Schulinterner Lehrplan
zum Kernlehrplan für die gymnasiale Oberstufe

Informatik

(Stand: 06.08.2022)

Inhalt

1 Die Fachgruppe Informatik des Immanuel-Kant-Gymnasiums Heiligenhaus

Beim Immanuel-Kant-Gymnasium handelt es sich um eine vierzügige Schule in Heiligenhaus mit zurzeit ca. 800 Schülerinnen und Schülern und 60 Lehrerinnen und Lehrern. Das Einzugsgebiet der Schule umfasst Heiligenhaus sowie umliegende Städte.

Das Fach Informatik wird am Immanuel-Kant-Gymnasium ab der Jahrgangsstufe 5 unterrichtet.

Im Wahlpflichtbereich II (WP II) wird Informatik als Fachkombination mit Mathematik dreistündig unterrichtet. In der zweijährigen Laufzeit dieser Kurse wird in altersstufengerechter Weise unter anderem auf Grundlagen der Algorithmik am Beispiel didaktischer Lernumgebungen eingegangen.

In der Sekundarstufe II bietet das Immanuel-Kant-Gymnasium für die eigenen Schülerinnen und Schüler in allen Jahrgangsstufen jeweils Grundkurse in Informatik an.

Um insbesondere Schülerinnen und Schülern gerecht zu werden, die in der Sekundarstufe I keinen Informatikunterricht besucht haben, wird in Kursen der Einführungsphase besonderer Wert darauf gelegt, dass keine Vorkenntnisse aus der Sekundarstufe I zum erfolgreichen Durchlaufen des Kurses erforderlich sind.

Der Unterricht der Sekundarstufe II wird mit Hilfe der Programmiersprache Java durchgeführt.

Durch projektartiges Vorgehen, offene Aufgaben und Möglichkeiten, Problemlösungen zu verfeinern oder zu optimieren, entspricht der Informatikunterricht 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 Immanuel-Kant-Gymnasiums aus vier Lehrkräften, denen zwei Computerräume mit je 32 Computerarbeitsplätzen zur Verfügung stehen. Alle Arbeitsplätze sind an das schulinterne Rechnernetz angeschlossen, so dass Schülerinnen und Schüler Zugang zum zentralen Server der Schule zum Zugriff auf ihre eigenen Daten, zur Recherche im Internet oder zur Bearbeitung schulischer Aufgaben haben.

Der Unterricht erfolgt im 45-Minuten-Takt. Die Kursblockung sieht grundsätzlich für Grundkurse eine Doppelstunde und eine Einzelstunde vor.

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

I) Einführungsphase

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

Thema:
Grundlagen der objektorientierten Analyse, Modellierung und Implementierung

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

Inhaltsfelder:
• Daten und ihre Strukturierung
• Algorithmen

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

Zeitbedarf: 33 Stunden</td>
<td><u>Unterrichtsvorhaben E-II</u>

Thema:
Modellierung und Implementierung von Klassen- und Objektbeziehungen und Sichtbarkeitsbereiche von Attributen und Methoden

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

Inhaltsfelder:
• Daten und ihre Strukturierung

Inhaltliche Schwerpunkte:
• Objekte und Klassen

Zeitbedarf: 10 Stunden</td>
</tr>
</table>

Einführungsphase	
Unterrichtsvorhaben E-III **Thema**: *Such- und Sortieralgorithmen anhand kontextbezogener Beispiele* **Zentrale Kompetenzen:** • Modellieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Daten und ihre Strukturierung • Algorithmen **Inhaltliche Schwerpunkte:** • Entwurf und Implementierung einfacher Algorithmen • Algorithmen zum Suchen und Sortieren • Analyse der Effizienz **Zeitbedarf**: 13 Stunden	*Unterrichtsvorhaben E-IV* **Thema**: *Das Konzept der Vererbung in Java* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren **Inhaltsfelder:** • Daten und ihre Strukturierung • Algorithmen **Inhaltliche Schwerpunkte:** • Objekte und Klassen • Syntax und Semantik einer Programmiersprache • Analyse, Entwurf und Implementierung einfacher Algorithmen **Zeitbedarf**: 6 Stunden

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

Thema:
Geschichte der digitalen Datenverarbeitung bis zur Von-Neumann-Architektur und Grundlagen des Datenschutzes

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

Inhaltsfelder:
• Informatik, Mensch und Gesellschaft
• Informatiksysteme
• Formale Sprachen und Automaten

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

Zeitbedarf: 12 Stunden</td><td></td></tr>
<tr><th colspan="2">Summe Einführungsphase: 74</th></tr>
</table>

II) Qualifikationsphase (Q1 und Q2) – GRUNDKURS

<table>
<tr><th colspan="2">Qualifikationsphase 1</th></tr>
<tr>
<td>

<u>*Unterrichtsvorhaben Q1-I*</u>

Thema:
Wiederholung der objektorientierten Modellierung und Programmierung

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: 4 Stunden

</td>
<td>

<u>*Unterrichtsvorhaben Q1-II*</u>

Thema:
Modellierung und Implementierung von Anwendungen mit dynamischen, linearen Datenstrukturen und Betrachtung von Such- und Sortieralgorithmen auf 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: 28 Stunden

</td>
</tr>
</table>

<table>
<tr><th colspan="2">Qualifikationsphase 1</th></tr>
<tr>
<td><u>Unterrichtsvorhaben Q1-III</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: 26 Stunden</td>
<td><u>Unterrichtsvorhaben Q1-IV</u>

Thema:
Sicherheit und Datenschutz in Netzstrukturen

Zentrale Kompetenzen:
• Argumentieren
• Implementieren
• 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: 16 Stunden</td>
</tr>
<tr><td colspan="2">Summe Qualifikationsphase 1: 74 Stunden</td></tr>
</table>

Qualifikationsphase 2	
Unterrichtsvorhaben Q2-I **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**: 16 Stunden	*Unterrichtsvorhaben Q2-II* **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**: 18 Stunden

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

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</td><td><u>Unterrichtsvorhaben Q2-IV</u>

Thema:
Wiederholung und Vertiefung ausgewählter Kompetenzen und Inhalte der Qualifikationsphase

Zeitbedarf: 10 Stunden</td></tr>
<tr><th colspan="2">Summe Qualifikationsphase 2: 56 Stunden</th></tr>
</table>

II) Qualifikationsphase (Q1 und Q2) – LEISTUNGSKURS

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

Thema:
Wiederholung der objektorientierten Modellierung und Programmierung unter Hinzunahme zweidimensionaler Arrays und optionaler GUI-Gestaltung

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: 15 Stunden
</td>
<td>
<u>Unterrichtsvorhaben Q1-II</u>

Thema:
Modellierung und Implementierung von dynamischen, linearen Datenstrukturen und Anwendungen mit dynamischen, linearen Datenstrukturen in kontextbezogenen Problemstellungen, auch unter Berücksichtigung der Gestaltung einer Benutzungsoberfläche sowie Modellierung, Implementierung, Analyse und Beurteilung von Such- und Sortierverfahren unterschiedlicher Komplexitätsklassen in kontextbezogenen Problemstellungen

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: 40 Stunden
</td>
</tr>
</table>

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

Thema:
Modellierung, Implementierung und Nutzung von relationalen Datenbanken in Anwendungskontexten sowie projektorientierte Softwareentwicklung am Beispiel einer Anwendung mit Datenbankanbindung

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

Zeitbedarf: 40 Stunden</td>
<td><u>Unterrichtsvorhaben Q1-IV</u>

Thema:
Modellierung und Implementierung von dynamischen nicht-linearen Datenstrukturen und von Anwendungen mit dynamischen nicht-linearen Datenstrukturen in kontextbezogenen Problemstellungen

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: 40 Stunden</td>
</tr>
<tr><td colspan="2">Summe Qualifikationsphase 1: 135 Stunden</td></tr>
</table>

Qualifikationsphase 2	
Unterrichtsvorhaben Q2-I **Thema:** *Sicherheit und Datenschutz in Informatiksystemen sowie Grenzen und Auswirkungen der Automatisierung* **Zentrale Kompetenzen:** • Argumentieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Informatiksysteme • Informatik, Mensch und Gesellschaft **Inhaltliche Schwerpunkte:** • Sicherheit • Nutzung von Informatiksystemen • Wirkungen der Automatisierung • Grenzen der Automatisierung **Zeitbedarf**: 20 Stunden	*Unterrichtsvorhaben Q2-II* **Thema:** *Grundlagen von Automaten und formalen Sprachen sowie die Modellierung und Implementierung eines Parsers zu einer formalen Sprache* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Endliche Automaten und formale Sprachen **Inhaltliche Schwerpunkte:** • Endliche Automaten und Kellerautomaten • Grammatiken regulärer Sprachen und kontextfreier Sprachen • Scanner, Parser und Interpreter für eine reguläre Sprache • Möglichkeiten und Grenzen von Automaten und formalen Sprachen **Zeitbedarf**: 30 Stunden

