STÄDT. MATHEMATISCH-NATURWISSENSCHAFTLICHES GYMNASIUM
MÖNCHENGLADBACH
Sekundarstufen I und II • Sekundarstufe I mit Ganztagsangebot
Rheydter Str. 65 • 41065 Mönchengladbach • Tel. (02161)92891-00 • FAX 92891-29

Schulinterner Lehrplan
zum Kernlehrplan für die gymnasiale Oberstufe

Mathematisch-Naturwissenschaftliches Gymnasium Mönchengladbach

Informatik

18.02.2015

Inhalt

1 DIE FACHGRUPPE DES MATH.-.NAT. GYMNASIUM MÖNCHENGLADBACH 3

2 ENTSCHEIDUNGEN ZUM UNTERRICHT 4

2.1 UNTERRICHTSVORHABEN 4

2.1.1 Übersicht über die Unterrichtsvorhaben 5

2.1.2 Konkretisierte Unterrichtsvorhaben 13

2.2 GRUNDSÄTZE DER FACHMETHODISCHEN UND FACHDIDAKTISCHEN ARBEIT 46

2.3 GRUNDSÄTZE DER LEISTUNGSBEWERTUNG UND LEISTUNGSRÜCKMELDUNG 47

2.4 LEHR- UND LERNMITTEL 47

3 ENTSCHEIDUNGEN ZU FACH- UND UNTERRICHTSÜBERGREIFENDEN FRAGEN 48

4 QUALITÄTSSICHERUNG UND EVALUATION 49

1 Die Fachgruppe des Math.-.Nat. Gymnasium Mönchengladbach

Beim Math.-Nat. Gymnasium handelt es sich um eine vierzügige Schule im Zentrum von Mönchengladbach. Das Einzugsgebiet der Schule umfasst einen Teil der Mönchengladbacher Innenstadt sowie umliegender Stadtteile.
Das Fach Informatik wird am Math.-Nat. Gymnasium ab der Jahrgangsstufe 8 im Wahlpflichtbereich II (WP II) vierstündig unterrichtet. Fast alle Schülerinnen und Schüler, die in der Oberstufe Informatik wählen haben den Kurs im WP II besucht. In der zweijährigen Laufzeit dieser Kurse wird in altersstufengerechter Weise unter anderem auf Grundlagen der Algorithmik am Beispiel einer didaktischen Lernumgebung, auf die technische Informatik am Beispiel von Schaltwerken und Schaltnetzen und auf Robotik eingegangen.

Im Rahmen des Kurses im WP II wird auch eine Einführung in das Arbeiten mit Office gegeben, auch wenn dieser Teil inhaltlich nicht unmittelbar dem Bereich der Informatik zuzuordnen ist.

In der Sekundarstufe II bietet das Math.-Nat. Gymnasium für die Schülerinnen und Schüler in allen Jahrgangsstufen einen Grundkurs in Informatik an. Einen Leistungskurs gibt es am Math.-Nat. Gymnasium zurzeit nicht.

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. In der Einführungsphase kommt dabei zusätzlich eine didaktische Bibliothek zum Einsatz, welche das Erstellen von grafischen Programmen erleichtert.

Durch projektartiges Vorgehen, offene Aufgaben und Möglichkeiten, Problemlösungen zu verfeinern oder zu optimieren, entspricht der 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 Math.-Nat. Gymnasiums aus fünf Lehrkräften, denen zwei Computerräume mit 14 bzw. 13 Plätzen zur Verfügung stehen. Alle Arbeitsplätze sind an das schulinterne Rechnernetz angeschlossen, so dass Schülerinnen und Schüler über einen individuell gestaltbaren Zugang zum zentralen Server der Schule alle Arbeitsplätze der zwei Räume zum Zugriff auf ihre eigenen Daten, zur Recherche im Internet oder zur Bearbeitung schulischer Aufgaben verwenden können.

Der Unterricht erfolgt im 90-Minuten-Takt.

