: Zeichnung der Kochkurve und des Pythgoras-Baumes

3. Untersuchung der Effizienz der Sortierverfahren „Sortieren durch direktes Einfügen“ und „Quicksort“ auf Arrays (a) Entwicklung eines rekursiven Sortierverfahren für ein Feld (z.B. Sortieren durch Mischen) (b) Grafische Veranschaulichung der Sortierverfahren (c) Implementation des Quicksort (d) Untersuchung der Anzahl der Vergleichsoperationen und des Speicherbedarf bei beiden Sortierverfahren (e) Beurteilung der Effizienz der beiden Sortierverfahren	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).	Beispiel: grafische Darstellung des Quicksort an einem Zahlenfeld und Implementation des Quicksort Beispiel: Sortierung des Adressbuches mit dem Quicksort

Unterrichsvorhaben Q1-III

Thema: *Modellierung und Implementierung von Anwendungen mit dynamischen, linearen Datenstrukturen (Vertiefung der Rekursion)*

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

Vorhabenbezogene Konkretisierung:
Nach Analyse einer Problemstellung in einem geeigneten Anwendungskontext, um beliebig viele Objekte zu speichern, die zwischen anderen gespeichert sind, wird die Klasse List eingeführt. Dabei wird der zunächst die Verkettung, der Aufbau, der Liste thematisiert und der Unterschied zum Array verdeutlicht, die Klasse modelliert und implementiert. Anschließend wird sie im Anwendungsbeispiel angewendet.
Als Spezialform der Liste, in dem Daten nach dem First-In-First-Out-Prinzip verwaltet werden, wird der Aufbau von Schlangen am Beispiel dargestellt und die Operationen der Klasse Queue erläutert. Anschließend wird für eine vorgegebene Anwendung die Klasse modelliert und implementiert. Eine Klasse für eine den Anforderungen der Anwendung entsprechende Oberfläche wird dabei von der Lehrkraft vorgegeben. Als weitere Spezialform, in der Daten nach dem Last-In-First-Out-Prinzip verwaltet werden, werden Unterschiede zwischen den Datenstrukturen Schlange und Stapel erarbeitet, die Klasse Stapel implementiert und die Anwendung zur Verwendung der Klasse Stapel modifiziert.
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: *ca.* 18 Stunden

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien oder Materialien
1. 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) Implementation der Klasse List. (c) Modellierung und Implementierung einer kontextbezogenen Anwendung unter Verwendung der Klasse List.	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), • 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	Beispiel: Das Adressbuch ist beschränkt auf die Größe des Feldes und kann nicht einfach erweitert werden. Hierfür wird zunächst die Klasse List eingeführt und dann das Adressbuch mit Speicherung der Kontakte in einer Liste modifiziert Beispiel: Erstellung einer Einkaufsliste
2. 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) Implementierung der Klasse Queue (d) Modellierung und Implementierung der Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse Queue		Beispiel: Es wird eine Routenplaner entwickelt, der immer das nächste Ziel angibt. Zum Abarbeiten der Ziele wird die Arbeitsweise einer Schlange erarbeitet, die Klasse Queue implementiert und im Routenplaner verwendet.
3. Die Datenstruktur Stapel im Anwendungskontext unter Nutzung der Klasse Stack		Die Arbeitsweise eines Stapels wird erarbeitet und die Klasse Stack implementiert.