Qualifikationsphase 2	
Unterrichtsvorhaben Q2-III **Thema:** *Grundlagen der Netzwerkkommunikation sowie Modellierung und Implementierung von Client-Server-Anwendungen in kontextbezogenen Problemstellungen* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Informatiksysteme • Algorithmen **Inhaltliche Schwerpunkte:** • Algorithmen in ausgewählten Kontexten • Einzelrechner und Rechnernetzwerke • Nutzung von Informatiksystemen **Zeitbedarf**: 25 Stunden	*Unterrichtsvorhaben Q2-IV* **Thema:** *Prinzipielle Arbeitsweise eines Computers sowie Modellierung und Implementierung eines Scanners, Parsers und Interpreters für eine einfache maschinennahe Programmiersprache* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Algorithmen • Formale Sprachen und Automaten • Informatiksysteme • Informatik, Mensch und Gesellschaft **Inhaltliche Schwerpunkte:** • Analyse, Entwurf und Implementierung von Algorithmen • Scanner, Parser und Interpreter für eine reguläre Sprache • Einzelrechner und Rechnernetzwerke **Zeitbedarf**: 15 Stunden

Qualifikationsphase 2	
Unterrichtsvorhaben Q2-V **Thema:** *Wiederholung und Vertiefung ausgewählter Kompetenzen und Inhalte der Qualifikationsphase* **Zeitbedarf:** 10 Stunden	
Summe Qualifikationsphase 2: 100 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 Immanuel-Kant-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. In diesen Bereichen sind Abweichungen von den vorgeschlagenen Vorgehensweisen möglich.

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

https://www.standardsicherung.schulministerium.nrw.de/cms/zentralabitur-gost/faecher/fach.php?fach=15 (abgerufen: 06.08.2022)

I) 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: Grundlagen der objektorientierten Analyse, Modellierung und Implementierung

Leitfragen: *Wie lassen sich Gegenstandsbereiche informatisch modellieren und realisieren?*

Vorhabenbezogene Konkretisierung:
Ein zentraler Bestandteil des Informatikunterrichts der Einführungsphase ist die Objektorientierte Programmierung. Dieses Unterrichtsvorhaben führt in die Grundlagen der Analyse, Modellierung und Implementierung in diesem Kontext ein.
Das Ziel dieses Unterrichtsvorhabens besteht darin, das Verhalten von Objekten flexibel zu programmieren. Ein erster Schwerpunkt liegt dabei auf der Erarbeitung von Kontrollstrukturen. Die Strukturen Wiederholung und bedingte Anweisung werden an einfachen Beispielen eingeführt und anschließend anhand komplexerer Problemstellungen erprobt. Da die zu entwickelnden Algorithmen zunehmend umfangreicher werden, werden systematische Vorgehensweisen zur Entwicklung von Algorithmen thematisiert.
Ein zweiter Schwerpunkt des Unterrichtsvorhabens liegt auf dem Einsatz von Variablen. Beginnend mit lokalen Variablen, die in Methoden und Zählschleifen zum Einsatz kommen, über Variablen in Form von Parametern und Rückgabewerten von Methoden, bis hin zu Variablen, die die Attribute einer Klasse realisieren, lernen die Schülerinnen und Schüler die unterschiedlichen Einsatzmöglichkeiten des Variablenkonzepts anzuwenden.
Beziehungen zwischen Klassen sollen zu einem späteren Zeitpunkt (Unterrichtsvorhaben EF-2) thematisiert werden.
Anschließend können die wesentlichen Eigenschaften von Algorithmen wie z.B. Korrektheit, Terminiertheit, Effizienz und Verständlichkeit sowie die Schritte einer Algorithmenentwicklung erarbeitet werden (Klärung der Anforderung, Visualisierung, Zerlegung in Teilprobleme).

Zeitbedarf: ca. 33 Stunden

Sequenzierung des Unterrichtsvorhabens: nächste Seite

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien, Methoden
1. Einführung in die objektorientierte Progammierung mit Java (a) Klassen (b) Objekte (c) Methoden (d) Anweisungen (e) Stylekonventionen	Die Schülerinnen und Schüler • ermitteln bei der Analyse einfacher Problemstellungen Objekte, ihre Eigenschaften und ihre Operationen (M) • analysieren und erläutern einfache Algorithmen und Programme (A) • modifizieren einfache Algorithmen und Programme (I) • nutzen die im Unterricht eingesetzten Informatiksysteme selbstständig, sicher, zielführend und verantwortungsbewusst (D) • implementieren einfache Algorithmen unter Beachtung der Syntax und Semantik einer Programmiersprache (I) • entwerfen einfache Algorithmen und stellen sie umgangssprachlich und grafisch dar (M) • testen Programme schrittweise anhand von Beispielen (I) • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen zu (M) • dokumentieren Klassen durch Beschreibung der Funktionalität der Methoden (D) • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I) • implementieren Algorithmen unter Verwendung von Variablen und Wertzuweisungen, Kontrollstrukturen sowie Methodenaufrufen (I)	*Beispiel*: Wombat-Tutorial in Greenfoot
2. Kontrollstrukturen und Algorithmenentwurf (a) Bedingte Anweisungen (b) Logische Verknüpfungen (c) While- und For-Schleifen (d) Variablen (int) (e) Struktogramme		*Beispiel*: Wombat-Tutorial in Greenfoot *Möglich*: Partnerpuzzle zum Thema bedingte Anweisungen und log. Verknüpfungen Mögliche Wiederholung und Vertiefung mit Roboter-Szenario nach den Herbstferien
3. Methoden und Variablen (a) Aufbau von Methoden (b) Konstruktor (c) Variablen (d) Datentypen		*Beispiel*: Roboter-Szenario in Greenfoot *Möglich*: Projekt steuerbarer Weihnachtsmann in Greenfoot (Spiel)
4. Algorithmus (a) Eigenschaften von Algorithmen (b) Algorithmen aus Alltagsbeispielen		

Unterrichtsvorhaben EF-II

Thema: Modellierung und Implementierung von Klassen- und Objektbeziehungen und Sichtbarkeitsbereiche von Attributen und Methoden

Leitfragen: *Wie werden realistische Systeme anforderungsspezifisch reduziert, als Entwurf modelliert und implementiert?*

Vorhabenbezogene Konkretisierung:
Das Unterrichtsvorhaben hat die Entwicklung von Objekt -und Klassenbeziehungen zum Schwerpunkt. Dazu werden, ausgehend von der Realität, über Objektidentifizierung und Entwurf bis hin zur Implementation kleine Softwareprodukte in Teilen oder ganzheitlich erstellt.
Zuerst identifizieren die Schülerinnen und Schüler Objekte und stellen diese dar. Aus diesen Objekten werden Klassen und ihre Beziehungen in Entwurfsdiagrammen erstellt.
Nach diesem ersten Modellierungsschritt werden über Klassendokumentationen Implementationsdiagramme entwickelt. Danach werden die Implementationsdiagramme in Java programmiert.
In diesem Unterrichtsvorhaben werden die grundlegenden Begriffe der Objektorientierung und Modellierungswerkzeuge wie Objektdiagramme und Klassendiagramme eingeführt.