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 Übersicht über die Unterrichtsvorhaben

I Einführungsphase

Einführungsphase	
Unterrichtsvorhaben E-I **Thema:** *Wirkung der Automatisierung in der Geschichte der Datenverarbeitung, Grundlagen von Programmierung und algorithmischen Grundstrukturen in Java anhand einfacher Rechenoperationen und grundlegender Datentypen* **Zentrale Kompetenzen:** • Implementieren • Darstellen und interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Algorithmen • Formale Sprachen und Automaten • Informatik, Mensch und Gesellschaft **Inhaltliche Schwerpunkte:** • Auswirkungen der Digitalisierung • Syntax und Semantik einer Programmiersprache • Analyse, Entwurf und Implementierung einfacher Algorithmen • Verwendung der grundlegenden Kontrollstrukturen (Verzweigung, Schleifen), Rechen- sowie Logikoperatoren • Betrachtung grundlegender Datentypen in ausgewählten informatischen Kontexten **Zeitbedarf**: 4 Stunden á 90 Minuten	*Unterrichtsvorhaben E-II* **Thema**: *Grundlagen der objektorientierten Programmierung, Modellierung und Implementierung in anwendungsorientierten Kontexten* **Zentrale Kompetenzen:** • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Daten und ihre Strukturierung • Formale Sprachen und Automaten **Inhaltliche Schwerpunkte:** • Objekte und Klassen • Syntax und Semantik einer Programmiersprache **Zeitbedarf**: 5 Stunden á 90 Minuten

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

Thema:
Grundlagen der objektorientierten Analyse, Modellierung und Implementierung graphischer Szenen mit Hilfe einer vorgegebenen Graphikbibliothek

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

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

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

Zeitbedarf: 12 Stunden á 90 Minuten</td>
<td><u>Unterrichtsvorhaben E-IV</u>

Thema:
Grundlagen graphischer Benutzeroberflächen, Ereignissteuerung und das Listenerkonzept in Java an einfachen Beispielen

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

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

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

Zeitbedarf: 5 Stunden á 90 Minuten</td>
</tr>
</table>

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

Thema:
Entwicklung und Implementierung von Such- und Sortieralgorithmen anhand kontextbezogener Beispiele

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

Inhaltsfelder:
• Algorithmen
• Formale Sprachen und Automaten
• Informatiksysteme

Inhaltliche Schwerpunkte:
• Algorithmen zum Suchen und Sortieren
• Analyse, Entwurf und Implementierung einfacher Algorithmen
• Syntax und Semantik einer Programmiersprache
• Digitalisierung

Zeitbedarf: 6 Stunden á 90 Minuten</td>
<td><u>Unterrichtsvorhaben E-VI</u>

Thema:
Der grundlegende Aufbau und die Funktionsweise von Netzstrukturen und deren Auswirkungen auf die Gesellschaft

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

Inhaltsfelder:
• Informatik, Mensch und Gesellschaft
• Informatiksysteme
• Daten und ihre Strukturierung

Inhaltliche Schwerpunkte:
• Aufbau von lokalen Netzwerken
• Adressierung in Netzwerken (lokal, Internet)
• Anwendungen im Netz(werk)
• Sicherheit und Gefahren in Netzwerken
• Auswirkungen der Vernetzung auf die Gesellschaft

Zeitbedarf: 3 Stunden á 90 Minuten</td>
</tr>
<tr><td colspan="2">Summe Einführungsphase: 35 Stunden á 90 Minuten</td></tr>
</table>

II Qualifikationsphase (Q1 und Q2 Grundkurs)

Qualifikationsphase 1	
Unterrichtsvorhaben Q1-I **Thema:** *Wiederholung der objektorientierten Modellierung und Programmierung anhand einer kontextbezogenen Problemstellung* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen und Automaten • Informatiksysteme **Inhaltliche Schwerpunkte:** • Objekte und Klassen • Analyse, Entwurf und Implementierung von Algorithmen • Syntax und Semantik einer Programmiersprache • Nutzung von Informatiksystemen **Zeitbedarf**: 5 Stunden á 90 Minuten	*Unterrichtsvorhaben Q1-II* **Thema:** *Modellierung und Implementierung von Anwendungen mit dynamischen, linearen Datenstrukturen* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen und Automaten **Inhaltliche Schwerpunkte:** • Objekte und Klassen • Analyse, Entwurf und Implementierung von Algorithmen • Algorithmen in ausgewählten informatischen Kontexten • Syntax und Semantik einer Programmiersprache **Zeitbedarf**: 15 Stunden á 90 Minuten