(a) Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen (b) Erarbeitung der Funktionalität der Klasse Stack (c) Implementation der Klasse Stack (d) Modellierung und Implementierung der Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse Stack	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).	Beispiel: Der Routenplaner wird modifiziert, so dass er zum Speichern der Route einen Stapel verwendet
4. Vertiefung - Anwendungen von Listen, Stapeln oder Schlangen in mindestens einem weiteren Kontext (a) Modellierung und Implementierung von Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse List, Queue und/oder Stack (b) Sortierung von Objekte in den Klassen List, Queue und Stack		Beispiele • Benotung von Klausuren. • Erstellung einer Lagerverwaltung

Unterrichsvorhaben 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: *ca.* 15 Stunden

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	*Beispiel:* Surf-Schule Es wird die Ablage von Daten in einer oder mehrere Tabellen untersucht, Vor- und Nachteile sowie die Begriffe Redundanz und Relation erarbeitet. Beispiel: Buchhandlung Die Verwaltung von Buchdaten in einer Relation wird untersucht und ein Tabellenschema mit allen Grundbegriffen erarbeitet. *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. Unter http://dokumentation.videocenter.schule.de/old/video/index.html (abgerufen: 30. 03. 2014) findet man den Link zu dem VideoCenter-System sowie nähere Informationen. Beispiel: Kollegiums-Datenbank

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)	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).	*Beispiel:* Onlinebuchhandel Ein Onlinehändler bietet Bücher zum Verkauf an. Hierfür müssen sich die Kunden online anmelden und erhalten eine Übersicht über die Bücher und können diese bestellen. Aus den Informationen werden die Entitäten, Attribute und Beziehungen abgeleitet, die in einem ER-Modell dargestellt werden. Anschließend werden die Regeln zur Umsetzung des ER-Modells in ein Datenbankschema schrittweise entwickelt und umgesetzt. *Beispiel:* Produkttester Aus Tabellen und Textinformationen wird ein ER-Modell entwickelt für eine Firma, die Produkte von verschiedenen Firmen durch Tester testen lässt. Das Er-Modell wird im Anschluss in ein relationales Datenbankschema übersetzt. *Beispiel:* Abiturprüfungen In einer Tabelle befinden sich fachweise zusammengefasst, die Ergebnisse aller Schüler. Unter Verwendung der Regeln für die Normalisierung wird eine relationales Datenbankschema mit mehreren Tabellen entwickelt.

Unterrichsvorhaben 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: ca. 10 Stunden

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-Verfahren) als Methoden Daten im Netz verschlüsselt zu übertragen	Die Schülerinnen und Schüler • 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).	*Beispiel:* Clientzugriff auf eine Datenbank Die Schülerinnen und Schüler erhalten eine Klassenvorlage mit grafischer Oberfläche zum Zugriff auf die Kollegiums-Datenbank. Es wird der DatabaseConnector eingerichtet, nachdem die Schülerinnen und Schüler die IP-Informationen ermittelt haben. Mit der Klasse QueryResult testen Sie, die im Unterrichtsvorhaben IV entwickelten SQL-Abfragen. *Beispiel:* RSA-Verfahren Die Schülerinnen und Schüler entwickeln einen KeyGenerator für 10-stellige Primzahlen nach dem RSA-Verfahren. Anschließend wird die Klasse genutzt um Schlüssel für die Codierung und Decodierung kurze Buchstabenfolgen in einer Verschlüsselungsklasse erzeugen und Nachrichten getauscht.
2. **Fallbeispiele zur Datenschutzproblematik und zum Urheberrecht**		*Materialien:* *Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1 5* - Datenschutz beim Videocenter, Materialblatt-Datenschutzgesetz (Download Q1-V.2)

Unterrichsvorhaben 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: ca. 18 Stunden

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	*Beispiel:* Namensbaum Der komplette eigene Name wird buchstabenweise in einer binären Baumstruktur gespeichert. Zum Einfügen der Buchstaben wird er lexikografische Vergleich von Buchstaben genutzt. Der Baum wird grafisch dargestellt und auf Tiefe und Vollständigkeit untersucht. 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.