Zeitbedarf: ca. 10 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien, Methoden
1. Unified Modeling Language (UML-Diagramme) (a) Klassen und Objekte (b) Entwurfsdiagramm (c) Implementationsdiagramm (d) Objektdiagramm	Die Schülerinnen und Schüler • ermitteln bei der Analyse einfacher Problemstellungen Objekte, ihre Eigenschaften und ihre Operationen (M) • nutzen die im Unterricht eingesetzten Informatiksysteme selbstständig, sicher, zielführend und verantwortungsbewusst (D) • implementieren einfache Algorithmen unter Beachtung der Syntax und Semantik einer Programmiersprache (I) • entwerfen einfache Algorithmen und stellen sie umgangssprachlich und grafisch dar (M) • interpretieren Fehlermeldungen und korrigieren den Quellcode (I) • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen zu (M) • modellieren Klassen mit ihren Attributen, ihren Methoden und Assoziationsbeziehungen (M) • analysieren und erläutern eine objektorientierte Modellierung (A) • ordnen Klassen, Attributen und Methoden ihren Sichtbarkeitsbereich zu (M) • stellen den Zustand eines Objekts dar (D) • stellen Klassen- und Assoziationsbeziehungen in Diagrammen grafisch dar (D)	Mögliche Überleitung von Greenfoot zu BlueJ Mögliches Programm für Diagramme: DIA
2. Datenkapselung (a) Sichtbarkeitsbereiche (public, private) (b) get- und set-Methoden		*Beispielprogramm*: Hogwarts-Szenario in BlueJ

Unterrichtsvorhaben EF-III

Thema: Such- und Sortieralgorithmen

Leitfragen: *Wie können Objekte bzw. Daten effizient gesucht und sortiert werden?*

Vorhabenbezogene Konkretisierung:
Zunächst lernen die Schülerinnen und Schüler Arrays als eine erste Datenstruktur kennen.
Im zweiten Teil dieses Unterrichtsvorhaben wird die Datenstruktur Array erneut bei der Erarbeitung von Such- und Sortieralgorithmen aufgegriffen. Der Schwerpunkt des Vorhabens liegt dabei auf den Algorithmen selbst und nicht auf deren Implementierung in einer Programmiersprache, auf die in diesem Vorhaben vollständig verzichtet werden soll.
Daran anschließend lernen die Schülerinnen und Schüler Strategien des Sortierens (Selection Sort, Insertion Sort, Bubble Sort, Quick Sort) kennen. Der Einstieg über die intuitiven Sortierstrategien dient dazu, die jeweiligen Strategien handlungsorientiert zu erkunden und intuitive Effizienzbetrachtungen der Suchalgorithmen vorzunehmen.
Schließlich wird die Effizienz unterschiedlicher Sortierverfahren beurteilt.
Durch die Analyse von verschiedenen Fehlermeldungen des Java-Compilers werden die Schülerinnen und Schüler in der selbstständigen Fehlerkorrektur geschult.

Zeitbedarf: ca. 13 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien, Methoden
1. Statische Datenstruktur Array (a) Eigenschaften (b) Operationen auf Arrays (c) Arrays mit komplexen Datentypen	Die Schülerinnen und Schüler • nutzen die im Unterricht eingesetzten Informatiksysteme selbstständig, sicher, zielführend und verantwortungsbewusst (D) • implementieren einfache Algorithmen unter Beachtung der Syntax und Semantik einer Programmiersprache (I) • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare Datensammlungen zu (M) • analysieren Such- und Sortieralgorithmen und wenden sie auf Beispiele an (D) • entwerfen einen weiteren Algorithmus zum Sortieren (M) • beurteilen die Effizienz von Algorithmen am Beispiel von Sortierverfahren hinsichtlich Zeitaufwand und Speicherplatzbedarf • interpretieren Fehlermeldungen und korrigieren den Quellcode (I) • nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung und gemeinsamen Verwendung von Daten unter Berücksichtigung der Rechteverwaltung (K)	*Beispiel*: jApplet in BlueJ analysieren und modifizieren *Beispiel*: Bundesliga-Tabelle in BlueJ
2. Suchen und Sortieren (a) Intuitive Sortierstrategien (b) Sortierverfahren (Insertion-, Selection-, Bubble- und Quick Sort) (c) Laufzeitanalyse		*Möglich*: Gruppenpuzzle zu Sortierverfahren Empfohlene Datenstruktur: Array *Möglich*: Glossar-Erstellung in Gruppen zu allen Begrifflichkeiten und Fachkonzepten
3. Fehlermeldungen in Java (a) Analyse häufiger Fehlermeldungen in Java (b) Fehlerkorrektur im Quelltext		

Unterrichtsvorhaben EF-IV

Thema: Das Konzept der Vererbung in Java

Leitfragen: *Wie lassen sich komplexere Beziehungen zwischen Objekten und Klassen realisieren?*

Vorhabenbezogene Konkretisierung:
In diesem Unterrichtsvorhaben wird das Prinzip der Vererbung im objektorientierten Sinne angesprochen. Dazu werden die wichtigsten Varianten der Vererbung anhand eines Beispielprojekts (z.B. Fantasy-Rollenspiel) erarbeitet. Zunächst wird die Vererbung als Spezialisierung im Sinne einer einfachen Erweiterung einer Oberklasse vorgestellt. Daraufhin wird das Verständnis von Vererbung um den Aspekt der späten Bindung erweitert, indem Methoden einer Oberklasse überschrieben werden. Modellierungen sollen in Form von Implementationsdiagrammen erstellt werden.
Zum Abschluss kann auf das Prinzip der abstrakten Klassen und Methoden eingegangen werden. Dieser Inhalt ist aber nicht obligatorisch für die Einführungsphase.

Zeitbedarf: ca. 6 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien, Methoden
1. Vererbung (a) Vererbungshierarchie (Ober- und Unterklasse) (b) Implementierung in Java (c) Polymorphie (Überschreiben von Methoden) (d) Instanceof-Abfrage → Typecast (e) Abstrakte Klassen und Methoden	Die Schülerinnen und Schüler • modellieren Klassen mit ihren Attributen, ihren Methoden und Assoziationsbeziehungen (M) • modellieren Klassen unter Verwendung von Vererbung (M) • stellen die Kommunikation zwischen Objekten grafisch dar (M) • stellen Klassen, Assoziations- und Vererbungsbeziehungen in Diagrammen grafisch dar (D) • implementieren einfache Algorithmen unter Beachtung der Syntax und Semantik einer Programmiersprache (I) • interpretieren Fehlermeldungen und korrigieren den Quellcode (I) • analysieren und erläutern eine objektorientierte Modellierung (A)	*Beispiel*: Fantasy-Rollenspiel in BlueJ

Unterrichtsvorhaben EF-V

Thema: Geschichte der digitalen Datenverarbeitung bis zur Von-Neumann-Architektur und Einführung in die Grundlagen, Anwendungsgebiete und Verarbeitung binärer Codierung

Leitfragen: *Welche Entwicklung durchlief die moderne Datenverarbeitung? Wie werden binäre Informationen gespeichert und wie können sie davon ausgehend weiter verarbeitet werden? Wie ist der Grundaufbau einer digitalen Rechenmaschine?*

Vorhabenbezogene Konkretisierung:
Das folgende Unterrichtsvorhaben stellt den Abschluss der Einführungsphase dar.
Schülerinnen und Schüler sollen zunächst selbstständig informatische Themenbereiche aus dem Kontext der Geschichte der Datenverarbeitung erarbeiten. Diese Themenbereiche werden in Kleingruppen bearbeitet und in Form von Präsentationen vorgestellt. Schülerinnen und Schüler sollen dabei mit Unterstützung des Lehrenden selbstständige Recherchen zu ihren Themen anstellen und auch eine sinnvolle Eingrenzung ihres Themas vornehmen.
Anschließend wird verstärkt auf den Aspekt des Datenschutzes eingegangen. Dazu wird das Bundesdatenschutzgesetz in Auszügen behandelt und auf schülernahe Beispielsituationen zur Anwendung gebracht. Dabei steht keine formale juristische Bewertung der Beispielsituationen im Vordergrund, die im Rahmen eines Informatikunterrichts auch nicht geleistet werden kann, sondern vielmehr eine persönliche Einschätzung von Fällen im Geiste des Datenschutzgesetzes.

Zeitbedarf: ca. 12 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien, Methoden
1. Selbstständige Erarbeitung von Themen durch die Schülerinnen und Schüler (a) Themen zur Erarbeitung in Kleingruppen (b) Vorstellung und Diskussion durch Schülerinnen und Schüler	Die Schülerinnen und Schüler • bewerten anhand von Fallbeispielen die Auswirkungen des Einsatzes von Informatiksystemen (A) • erläutern wesentliche Grundlagen der Geschichte der digitalen Datenverarbeitung (A) • stellen ganze Zahlen und Zeichen in Binärcodes dar (D) • interpretieren Binärcodes als Zahlen und Zeichen (D) • nutzen das Internet zur Recherche, zum Datenaustausch und zur Kommunikation (K) • beschreiben und erläutern den strukturellen Aufbau und die Arbeitsweise singulärer Rechner am Beispiel der „Von-Neumann-Architektur“ (A)	*Mögliche Themen:* • „Eine kleine Geschichte der Digitalisierung: vom Morsen zum modernen Digitalcomputer“ • „Von Nullen, Einsen und mehr: Stellenwertsysteme und wie man mit ihnen rechnet“ • „Kodieren von Texten und Bildern: ASCII, RGB und mehr“ • „Technische Grundlagen des Internets“ • „Auswirkungen der Digitalisierung und des Einsatzes von Informationssystemen auf die Gesellschaft“ • „Projektmanagement am Beispiel ... (z.B. Spieleprogrammierung)“ • „Der Von-Neumann-Rechner“ • „Von der Schrift zum Smartphone“
2. Vertiefung des Themas Datenschutz (a) Erarbeitung grundlegender Begriffe des Datenschutzen (b) Problematisierung und Anknüpfung an die Lebenswelt der Schülerinnen und Schüler (c) Diskussion und Bewertung von Fallbeispielen aus dem Themenbereich „Datenschutz“		

II) Qualifikationsphase – GRUNDKURS

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 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 zunächst ein Entwurfsdiagramm entwickelt werden. Die Schülerinnen und Schülern erläutern und modifizieren den ersten Entwurf und modellieren weitere Klassen und Methoden für eine entsprechende Anwendung. Klassen und ihre Beziehungen werden anschließend in einem Implementationsdiagramm dargestellt. Dabei werden Sichtbarkeitsbereiche zugeordnet.