Qualifikationsphase 1	
Unterrichtsvorhaben Q1-III **Thema:** *Suchen und Sortieren auf linearen Datenstrukturen* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Algorithmen • Formale Sprachen und Automaten **Inhaltliche Schwerpunkte:** • Analyse, Entwurf und Implementierung von Algorithmen • Algorithmen in ausgewählten informatischen Kontexten • Syntax und Semantik einer Programmiersprache **Zeitbedarf**: 4 Stunden á 90 Minuten	*Unterrichtsvorhaben Q1-IV* **Thema:** *Rekursive Programmierung und 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**: 10 Stunden á 90 Minuten

Qualifikationsphase 1	
Unterrichtsvorhaben Q1-V **Thema:** *Sicherheit und Datenschutz in Netzstrukturen* **Zentrale Kompetenzen:** • Argumentieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Informatiksysteme • Informatik, Mensch und Gesellschaft **Inhaltliche Schwerpunkte:** • Einzelrechner und Rechnernetzwerke • Sicherheit • Nutzung von Informatiksystemen, Wirkungen der Automatisierung **Zeitbedarf**: 3 Stunden á 90 Minuten	
Summe Qualifikationsphase 1: 37 Stunden á 90 Minuten	

Qualifikationsphase 2	
Unterrichtsvorhaben Q2-I **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**: 10 Stunden á 90 Minuten	*Unterrichtsvorhaben Q2-II* **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**: 12 Stunden á 90 Minuten

Qualifikationsphase 2	
Unterrichtsvorhaben Q2-III **Thema:** *Prinzipielle Arbeitsweise eines Computers und Grenzen der Automatisierbarkeit* **Zentrale Kompetenzen:** • Argumentieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Informatiksysteme • Informatik, Mensch und Gesellschaft **Inhaltliche Schwerpunkte:** • Einzelrechner und Rechnernetzwerke • Grenzen der Automatisierung **Zeitbedarf:** 3 Stunden á 90 Minuten	
Summe Qualifikationsphase 2: 25 Stunden á 90 Minuten	

2.1.2 Konkretisierte Unterrichtsvorhaben

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

In der Einführungsphase wird die eine didaktische Bibliothek zur Darstellung grafischer Objekte verwendet. Die Wahl einer spezifischen grafischen Bibliothek ist nicht vorgeschrieben. Die Nutzung der modifizierten Bibliothek aus Barnes/Kölling, „Java lernen mit BlueJ“ wird jedoch empfohlen. Die Entwicklungsumgebung BlueJ ist frei zu erhalten unter

http://bluej.org/ (Windows, Mac, Linux, plattformunabhängiges Jar)

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

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

Die genutzte Entwicklungsumgebung in der Qualifikationsphase wird der Wahl der Lehrerin bzw. des Lehrers überlassen. Eine weitere Nutzung von BlueJ, sowie ein Wechsel auf eine komplexere IDE - etwa Eclipse - sind denkbar.

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 Programmierung und algorithmischer Grundstrukturen in Java anhand einfacher Rechenoperationen und grundlegender Datentypen

Leitfragen: *Womit beschäftigt sich die Informatik? Wie ist ein Programm in Java aufgebaut? Welche grundlegenden Funktionen und Mechanismen gibt es?*