	„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).	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.
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 Implementationsdiagramms (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		*Beispiel:* Ahnenbaum Der Ahnenbaum einer fiktiven Person soll erstellt werden. Dazu wird die Klasse Ahne implementiert, deren Objekte im Binärbaum gespeichert werden. Zum Einfügen eines weiteren Ahnen muss die Vorfahrenlinie mit übergeben werden und der Ahne kann nur eingetragen werden, wenn die Ahnenlinie bis zum neuen Eintrag nicht unterbrochen ist. *Beispiel*: Zahlenbaum *als binärer Baum* In einem *binären Baum* werden ganze Zahlen der Größe nach geordnet abgespeichert. Alle Zahlen, die nach dieser Ordnung kleiner die Zahl im aktuellen Teilbaum sind, sind in dessen linkem Teilbaum, alle die größer als die Zahl im aktuellen Teilbaum stehen, sind in dessen rechtem Teilbaum. (Dies gilt für alle Teilbäume.) Folgende Funktionalitäten werden benötigt: • Einfügen ganzen Zahlen in den Baum • Suchen nach einer bestimmten Zahl • Ausgabe des kompletten Datenbestands in sortierter Reihenfolge • Löschen einer Zahl aus dem Baum

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*: Benutzerverwaltung als *Suchbaum* In einem binären *Suchbaum* werden für eine Anmeldung der Benutzernamen und das Passwort lexikographisch nach dem Benutzernamen geordnet abgespeichert. Alle Benutzernamen, die nach dieser Ordnung vor dem Benutzernamen im aktuellen Teilbaum stehen, sind in dessen linkem Teilbaum, alle die nach dem Benutzeramen im aktuellen Teilbaum stehen, sind in dessen rechtem Teilbaum. (Dies gilt für alle Teilbäume.) Folgende Funktionalitäten werden benötigt: • Einfügen der Anmeldeinformationen in den Baum • Suchen nach einem Passwort über den Schlüssel Benutzername • Ausgabe des kompletten Datenbestands in nach Benutzernamen sortierter Reihenfolge • Löschen von Benutzerinformationen • Ändern eines Passwortes für einen Benutzernamen
4. Übung und Vertiefungen der Verwendung von Binärbäumen oder binären Suchbäumen anhand weiterer Problemstellungen		*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. *oder* *Beispiel*: Lottospiel Es soll eine Anwendung entwickelt werden, in der eine Ziehung der Lottozahlen in einem Suchbaum (Objekt der Klasse BinarySearchTree) verwaltet. Bei jeder neuen Ziehung muss der Suchbaum neu erstellt werden und dann die gezogenen Zahlen der Größe nach aufsteigend angezeigt werden. *oder* *Beispiel:* Ahnenbaum (s.o.)

Unterrichsvorhaben 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?*

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.

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

Zeitbedarf: ca. 15 Stunden

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 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),	*Beispiele:* Parkschein-Automat, Kaffee-Automat: Untersuchung der Zustandsänderung des Kaffee-Automaten für den Einwurf einer Münze bzw. Drücken einer Taste (Transduktoren) mit Übergangsfunktion und Ausgabefunktion. Untersuchung auf deterministisch und Endlichkeit einer Mausfalle. Akzeptor für bestimmte Zahlen, Akzeptor für Teilwörter in längeren Zeichenketten, Akzeptor für Terme
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 Grammatiken zu endlichen Automaten		*Beispiele:* reguläre Grammatik für Wörter mit ungerader Parität, Grammatik für Wörter, die bestimmte Zahlen repräsentieren, Aufstellen von Formeln

3. Grenzen endlicher Automaten	• 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).	*Beispiele:* Klammerausdrücke, a^nb^n im Vergleich zu $(ab)^n$ Rührwerk: Eine Futtermischung wird unter bestimmten Vorgaben erstellt und auf Endlichkeit bzgl. der Anweisungen untersucht.

Unterrichsvorhaben 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: ca. 9 Stunden

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 eingegebenen Zahl mit einem Rechnermodell
2. Grenzen der Automatisierbarkeit a) Vorstellung des Halteproblems b) Unlösbarkeit des Halteproblems c) Beurteilung des Einsatzes von Informatiksystemen hinsichtlich prinzipieller Möglichkeiten und prinzipieller Grenzen		*Beispiel:* Halteproblem

Stand 08.2022