Zeitbedarf: ca. 4 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien, Methoden
1. Wiederholung und Erweiterung der objektorientierten Modellierung und Programmierung durch Analyse und Erweiterung eines kontextbezogenen Beispiels (a) Analyse der Problemstellung (b) Analyse der Modellierung (Entwurfs & Implementationsdiagramm) (c) Präsentation und Diskussion über Modellierungen (unter Verwendung wichtiger Fachbegriffe) (d) (Optional: Implementierung der Anwendung oder von Teilen der Anwendung)	Die Schülerinnen und Schüler • analysieren und erläutern objektorientierte Modellierungen (A) • modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (M) • ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M) • stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D) • dokumentieren Klassen (D) • nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung und gemeinsamen Verwendung von Daten unter Berücksichtigung der Rechteverwaltung (K) • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare Datensammlungen zu (M)	*Beispiel*: Modellierung eines Fußballturniers

Unterrichtsvorhaben Q1-II

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

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

Vorhabenbezogene Konkretisierung:
Zu Beginn der Unterrichtseinheit werden die Grenzen von statischen Datenstrukturen am Beispiel des Arrays thematisiert und von den Schülern herausgearbeitet. Daraufhin folgt ein Übergang zu dynamischen Datenstrukturen. Nach Analyse einer Problemstellung in einem geeigneten Anwendungskontext, in dem Daten nach dem First-In-First-Out-Prinzip verwaltet werden, wird der Aufbau von Schlangen an einem Beispiel dargestellt und die Operationen der Klasse Queue erläutert. Anschließend werden für die Anwendung notwendige Klassen modelliert und implementiert. Eine den Anforderungen der Anwendung entsprechende Oberfläche wird von den SuS mit Hilfe eines JApplets selbst erstellt. Die Klasse Queue wird dabei von der Lehrkraft vorgegeben. 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: ca. 28 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien, Methoden
1. Die Datenstruktur Schlange im Anwendungskontext unter Nutzung der Klasse Queue (a) Analyse der Problemstellung und Herausarbeiten der Grenzen von statischen Datenstrukturen (Array) (b) FIFO-Prinzip & Verkettung von Objekten (c) Abiturklasse Queue und ihre Methoden (d) Modellierung und Implementierung einer Anwendung unter Verwendung eines oder mehrere Objekte der Klasse Queue	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	*Beispiel*: Wartende Helden (Grenzen von Arrays) *Beispiel*: Wäscheleine (generische Queue) *Beispiel*: Wartezimmer (Queue-Applet)
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 (LIFO-Prinzip) (c) Modellierung und Implementierung einer Anwendung unter Verwendung eines oder mehrere Objekte der Klasse Stack		*Beispiele: BlackJack, Informatik-Biber-Aufgabe (Teller), Palindrom-Test*
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		*Beispiele: Vokabel-Trainer, Listen-Applet (Studienseminar Jülich)*

(b) Modellierung und Implementierung einer kontextbezogenen Anwendung unter Verwendung der Klasse List.	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)	
4. Vertiefung – Anwendung von Listen, Stapeln und Schlangen in mindestens einem weiteren Kontext		*Beispiele: Bowling-Bahn, Autovermietung*

Unterrichtsvorhaben Q1-III:

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.

Die Abiturklassen *DatabaseConnector* und *QueryResult* werden analysiert und verwendet, um eine Datenbank in ein Java-Programm einzubinden und Anfragen stellen und auswerten zu können.

Zeitbedarf: 26 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 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),	*Beispiele:* Surfschule, Online-Buchhandel *Tools:* XAMPP
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 Datenbankschemas 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)	• überprüfen Datenbankschemata auf vorgegebene Normalisierungseigenschaften (D).	
3. Implementierung einer Datenbank a) Einbindung einer Datenbank in ein Java-Programm • Verbindungsherstellung zur Datenbank • Anfragestellung • Auswertung der Ergebnisrelationen		Abiturklassen *DatabaseConnector* und *QueryResult*

Unterrichtsvorhaben Q1-IV:

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: 16 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) 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).	
2. Fallbeispiele zur Datenschutzproblematik und zum Urheberrecht		*Beispiele:* Soziale Netzwerke, Big Brother Awards, Gläserner Mensch

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:
Zu Beginn des Unterrichtsvorhabens werden rekursive Algorithmen eingeführt und anhand verschiedener Beispiele vertieft. 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.
Anhand von Beispielen für Baumstrukturen werden grundlegende Begriffe eingeführt und der rekursive Aufbau binärer Bäume dargestellt.
Anschließend werden für eine Problemstellung in einem Anwendungskontext Klassen modelliert und implementiert. Dabei werden die Operationen der Datenstruktur Binärbaum thematisiert und die entsprechende Klasse *BinaryTree* der Vorgaben für das Zentralabitur NRW verwendet. Die Funktionsweise von Methoden wird anhand grafischer Darstellungen von Binärbäumen erläutert.

Unter anderem sollen sich die S'uS mit den verschiedenen Baumtraversierungen (Pre-, Post- und Inorder) auseinander setzen.

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 Vorgaben für das Zentralabitur NRW weitere Klassen oder Methoden in einem Anwendungskontext modelliert und/oder implementiert. Auch in diesem Kontext werden grafische Darstellungen der Bäume verwendet.

Zeitbedarf: 16 Stunden

Sequenzierung des Unterrichtsvorhabens:

<table>
<tr><th>Unterrichtssequenzen</th><th>Zu entwickelnde Kompetenzen</th><th>Beispiele, Medien, Materialien</th></tr>
<tr><td>1. Einführung in die Rekursion
(a) Abgrenzung von rekursiven zu iterativen Algorithmen
(b) Grundlegende Begriffe (Rekursionstiefe, Rekursionsanker)
(c) Entwicklung und grafische Veranschaulichung eines rekursiven Sortierverfahrens</td><td rowspan="4">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“ und „Teilen und Herrschen“ (M),
• implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I),</td><td>Beispiele: ggT, Türme von Hanoi, Quicksort</td></tr>
<tr><td>2. Analyse von Baumstrukturen in verschiedenen Kontexten
(a) Grundlegende Begriffe (Grad, Tiefe, Höhe, Wurzel, Knoten, Blatt, Inhalt, Teilbaum, Ebene)
(b) Aufbau und Darstellung von binären Bäumen anhand von Baumstrukturen in verschiedenen Kontexten</td><td></td></tr>
<tr><td>3. Die Datenstruktur Binärbaum im Anwendungskontext unter Nutzung der Klasse BinaryTree
(a) Analyse der Problemstellung
(b) Erarbeitung der Klasse BinaryTree und beispielhafte Anwendung der Operationen
(c) Implementierung der Anwendung oder von Teilen der Anwendung
(d) Traversierung eines Binärbaums im Pre-, In- und Postorderdurchlauf</td><td>Beispiele: Akinator, Tiereraten, Muppet- und Simpsonsbaum</td></tr>
<tr><td></td><td>Beispiel: Muppet-Baum</td></tr>
</table>

4. Die Datenstruktur binärer Suchbaum im Anwendungskontext unter Verwendung der Klasse *BinarySearchTree* (a) Analyse der Problemstellung (b) Grafische Darstellung eines binären Suchbaums und Erarbeitung der Struktureigenschaften (c) Erarbeitung der Klasse *BinarySearchTree* und Realisierung einer geeigneten Ordnungsrelation (d) Implementierung einer Anwendung oder von Teilen einer Anwendung	• implementieren und erläutern iterative und rekursive Such- und Sortierverfahren (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).	