Vorhabenbezogene Konkretisierung:
Das erste Unterrichtsvorhaben stellt eine allgemeine Einführung in das Fach Informatik und die Programmiersprache Java dar. Dabei ist zu berücksichtigen, dass für manche Schülerinnen und Schüler in der Einführungsphase der erste Kontakt mit dem Unterrichtsfach Informatik stattfindet, so dass zu Beginn Grundlagen des Fachs behandelt werden müssen.
Zunächst werden die grundlegende Struktur von Programmen in Java, primitive Datentypen (int, char, double) und einfache Mechanismen, wie etwa Initialisierung von Variablen, Zuweisen von Werten und die Ausgabe von Ergebnissen in die Konsole, behandelt. Die Schüler sollen diese Strukturen und Mechanismen beim Erstellen eigener Programme nutzen.
Anschließend an die grundlegenden Funktionen werden einfache Kontrollstrukturen behandelt (Verzweigung, For-Schleife). Die Schüler entwerfen erste eigene Algorithmen unter Zuhilfenahme dieser Kontrollstrukturen. Im Zuge der Einführung der Verzweigungen werden

Bedingungen und die darin genutzten logischen Verknüpfungen besprochen.
Komplexere Datentypen (String) und die Objektorientierung als Paradigma sollen zu diesem Zeitpunkt noch nicht thematisiert werden.

Zeitbedarf: Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
Syntax und Semantik von Java • Aufbau einer Klasse in Java • Deklaration von Variablen • Einfache Datentypen und deren Darstellung in Java • Zuweisung von Werten • Methoden • Parameter • Ausgabe von Werten in die Konsole	Die Schülerinnen und Schüler • beschreiben und analysieren den Aufbau einfacher Programme in Java (A), • nutzen das im Unterricht eingesetzte Informatiksystem selbstständig, sicher und zielführend (D), • Entwerfen Programme zu gegebenen Problemstellungen, testen und korrigieren sie (I) • erkennen die Grenzen der bekannten Funktionalität und entwickeln Ideen zur Erweiterung ebendieser (M)	*Beispiel*: Einfacher Taschenrechner Initialisierung, Belegung und Modifikation von Variablen Durchführung einfacher Rechnungen (Addition, Multiplikation, Modulo) Ausgabe der Ergebnisse in der Konsole
Entwurf und Implementierung einfacher Algorithmen • Nutzung mathematischer Operationen in Java • Verzweigungen und ihre Bedingungen • Nutzung logischer Operationen in Java • Einsatz von Rückgabewerter, Typ einer Methode • Einsatz von Schleifen		*Beispiel:* Bilanzrechner

Unterrichtsvorhaben EF-II

Thema: Grundlagen der objektorientierten Programmierung, Modellierung und Implementierung in anwendungsorientierten Kontexten

Leitfragen: *Wie kann man die Basisdatentypen erweitern/ergänzen? Wie kann man eigene Klassen/Datentypen erstellen und nutzen? Wie kann man eigene Klassen und deren Beziehungen angemessen darstellen?*

Vorhabenbezogene Konkretisierung:
Anhand der Datentypen *„char“* und *„String“* wird das Konzept der Objektorientierung und das Erstellen eigener, neuer Klassen bzw. Datentypen thematisiert. Instanzen der Klasse String sollen dabei als Objekte genutzt und ihre Methodik sinnvoll eingesetzt werden.

Die Schüler lernen Klassen-, Entwurfs- und Implementationsdiagramme als Möglichkeit der Planung von Implementierungsprozessen kennen. So können die Schüler in Anwendungskontexten erste eigene Klassen und deren Beziehungen konzipieren, präsentieren und anschließend implementieren.

Im Rahmen eines Projektes sollen die Schüler selbstständig ein Programm entwerfen und umsetzen.

Zeitbedarf: Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
Objektorientierung am Beispiel der Klasse String • Erstellen neuer Objekte als Instanz der Klasse String (Schlüsselwort „new“) • Detailbetrachtung des Konstruktors • Analyse der Klasse String • Methodenaufruf auf Objekten der Klasse	Die Schülerinnen und Schüler • diskutieren die Notwendigkeit neuer Datentypen (A), • erstellen eigene Klassen-, Entwurfs- und Implementationsdiagramme (D), • entwerfen Programme zu gegebenen Problemstellungen, testen und korrigieren sie (I) • erkennen die Grenzen der bekannten	*Beispiel*: String Die Klasse String als Datentyp wird thematisiert. Zur bequemeren Verwaltung von Zeichenketten soll die Klasse String besprochen werden. Anschließend wird die Klasse String genutzt, um die Grundlagen der Objektorientierung (Klassen/Objekte, Methoden, Konstruktor, Methodenaufruf) zu erlernen

String	Funktionalität und entwickeln Ideen zur Erweiterung ebendieser (M)	
Entwurf und Implementierung eigener Klassen • Entwurf eigener Klassen in Entwurfs- und Implementationsdiagrammen • Berücksichtigung der Beziehungen von Klassen • Implementierung der entworfenen Klassen		*Beispiel:* Projekt „Spiel“ Die Schüler entwerfen eine Spielidee, modellieren ihre Idee mithilfe der bekannten Diagrammtypen und implementieren ihre Spiele schließlich. Es soll sich dabei um einfach zeichenbasierte Spiele handeln, die mindestens zwei Klassen umfassen, aber ohne grafische Oberfläche ausschließlich auf der Konsole ablaufen.

Unterrichtsvorhaben EF-III

Thema: Grundlagen der objektorientierten Analyse, Modellierung und Implementierung graphischer Szenen mit Hilfe einer vorgegebenen Graphikbibliothek

Leitfragen: *Wie plant man mit Hilfe der zur Verfügung stehenden Diagrammformen die Implementierung komplexer Probleme? Wie nutzt man vorgegebene Bibliotheken?*

Vorhabenbezogene Konkretisierung:
Die Schüler modellieren unter Verwendung einer vorgegebenen Grafikbibliothek verschiedene grafische Szenen. Sie stellen ihre Entwürfe mit Hilfe von Entwurfs- und Implementationsdiagrammen dar. Im Kontext dieser Grafikszenen erlenen die Schüler weitere Konzepte der objektorientierten Programmierung kennen (Vererbung, Ploymorphie, abstrakte Klassen).
Zur Wiederholung der schon bekannten Kontrollstrukturen können auch animierte Szenen erstellt werden.

Zeitbedarf: Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
Modellierung vorgegebener Grafikszenen • Analyse vorgegebener Grafikszenen • Entwurf geeigneter Klassen und Methoden • Erstellen von Entwurfs- und Implementationsdiagrammen • Zur Vertiefung von Schleifen können animierte Szenen genutzt werden	Die Schülerinnen und Schüler • entwerfen ein Modell der dargestellten Szene (M, D) • diskutieren die verschiedenen Entwürfe und überprüfen sie auf Stichhaltigkeit (A) • implementieren selbst die dargestellte Szene (I) • erweitern ihr konzeptionelles und praktisches Verständnis der Objektorientierung um den Begriff der Vererbung (M, I) • wenden eine didaktisch orientierte Entwicklungsumgebung zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I)	*Beispiel*: verschieden grafische Szenen Einfache Darstellungen wie Häuser und Bäume, diese können ausgebaut werden zu komplexeren Szenen, wie Straßenzüge oder Skylines. Diese können bewegte Objekte enthalten wie etwa Autos oder Flugzeuge.
Erweiterung des Modells der Objektorientierung • Einsatz von Vererbung (in Modell und Quellcode) • Nutzen und Erkennen von Polymorphie • Nutzung abstrakter Klassen		

Unterrichtsvorhaben EF-IV

Thema: Grundlagen graphischer Benutzeroberflächen, Ereignissteuerung und das Listenerkonzept in Java an einfachen Beispielen

Leitfragen: *Wie kann eine Benutzereingabe mit Hilfe grafischer Benutzeroberflächen verarbeitet werden? Wie baut sich eine GUI in Java auf?*