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?*

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.

Anhand kontextbezogener Beispiele werden Grammatiken regulärer Sprachen entwickelt, untersucht und modifiziert. Der Zusammenhang zwischen regulären Grammatiken und endlichen Automaten wird verdeutlicht.

An einem Beispiel werden die Grenzen endlicher Automaten ausgelotet.

Zeitbedarf: 18 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien oder Materialien
1. Endliche Automaten (a) Vom Vorwissen zu Automaten aus der Realwelt zur formalen Beschreibung eines endlichen Automaten (b) Untersuchung, Darstellung, Entwicklung und Modifikation endlicher Automaten (c) Entwicklung regulärer Grammatiken (d) Grenzen endlicher Automaten	Die Schülerinnen und Schüler • analysieren und erläutern die Eigenschaften endlicher Automaten einschließlich ihres Verhaltens auf bestimmte Eingaben (A), • entwickeln und modifizieren zu einer Problemstellung endliche Automaten (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), • entwickeln zur akzeptierten Sprache eines Automaten die zugehörige Grammatik (M), • entwickeln zu einer regulären Sprache eine Grammatik, die die Sprache erzeugt (M), • analysieren und erläutern Grammatiken regulärer Sprachen (A), • modifizieren Grammatiken regulärer Sprachen (M), • ermitteln die formale Sprache, die durch eine Grammatik erzeugt wird (A), • entwickeln zur Grammatik einer regulären Sprache einen zugehörigen endlichen Automaten (M), • beschreiben an Beispielen den Zusammenhang zwischen Automaten und Grammatiken (D), • zeigen die Grenzen endlicher Automaten und regulärer Grammatiken im Anwendungszusammenhang auf (A).	*Beispiele:* Cola-Automat o. ä., Akzeptor für bestimmte Eingaben *Beispiel:* Reguläre Sprache zur Darstellung von Termen *Beispiel:* Klammerautomat

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:* Einfache Addition 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		*Beispiele:* Halteproblem, Türme von Hanoi

<u>Unterrichtsvorhaben Q2-IV:</u>

Thema: Wiederholung und Vertiefung ausgewählter Kompetenzen und Inhalte der Qualifikationsphase

Zeitbedarf: 10 Stunden

II) Qualifikationsphase – LEISTUNGSKURS

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 unter Hinzunahme zweidimensionaler Arrays und optionaler GUI-Gestaltung

Leitfragen: *Wie modelliert man zu einer Problemstellung in einem geeigneten Anwendungskontext Java-Klassen inklusive ihrer Attribute, Methoden und Beziehungen? Wie lassen sich statische Datenstrukturen für den gewählten Anwendungskontext nutzen? Wie kann man die Modellierung und die Funktionsweise der Anwendung grafisch darstellen?*

Vorhabenbezogene Konkretisierung:
Zu einer Problemstellung in einem Anwendungskontext soll zunächst ein Entwurfsdiagramm entwickelt werden. Die Schülerinnen und Schülern erläutern und modifizieren den ersten Entwurf und modellieren weitere Klassen und Methoden für eine entsprechende Anwendung. Klassen und ihre Beziehungen werden anschließend in einem Implementationsdiagramm dargestellt. Anhand des Anwendungskontextes werden statische Datenstrukturen in Form von ein- und zweidimensionalen Arrays behandelt. Zudem werden Sichtbarkeitsbereiche zugeordnet. Daraufhin wird das Projekt in Java implementiert und durch die Entwicklung einer GUI erweitert.

Zeitbedarf: ca. 15 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien, Methoden
2. Wiederholung und Erweiterung der objektorientierten Modellierung und Programmierung durch Analyse und Erweiterung eines kontextbezogenen Beispiels (a) Analyse der Problemstellung (b) Analyse der Modellierung (Entwurfs- & Implementationsdiagramm) (c) Implementierung der Anwendung oder von Teilen der Anwendung (d) Entwicklung einer GUI für den Anwendungskontext (e) Präsentation und Diskussion über Modellierungen (unter Verwendung wichtiger Fachbegriffe)	Die Schülerinnen und Schüler • analysieren und erläutern objektorientierte Modellierungen (A) • modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (M) • ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M) • stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D) • dokumentieren Klassen (D) • nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung und gemeinsamen Verwendung von Daten unter Berücksichtigung der Rechteverwaltung (K) • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare Datensammlungen zu (M)	*Beispiel*: Modellierung und Implementierung eines Fußballturniers

Unterrichtsvorhaben Q1-II

Thema: Modellierung und Implementierung von dynamischen, linearen Datenstrukturen und Anwendungen mit dynamischen, linearen Datenstrukturen in kontextbezogenen Problemstellungen, auch unter Berücksichtigung der Gestaltung einer Benutzungsoberfläche sowie Modellierung, Implementierung, Analyse und Beurteilung von Such- und Sortierverfahren unterschiedlicher Komplexitätsklassen in kontextbezogenen Problemstellungen

Leitfragen: *Wie können beliebig viele linear angeordnete Daten im Anwendungskontext verwaltet werden? Wie lassen sich Daten mit Hilfe von Such- und Sortierverfahren sinnvoll strukturieren?*

Vorhabenbezogene Konkretisierung:
Zu Beginn der Unterrichtseinheit werden die Grenzen von statischen Datenstrukturen am Beispiel des Arrays thematisiert und von den Schülern herausgearbeitet. Daraufhin folgt ein Übergang zu dynamischen Datenstrukturen. Nach Analyse einer Problemstellung in einem geeigneten Anwendungskontext, in dem Daten nach dem First-In-First-Out-Prinzip verwaltet werden, wird der Aufbau von Schlangen an einem Beispiel dargestellt und die Operationen der Klasse Queue erläutert. Anschließend werden für die Anwendung notwendige Klassen modelliert und implementiert. Eine den Anforderungen der Anwendung entsprechende Oberfläche wird von den SuS mit Hilfe eines JApplets selbst erstellt. Die Klasse Queue wird dabei von der Lehrkraft vorgegeben. 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. Dabei wird auch eine Listenanwendung implementiert. Anschließend werden Such- und Sortierverfahren auf dynamischen, linearen Datenstrukturen analysiert und implementiert. 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. 40 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien, Methoden
5. Die Datenstruktur Schlange im Anwendungskontext unter Nutzung der Klasse Queue (a) Analyse der Problemstellung und Herausarbeiten der Grenzen von statischen Datenstrukturen (Array) (b) FIFO-Prinzip & Verkettung von Objekten (c) Abiturklasse Queue und ihre Methoden (d) Modellierung und Implementierung einer Anwendung unter Verwendung eines oder mehrere Objekte der Klasse Queue	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 von Programmen (I)	*Beispiel*: Wartende Helden (Grenzen von Arrays) *Beispiel*: Wäscheleine (generische Queue) *Beispiel*: Wartezimmer (Queue-Applet)
6. 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 (LIFO-Prinzip) (c) Modellierung und Implementierung einer Anwendung unter Verwendung eines oder mehrere Objekte der Klasse Stack		*Beispiele: BlackJack, Informatik-Biber-Aufgabe (Teller), Palindrom-Test*
7. 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		*Beispiele: Vokabel-Trainer, ToDo Liste*

(b) Modellierung und Implementierung einer kontextbezogenen Anwendung unter Verwendung der Klasse List.	• interpretieren Fehlermeldungen und korrigieren den Quellcode (I) • testen Programme systematisch anhand von Beispielen und mit Hilfe von Testanwendungen (I) • stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D)	
8. Such- und Sortierverfahren (a) Erarbeitung und Implementierung von Suchverfahren (b) Erarbeitung von Sortierverfahren (c) Laufzeitanalyse von Such- und Sortierverfahren		*Beispiele: Lineare und Binäre Suche, Selection-, Insertion-, Bubble-, Quicksort*
9. Vertiefung – Anwendung von Listen, Stapeln und Schlangen in mindestens einem weiteren Kontext		*Beispiele: Bowling-Bahn, Autovermietung*

Unterrichtsvorhaben Q1-III:

Thema: Modellierung, Implementierung und Nutzung von relationalen Datenbanken in Anwendungskontexten sowie projektorientierte Softwareentwicklung am Beispiel einer Anwendung mit Datenbankanbindung

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.

Die Abiturklassen *DatabaseConnector* und *QueryResult* werden analysiert und verwendet, um eine Datenbank in ein Java-Programm einzubinden und Anfragen stellen und auswerten zu können. Als Abschluss wird ein Quizspiel inklusive Benutzeranmeldung modelliert und implementiert.