Vorhabenbezogene Konkretisierung:
Die Schüler analysieren grafische Benutzeroberflächen und lernen das Listenerkonzept von Java kennen. Dabei erlernen sie an einfachen Benutzeroberflächen die Grundlagen der Ereignissteuerung. Die Eingaben bleiben dabei auf einfache Eingaben durch Maus und Tastatur beschränkt. Ziel dieses Unterrichtsvorhabens ist es nicht die Schüler in die Lage zu versetzen komplexe grafische Oberflächen zu implementieren. Die Schüler sollen am Ende in der Lage sein den Quellcode vorgegebener Projekte zu verstehen und die vorgegebene grafische Oberfläche mit der gewünschten Funktion zu versehen.
Aus diesem Grund wird auch davon abgeraten Swing einzusetzen. Die Funktion von AWT ist, wenn auch veraltet, so dennoch völlig ausreichend die grundlegen Prinzipien der GUI zu erlernen.

Zeitbedarf: Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
GUI • Analyse von grafischen Benutzeroberflächen • Entwurf eigener Benutzeroberflächen	Die Schülerinnen und Schüler • analysieren und erläutern die informatische Struktur von GUIs (A) • modellieren auf dieser Grundlage eigene GUIs (M) • nutzen die Listener Interfaces zur Erstellung eigener (I) • implementieren eigene GUIs (I)	
Listenerkonzept • Erweiterung des objektorientierten Konzeptes um Listenerklassen • Modellierung mit Listenerklassen • Implementierung der entsprechenden Listener		

Unterrichtsvorhaben E-V

Thema: Entwicklung und Implementierung von Such- und Sortieralgorithmen anhand kontextbezogener Beispiele

Leitfragen: *Wie können Objekte bzw. Daten effizient sortiert werden, so dass eine schnelle Suche möglich wird?*

Vorhabenbezogene Konkretisierung:
Dieses Unterrichtsvorhaben beschäftigt sich mit der Erarbeitung von Such- und Sortieralgorithmen. Dies umfasst sowohl die Algorithmen selbst, als auch deren Implementierung in Java (so weit den Schülern dies mögliche ist).
Zunächst erarbeiten die Schülerinnen und Schüler mögliche Einsatzszenarien für Such- und Sortieralgorithmen, um sich der Bedeutung einer effizienten Lösung dieser Probleme bewusst zu werden. Anschließend werden Strategien zur Sortierung mit Hilfe eines explorativen Spiels von den Schülerinnen und Schülern selbst erarbeitet und hinsichtlich der Anzahl notwendiger Vergleiche auf ihre Effizienz untersucht.
Daran anschließend werden die erarbeiteten Strategien systematisiert und im Pseudocode notiert. Die Schülerinnen und Schüler sollen auf diese Weise das *Sortieren durch Vertauschen*, das *Sortieren durch Auswählen* und mindestens einen weiteren Sortieralgorithmus, kennen lernen. Ein Teil dieser Algorithmen soll anschließend kurz implementiert werden.
Ein rekursives Sortierverfahren soll behandelt und nach Effizienzgesichtspunkten untersucht werden.

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
Explorative Erarbeitung von Sortierverfahren • Sortierprobleme im Kontext informatischer Systeme und im Alltag (z.B. Dateisortierung, Tabellenkalkulation, Telefonbuch, Bundesligatabelle, usw.) • Vergleich zweier Elemente als Grundlage	Die Schülerinnen und Schüler • beurteilen die Effizienz von Algorithmen am Beispiel von Sortierverfahren hinsichtlich Zeit und Speicherplatzbedarf (A), • entwerfen einen weiteren Algorithmus	*Beispiel*: Sortieren mit Waage Die Schülerinnen und Schüler bekommen die Aufgabe, kleine, optisch identische Kunststoffbehälter aufsteigend nach ihrem Gewicht zu sortieren. Dazu steht ihnen eine Balkenwaage zur Verfügung, mit deren Hilfe sie das Gewicht zweier Behälter vergleichen