Zeitbedarf: 40 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 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),	*Beispiele:* Surfschule, Online-Buchhandel, Social Media *Tools:* XAMPP, Instahub
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 Datenbankschemata 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)	• überprüfen Datenbankschemata auf vorgegebene Normalisierungseigenschaften (D).	
3. Implementierung einer Datenbank (a) Einbindung einer Datenbank in ein Java-Programm • Verbindungsherstellung zur Datenbank • Anfragestellung • Auswertung der Ergebnisrelationen		*Beispiel*: Quizspiel *Tools:* XAMPP Abiturklassen *DatabaseConnector* und *QueryResult*

Unterrichtsvorhaben Q1-IV:

Thema: Modellierung und Implementierung von dynamischen nicht-linearen Datenstrukturen und von Anwendungen mit dynamischen nicht-linearen Datenstrukturen in kontextbezogenen Problemstellungen

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? Wie werden Graphen in informatischen Kontexten verwendet? Welche Möglichkeiten gibt es Graphen zu durchlaufen? Wie können Graphen implementiert werden?*

Vorhabenbezogene Konkretisierung:
Zu Beginn des Unterrichtsvorhabens werden rekursive Algorithmen eingeführt und anhand verschiedener Beispiele vertieft. 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.
Anhand von Beispielen für Baumstrukturen werden grundlegende Begriffe eingeführt und der rekursive Aufbau binärer Bäume dargestellt.
Anschließend werden für eine Problemstellung in einem Anwendungskontext Klassen modelliert und implementiert. Dabei werden die Operationen der Datenstruktur Binärbaum thematisiert und die entsprechende Klasse *BinaryTree* der Vorgaben für das Zentralabitur NRW verwendet. Die Funktionsweise von Methoden wird anhand grafischer Darstellungen von Binärbäumen erläutert.

Unter anderem sollen sich die S'uS mit den verschiedenen Baumtraversierungen (Pre-, Post- und Inorder) auseinandersetzen.

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 Vorgaben für das Zentralabitur NRW weitere Klassen oder Methoden in einem Anwendungskontext modelliert und implementiert. Auch in diesem Kontext werden grafische Darstellungen der Bäume verwendet.

Als weitere nicht lineare Datenstruktur werden anschließend gerichtete und ungerichtete Graphen erarbeitet. Anhand eines Anwendungskontextes werden die einzelnen Elemente eines Graphen eingeführt. Dabei werden klassische Problemstellungen innerhalb der Graphentheorie betrachtet. Adjazenzmatrizen werden mit Hilfe von zweidimensionalen Arrays implementiert.

In einem entsprechenden anwendungsbezogenen Projekt werden die Abiturklassen *Graph*, *Vertex* und *Edge* verwendet. Die S'uS sollen sich mit der der Breiten- und Tiefensuche auseinandersetzen und diese mit Hilfe rekursiver Algorithmen implementieren.

Zeitbedarf: 40 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Einführung in die Rekursion (a) Abgrenzung von rekursiven zu iterativen Algorithmen (b) Grundlegende Begriffe (Rekursionstiefe, Rekursionsanker) (c) Entwicklung und grafische Veranschaulichung eines rekursiven Sortierverfahrens	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), • beurteilen die Effizienz von Algorithmen unter Berücksichtigung des Speicherbedarfs und der Zahl der Operationen (A), • ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare und nichtlineare Datensammlungen zu (M), • modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M), • verwenden bei der Modellierung geeigneter Problemstellungen die Möglichkeiten der Polymorphie (M), • entwickeln iterative und rekursive Algorithmen unter Nutzung der Konstruktionsstrategien „Modularisierung“ und „Teilen und Herrschen“ und „Backtracking“(M), • entwickeln mit didaktisch orientierten Entwicklungsumgebungen einfache	*Beispiele:* ggT, Türme von Hanoi, Quicksort
2. Analyse von Baumstrukturen in verschiedenen Kontexten (c) Grundlegende Begriffe (Grad, Tiefe, Höhe, Wurzel, Knoten, Blatt, Inhalt, Teilbaum, Ebene) (d) Aufbau und Darstellung von binären Bäumen anhand von Baumstrukturen in verschiedenen Kontexten		
3. Die Datenstruktur Binärbaum im Anwendungskontext unter Nutzung der Klasse *BinaryTree* (a) Analyse der Problemstellung (b) Erarbeitung der Klasse *BinaryTree* und beispielhafte Anwendung der Operationen (c) Implementierung der Anwendung oder von Teilen der Anwendung (d) Traversierung eines Binärbaums im Pre-, In- und Postorderdurchlauf		*Beispiele:* Akinator, Tiereraten, Muppet- und Simpsonsbaum
4. Die Datenstruktur binärer Suchbaum im Anwendungskontext unter Verwendung der Klasse *BinarySearchTree*		*Beispiel:* Muppet-Baum

(a) Analyse der Problemstellung (b) Grafische Darstellung eines binären Suchbaums und Erarbeitung der Struktureigenschaften (c) Erarbeitung der Klasse *BinarySearchTree* und Realisierung einer geeigneten Ordnungsrelation (d) Implementierung einer Anwendung oder von Teilen einer Anwendung	Benutzungsoberflächen zur Kommunikation mit einem Informatiksystem (M), • implementieren Operationen dynamischer (linearer oder nichtlinearer) Datenstrukturen (I), • implementieren und erläutern iterative und rekursive Such- und Sortierverfahren unterschiedlicher Komplexitätsklassen (Speicherbedarf und Laufzeitverhalten) (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • testen Programme systematisch anhand von Beispielen und mithilfe von Testanwendungen(I), • wenden didaktisch orientierte Entwicklungsumgebungen zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I), • stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D), • stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D), • nutzen bereitgestellte Informatiksysteme und das Internet reflektiert zur Erschließung, Aufbereitung und Präsentation fachlicher Inhalte (D), • nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung von Daten, zur Organisation von Arbeitsabläufen sowie zur Verteilung und Zusammenführung von Arbeitsanteilen (K).	
5. Analyse von Graphen in verschiedenen Kontexten (a) Grundlegende Begriffe (Graph, gerichtet – ungerichtet, Knoten, Kanten, Kantengewicht) (b) Aufbau und Darstellung von Graphen anhand von Graphenstrukturen in verschiedenen Kontexten (Adjazenzmatrix, Adjazenzliste)		*Beispiele:* Wegsuche und Eulerkreis
6. Die Datenstruktur Graph im Anwendungskontext unter Nutzung der Klassen Graph, Vertex und Edge. (a) Erarbeitung der Klassen Graph, Vertex und Edge und beispielhafte Anwendung der Operationen (b) Bestimmung von Wegen in Graphen im Anwendungskontext (Tiefensuche, Breitensuche) (c) Bestimmung von kürzesten Wegen in Graphen im Anwendungskontext (Backtracking, Dijkstra). (d) Bestimmung von minimalen Spannbäumen eines Graphen im Anwendungskontext.		*Beispiel: Dijkstra, Soziale Netzwerke*

Unterrichtsvorhaben Q2-I:

Thema: Sicherheit und Datenschutz in Informatiksystemen sowie Grenzen und Auswirkungen der Automatisierung

Leitfragen: *Welche moralische und rechtliche Verantwortung tragen Informatikerinnen und Informatiker hinsichtlich des Datenschutzes, des Urheberrechts und der gesellschaftlichen Auswirkungen informatischer Systeme? Wie kann Datensicherheit z. B. mit Hilfe von Verschlüsselungsverfahren gewährleistet werden? Wo liegen die Grenzen der Automatisierbarkeit?*

Vorhabenbezogene Konkretisierung:
Anhand eines Fallbeispiels wird der Begriffe der personenbezogenen Daten, informationeller Selbstbestimmung, Datensparsamkeit usw. eingeführt. Anschließend wird die moralische Notwendigkeit betrachtet personenbezogene Daten zu schützen. In diesem Zusammenhang werden rechtliche Grundlagen bezüglich des Datenschutzes erarbeitet.
Ein weiteres Beispiel dient der Einführung der Begriffe der Urheberschaft und des Urheberrechts.

Anschließend werden symmetrische und asymmetrische Verschlüsselungsverfahren zur Umsetzung von Sicherheitszielen eingeführt. Dabei werden die grundlegenden Verfahren dieser Verschlüsselungen und die Möglichkeiten eines Angriffs auf sie in den Mittelpunkt gestellt. Im Kon-text der symmetrischen Verschlüsselungsverfahren sollte das Problem des Schlüsselaustausches angesprochen werden. Im Anschluss an asymmetrische Verschlüsselungsverfahren wird das Prinzip des Signierens thematisiert.