eines Sortieralgorithmus • Erarbeitung eines Sortieralgorithmus durch die Schülerinnen und Schüler	zum Sortieren (M), • analysieren Such- und Sortieralgorithmen und wenden sie auf Beispiele an (D).	können.
Systematisierung von Algorithmen und Effizienzbetrachtungen • Formulierung (falls selbst gefunden) oder Erläuterung von mehreren Algorithmen im Pseudocode (auf jeden Fall: Sortieren durch Vertauschen, Sortieren durch Auswählen) • Anwendung von Sortieralgorithmen auf verschiedene Beispiele • Implementierung eines Algorithmus • Bewertung von Algorithmen anhand der Anzahl der nötigen Vergleiche • Variante des Sortierens durch Auswählen (Nutzung eines einzigen oder zweier Felder bzw. lediglich eines einzigen zusätzlichen Ablageplatzes oder mehrerer neuer Ablageplätze) • Effizienzbetrachtungen an einem konkreten Beispiel bezüglich der Rechenzeit und des Speicherplatzbedarfs • Analyse des weiteren Sortieralgorithmus (sofern nicht in Sequenz 1 und 2 bereits geschehen)		*Beispiel:* Sortieren durch Auswählen, Sortieren durch Vertauschen, Mergesort Quicksort ist als Beispiel für einen Algorithmus nach dem Prinzip *Teile und Herrsche* gut zu behandeln. Kenntnisse in rekursiver Programmierung sind nicht erforderlich, da eine Implementierung nicht angestrebt wird.

Unterrichtsvorhaben EF-VI

Thema: Der grundlegende Aufbau und die Funktionsweise von Netzstrukturen und deren Auswirkungen auf die Gesellschaft

Leitfragen: *Wie hat sich die digitale Datenverarbeitung entwickelt? Was sind grundlegende Elemente von Computernetzwerken? Was sind die Grundlagen der Kommunikation in vernetzten Systemen? Wie funktioniert der Client-Server-Zugriff?*

Vorhabenbezogene Konkretisierung:
Die Schüler lernen in Ausschnitten die Geschichte der digitalen Datenverarbeitung kennen. Anschließend wird der Aufbau von lokalen Netzwerken und Adressierung in Netzwerken thematisiert. Dabei werden nur die einfachen Grundlagen thematisiert (etwa ein Schichtenmodell).

Eine Anwendung, die auf vernetzen Systemen basiert wird analysiert und deren Funktionsweise erörtert. Dies kann ein Mailprogramm oder ein Browser sein.

Die Auswirkungen der Vernetzung, sowie Chancen und Gefahren werden von den Schülern diskutiert.

Zeitbedarf: Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
Vernetzte Systeme • Betrachtung der Geschichte vernetzter Systeme • Verfahren zur Adressierung in verschiedenen Netzwerken wird erarbeitet • Netztopoligie als Grundlage von Client-Server-Strukturen und Protokolle in Netzwerken • Anwendungen in Netzwerken • Sicherheit in Netwerken, Verschlüsselung, Authentifizierung, Kryptographie	Die Schülerinnen und Schüler • erläutern wesentliche Grundlagen der Geschichte der digitalen Datenverarbeitung (A), • nutzen das Internet zur Recherche, zum Datenaustausch und zur Kommunikation. (K). • beschreiben und erläutern Topologien, die Client-Server-Struktur und Protokolle sowie ein Schichtenmodell in Netzwerken (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).	

II Qualifikationsphase (Q1 und Q2 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 Q1-1

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

Leitfragen: *Wie modelliert und implementiert man zu einer Problemstellung in einem geeigneten Anwendungskontext Java-Klassen inklusive ihrer Attribute, Methoden und Beziehungen? Wie kann man die Modellierung und die Funktionsweise der Anwendung grafisch darstellen?*

Vorhabenbezogenen Konkretisierung:
Zu einer Problemstellung in einem Anwendungskontext soll eine Java-Anwendung entwickelt werden. Die Problemstellung soll so gewählt sein, dass für diese Anwendung die Verwendung einer abstrakten Oberklasse als Generalisierung verschiedener Unterklassen sinnvoll erscheint und eine Klasse durch eine Unterklasse spezialisiert werden kann. Um die Aufgabe einzugrenzen, können (nach der ersten Problemanalyse) einige Teile (Modellierungen oder Teile von Java-Klassen) vorgegeben werden.