Zeitbedarf: 20 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Fallbeispiele zur Datenschutzproblematik und zum Urheberrecht	Die Schülerinnen und Schüler • 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).	*Beispiele:* Soziale Netzwerke, Big Brother Awards, Gläserner Mensch, Arztpraxis, Big Data, Creative-Commons-Lizenzen
2. Möglichkeiten zum Schutz von personenbezogenen Daten (a) Symmetrische und asymmetrische kryptografische Verfahren (Cäsar-, Vigenère-, RSA-Verfahren) als Methoden, Daten im Netz verschlüsselt zu übertragen		

Unterrichtsvorhaben Q2-II:

Thema: Grundlagen von Automaten und formalen Sprachen sowie die Modellierung und Implementierung eines Parsers zu einer formalen Sprache

Leitfragen: *Wie kann man Automaten genau beschreiben? Wie können 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? Wie kann ein Parser für eine einfache formale Sprache entwickelt werden?*

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 als Alternative zu einem entsprechenden deterministischen Akzeptor eingeführt. Auch die Umwandlung eines nichtdeterministischen Automaten in einen deterministischen Automaten wird thematisiert.

Anhand kontextbezogener Beispiele werden Grammatiken regulärer Sprachen entwickelt, untersucht und modifiziert. Der 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. Zu einer einfachen regulären Sprache wird ein Parser in Form eines Java-Programms entwickelt.

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

Zeitbedarf: 30 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien oder Materialien
1. Endliche Automaten (a) Erarbeitung der formalen Beschreibung eines endlichen Automaten auf der Grundlage von Automaten in bekannten Kontexten (b) Untersuchung, Darstellung und Entwicklung endlicher Automaten (c) Umwandlung nichtdeterministischer endlicher Automaten in deterministische endliche 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), • erläutern die Grenzen endlicher Automaten und regulärer Sprachen im Anwendungszusammenhang (A), • ermitteln die formale Sprache, die durch eine Grammatik erzeugt wird (A), • entwickeln und modifizieren zu einer Problemstellung endliche Automaten oder Kellerautomaten (M), • entwickeln zur akzeptierten Sprache eines Automaten die zugehörige Grammatik (M), • modellieren und implementieren Scanner, Parser und Interpreter zu einer gegebenen regulären Sprache (I), • entwickeln zur Grammatik einer regulären oder kontextfreien Sprache einen zugehörigen endlichen Automaten oder einen Kellerautomaten (M), • modifizieren Grammatiken regulärer und kontextfreier Sprachen (M), • entwickeln zu einer regulären oder kontextfreien Sprache eine Grammatik, die die Sprache erzeugt (M),	*Beispiele:* Cola-Automat o. ä., Akzeptor für bestimmte Eingaben
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 (e) Entwicklung eines Parsers für eine einfache reguläre Sprache		*Beispiel:* Reguläre Sprache zur Darstellung von Termen
3. Grenzen endlicher Automaten		*Beispiel:* Klammerautomat

4. Entwicklung eines Kellerautomaten als Antwort auf die Grenzen endlicher Automaten (a) Erweiterung eines DEA um eine einzelne Speichervariable zum Zählen von Eingabezeichen (z.B. Klammern) und Problematisierung dieses Ansatzes (b) Entwicklung eines Automaten mit Kellerspeicher (c) Anwendung eines Kellerautomaten zur Syntaxüberprüfung auf Grundlage von nicht-regulären Grammatiken (d) Implementierung eines Kellerautomaten zur Syntaxüberprüfung (Backtracking)	• 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).	*Beispiel:* Klammerautomat, Palindromtest

Unterrichtsvorhaben Q2-III:

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

Leitfragen: *Wie werden Daten in Netzen übermittelt? Wie entwickelt man ein Client-Server-System im Anwendungskontext? Welche Algorithmen sind zu implementieren?*

Vorhabenbezogene Konkretisierung:
Zunächst werden die Grundlagen von Datenübertragung in Netzwerken erarbeitet. Ein Schichtenmodell als Strukturierungsprinzip für Netzwerkkommunikation wird eingeführt. Im Anschluss werden von den Schülerinnen und Schülern Antworten auf grundsätzliche Herausforderungen im Bereich Netzwerkkommunikation erarbeitet: Vor- und Nachteile verschiedener Topologien, Adressierung/Routing in IP-Netzen sowie die Gestaltung von Protokollen für die Anwendungsebene.

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

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

Zeitbedarf: 25 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien oder Materialien
1. Grundlagen der Datenübertragung in Netzwerken (a) Schichtenmodell • Erarbeitung eines standardisierten Schichtenmodells für Netzwerkkommunikation (b) Topologien • Erarbeitung und Vergleich ausgewählter Netzwerktopologien (c) Routing • Analyse von Grundlagen der Adressierung in IP-Netzwerken (d) Protokolle • Erarbeitung des beispielhaften Aufbaus eines Protokolls auf der Anwendungsschicht	Die Schülerinnen und Schüler • beschreiben und erläutern Netzwerk-Topologien, die Client-Server-Struktur und Protokolle sowie ein Schichtenmodell in Netzwerken (A), • analysieren und erläutern Algorithmen und Programme (A), • analysieren und erläutern Protokolle zur Kommunikation in einem Client-Server-Netzwerk (A), • entwickeln und implementieren Algorithmen und Methoden zur Client-Server-Kommunikation (I), • entwickeln und erweitern Protokolle zur Kommunikation in einem Client-Server-Netzwerk (M), • analysieren und erläutern Algorithmen und Methoden zur Client-Server-Kommunikation (A), • modifizieren Algorithmen und Programme (I) • ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihr Operationen und Beziehungen (M), • modellieren Klassen mit ihren Attributen, Methoden und Assoziationsbeziehungen unter Angabe von Multiplizitäten (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen,	
2. Analyse, Modellierung und Implementierung von Netzwerkanwendungen in Client-Server-Struktur (a) Nutzung einfacher Server-Dienste mittels Client • Modellierung und Implementierung von Clients für einfache Serverdienste (b) Anbieten von Diensten mittels Server • Analyse vorgegebener Implementationen einfacher Server		*Beispiel:* Echo- bzw. Daytime-Clients und –Server

3. Entwicklung eines vollständigen Client-Server-Systems • Protokollentwurf, Dialogorientierung • Modellierung mittels Entwurfs- und Implementationsdiagramm • Bedeutung von Nebenläufigkeit • Implementierung	Objekttypen oder lineare und nichtlineare Datensammlungen zu (M), • analysieren und erläutern objektorientierte Modellierungen (A), • erläutern das Prinzip der Nebenläufigkeit (A), • stellen Klassen und ihre Beziehungen grafisch dar (D), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I).	*Beispiel*: Messenger-Dienst

Unterrichtsvorhaben Q2-IV:

Thema: Prinzipielle Arbeitsweise eines Computers sowie eines Scanners, Parsers und Interpreters für eine einfache maschinennahe Programmiersprache

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. Grundlegenden Begrifflichkeiten bei der maschinellen Übersetzung von einer Hochsprache in eine maschinenverständliche Sprache werden definiert, veranschaulicht und zum Vorwissen aus dem Unterrichtsvorhaben Q2-III in Beziehung gesetzt. Dabei werden die Bestandteile eines Compilers wiederholt.

Mit Hilfe eines Simulators wird maschinennahe Programmierung verwendet.

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: 15 Stunden

Sequenzierung des Unterrichtsvorhabens:

<table>
<tr><th>Unterrichtssequenzen</th><th>Zu entwickelnde Kompetenzen</th><th>Beispiele, Medien oder Materialien</th></tr>
<tr><td>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</td><td rowspan="2">Die Schülerinnen und Schüler

• erläutern die Ausführung eines einfachen maschinennahen Programms sowie die Datenspeicherung auf einer „Von-Neumann-Architektur“ (A),
• testen Programme systematisch anhand von Beispielen und mit Hilfe von Testanwendungen (I),
• untersuchen und beurteilen Grenzen des Problemlösens mit Informatiksystemen (A).</td><td>Beispiel: Einfache Addition mit einem Rechnermodell, Assembler</td></tr>
<tr><td>2. Simulation der Phasen eines Compilers
(a) Prozesse beim Compiler:
• scannen
• parsen
• übersetzen/interpretieren
(b) Arten von Fehlern:
• lexikalischer Fehler
• syntaktischer Fehler
• semantischer Fehler
(c) Einordnung der neuen Begriffe in den Gesamtkontext der formalen Sprachen
• Automaten
• Grammatiken
• Sprachen</td><td>Beispiele: Scanner, Parser und Interpreter für geometrische Figuren, Halteproblem</td></tr>
</table>

Unterrichtsvorhaben Q2-V:

Thema: Wiederholung und Vertiefung ausgewählter Kompetenzen und Inhalte der Qualifikationsphase

Zeitbedarf: 10 Stunden

2.2 Grundsätze der fachmethodischen und fachdidaktischen Arbeit

In Absprache mit der Lehrerkonferenz sowie unter Berücksichtigung des Schulprogramms hat die Fachkonferenz Informatik des Immanuel-Kant-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 und Arbeit im Plenum.
11) Die Lernumgebung ist vorbereitet; der Ordnungsrahmen wird eingehalten.
12) Die Lehr- und Lernzeit wird intensiv für Unterrichtszwecke genutzt.
13) Es herrscht ein positives pädagogisches Klima im Unterricht.

Fachliche Grundsätze:

14) Der Unterricht unterliegt der Wissenschaftsorientierung und ist dementsprechend eng verzahnt mit seiner Bezugswissenschaft.
15) Der Unterricht ist problemorientiert.
16) Der Unterricht folgt dem Prinzip des Exemplarischen und soll ermöglichen, informatische Strukturen und Gesetzmäßigkeiten in den ausgewählten Problemen und Projekten zu erkennen.
17) Der Unterricht ist anschaulich sowie gegenwarts- und zukunftsorientiert.
18) Der Unterricht ist handlungsorientiert, d.h. projekt- und produktorientiert angelegt.
19) Im Unterricht werden sowohl für die Schule didaktisch reduzierte als auch reale Informatiksysteme aus der Wissenschafts-, Berufs- und Lebenswelt eingesetzt.
20) Der Unterricht beinhaltet reale Begegnung mit Informatiksystemen.

2.3 Grundsätze der Leistungsbewertung und Leistungsrückmeldung

Auf der Grundlage von §13 - §16 der APO-GOSt sowie Kapitel 3 des Kernlehrplans Informatik für die gymnasiale Oberstufe hat die Fachkonferenz des Immanuel-Kant-Gymnasiums im Einklang mit dem entsprechenden schulbezogenen Konzept die nachfolgenden Grundsätze zur Leistungsbewertung und Leistungsrückmeldung beschlossen. Die nachfolgenden Absprachen stellen die Minimalanforderungen an das lerngruppenübergreifende gemeinsame Handeln der Fachgruppenmitglieder dar. Bezogen auf die einzelne Lerngruppe kommen ergänzend weitere der in den Folgeabschnitten genannten Instrumente der Leistungsüberprüfung zum Einsatz.

2.3.1 Beurteilungsbereich Klausuren

Verbindliche Absprachen:
Bei der Formulierung von Aufgaben werden die für die Abiturprüfungen geltenden Operatoren des Faches Informatik schrittweise eingeführt, erläutert und dann im Rahmen der Aufgabenstellungen für die Klausuren benutzt.

Instrumente:

- Einführungsphase: 1 Klausur je Halbjahr
  Dauer der Klausur: 2 Unterrichtsstunden
- Grundkurse Q 1: 2 Klausuren je Halbjahr
  Dauer der Klausuren: 2-3 Unterrichtsstunden
- Grundkurse Q 2.1: 2 Klausuren
  Dauer der Klausuren: 3 Unterrichtsstunden
- Grundkurse Q 2.2: 1 Klausur unter Abiturbedingungen
- Anstelle einer Klausur kann gemäß dem Beschluss der Lehrerkonferenz in Q 1.2 eine Facharbeit geschrieben werden.

Die Aufgabentypen, sowie die Anforderungsbereiche I-III sind entsprechend den Vorgaben in Kapitel 3 des Kernlehrplans zu beachten.

Kriterien

Die Bewertung der schriftlichen Leistungen in Klausuren erfolgt über ein Raster mit Hilfspunkten, die im Erwartungshorizont den einzelnen Kriterien zugeordnet sind.

Von diesem kann aber im Einzelfall begründet abgewichen werden, wenn sich z.B. besonders originelle Teillösungen nicht durch Hilfspunkte gemäß den Kriterien des Erwartungshorizontes abbilden lassen oder eine Abwertung wegen besonders schwacher Darstellung (APO-GOSt §13 (2)) angemessen erscheint.

Die Note ausreichend (5 Punkte) soll bei Erreichen von 45 % der Hilfspunkte erteilt werden.

2.3.2 Beurteilungsbereich Sonstige Mitarbeit

Den Schülerinnen und Schülern werden die Kriterien zum Beurteilungsbereich „sonstige Mitarbeit“ zu Beginn des Schuljahres genannt.

Verbindliche Absprachen der Fachkonferenz

- Der Schwerpunkt des Unterrichts wird auf die objektorientierte Programmierung in Java gelegt.

Leistungsaspekte

Mündliche Leistungen
- Beteiligung am Unterrichtsgespräch
- Zusammenfassungen zur Vor- und Nachbereitung des Unterrichts
- Präsentation von Arbeitsergebnissen
- Referate
- Mitarbeit in allen Arbeitsphasen

Praktische Leistungen am Computer
- Implementierung, Test und Anwendung von Informatiksystemen

Sonstige schriftliche Leistungen
- Lernerfolgsüberprüfung durch kurze schriftliche Übungen
- Schriftliche Übung dauern ca. 20 Minuten und umfassen den Stoff der letzten ca. 4–6 Stunden.
- Bearbeitung von schriftlichen Aufgaben im Unterricht

Kriterien

Die folgenden allgemeinen Kriterien gelten sowohl für die mündlichen als auch für die schriftlichen Formen der sonstigen Mitarbeit.

Die Bewertungskriterien stützen sich auf
- die Qualität der Beiträge,
- die Quantität der Beiträge und
- die Kontinuität der Beiträge.

Besonderes Augenmerk ist dabei auf
- die sachliche Richtigkeit,
- die angemessene Verwendung der Fachsprache,
- die Darstellungskompetenz,
- die Komplexität und den Grad der Abstraktion,
- die Selbstständigkeit im Arbeitsprozess,
- die Präzision und
- die Differenziertheit der Reflexion zu legen.

Bei Gruppenarbeiten auch auf

- das Einbringen in die Arbeit der Gruppe,
- die Durchführung fachlicher Arbeitsanteile und
- die Qualität des entwickelten Produktes.

Grundsätze der Leistungsrückmeldung und Beratung

Die Grundsätze der Leistungsbewertung werden zu Beginn eines jeden Halbjahres den Schülerinnen und Schülern transparent gemacht. Leistungsrückmeldungen können erfolgen

- nach einer mündlichen Überprüfung,
- bei Rückgabe von schriftlichen Leistungsüberprüfungen,
- nach Abschluss eines Projektes,
- nach einem Vortrag oder einer Präsentation,
- bei auffälligen Leistungsveränderungen,
- auf Anfrage,
- als Quartalsfeedback und
- zu Eltern- oder Schülersprechtagen.

3 Entscheidungen zu fach- und unterrichtsübergreifenden Fragen

Die Fachkonferenz Informatik hat sich im Rahmen des Schulprogramms für folgende zentrale Schwerpunkte entschieden:

Zusammenarbeit mit anderen Fächern

Im Informatikunterricht werden Kompetenzen anhand informatischer Inhalte in verschiedenen Anwendungskontexten erworben, in denen Schülerinnen und Schülern aus anderen Fächern Kenntnisse mitbringen können. Diese können insbesondere bei der Auswahl und Bearbeitung von Softwareprojekten berücksichtigt werden und in einem hinsichtlich der informatischen Problemstellung angemessenem Maß in den Unterricht Eingang finden.

Vorbereitung auf die Erstellung der Facharbeit

Möglichst schon im zweiten Halbjahr der Einführungsphase, spätestens jedoch im ersten Halbjahr des ersten Jahres der Qualifikationsphase werden im Unterricht an geeigneten Stellen Hinweise zur Erstellung von Facharbeiten gegeben. Das betrifft u. a. Themenvorschläge, Hinweise zu den Anforderungen und zur Bewertung.

4 Qualitätssicherung und Evaluation

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. Regelmäßig werden in den Sitzungen der Fachkonferenz Erfahrungen ausgetauscht und ggf. Änderungen 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.