Die Schülerinnen und Schülern erläutern und modifizieren den ersten Entwurf und modellieren sowie implementieren weitere Klassen und Methoden für eine entsprechende Anwendung. Klassen und ihre Beziehungen werden in einem Implementationsdiagramm dargestellt. Dabei werden Sichtbarkeitsbereiche zugeordnet. Der Nachrichtenaustausch zwischen verschiedenen Objekten wird verdeutlicht, indem die Kommunikation zwischen zwei ausgewählten Objekten grafisch dargestellt wird. In diesem Zusammenhang wird das Nachrichtenkonzept der objektorientierten Programmierung wiederholt.

Zeitbedarf: 8 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
Wiederholung und Erweiterung der objektorientierten Modellierung und Programmierung durch Analyse und Erweiterung eines kontextbezogenen Beispiels • Analyse der Problemstellung • Analyse der Modellierung (Implementationsdiagramm) • Erweiterung der Modellierung im Implementationsdiagramm (Vererbung, abstrakte Klasse) • Kommunikation zwischen mindestens zwei Objekten (grafische Darstellung) • Implementierung der Anwendung oder von Teilen der Anwendung	Die Schülerinnen und Schüler • analysieren und erläutern objektorientierte Modellierungen (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (M), • ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M), • modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I),	*Beispiel:* Bibliothek Ein Verwaltungsprogramm für die Inhalte einer Bibliothek soll erstellt werden. Dabei bieten die verschieden Medien die Möglichkeit über eine abstrakte Klasse eine einheitliche Schnittstelle zu schaffen. Grafische Oberflächen können in diesem Projekt erneut thematisiert werden, die Oberfläche kann den Schülern jedoch zur Verfügung gestellt werden. Die zugrunde liegende Datenstruktur soll dabei zunächst ein Array sein. Dies soll später den Übergang in die linearen Datenstrukturen ermöglichen. *Beispiel:* Rangierbahnhof Es soll die Simulation eines Güterbahnhofes programmiert werden. Verschiedene Arten von Waggons bieten die Möglichkeit abstrakte Klassen in das Projekt einzubinden. Unterschiedliche Gleistypen (Durchgangs-/Abstellgleis) benötigen verschiedene Formen der Verwaltung. Eine grafische Oberfläche kann den Schülern zur Verfügung gestellt werden. Als Datenstruktur zur Verwaltung der Gleise kann nur das Array genutzt werden.

	• wenden eine didaktisch orientierte Entwicklungsumgebung zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D), • dokumentieren Klassen (D)	

Unterrichtsvorhaben Q1-II:

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

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

Vorhabenbezogene Konkretisierung:

Nach Analyse einer Problemstellung in einem geeigneten Anwendungskontext, werden die Grenzen der Datenstruktur Array aufgezeigt. Der Entwurf eines neuen Datentyps wird erörtert, dabei soll der Begriff des abstrakten Datentyps eingeführt werden. Die Struktur und Funktionalität der Liste werden erläutert und die Liste wird anschließend in den Anwendungskontext implementiert. Im Mittelpunkt der Implementierung soll dabei die Anwendung von Operationen auf der Datenstruktur Liste stehen, nicht aber die Implementierung der Datenstruktur selbst. Klassen, die zur Gestaltung des Anwendungskontextes und nicht zur Umsetzung der Liste im Anwendungskontext dienen, sowie die Klasse Liste selbst, können dabei vom Lehrer vorgegeben werden.
Im Anwendungskontext wird das Last-In-First-Out-Prinzip erarbeitet. Der Aufbau und die Funktionalität des Stacks werden erörtert. Die Klasse Stack soll anschließend in den Anwendungskontext implementiert werden. Auch hier soll das Operieren auf der Datenstruktur Stack Schwerpunkt der Betrachtung sein.
Im Weiteren wird in einem Anwendungskontext das First-in-First-out Prinzip in Abgrenzung zum Stack diskutiert. Die Datenstruktur

Queue wird eingeführt und Unterschiede und Gemeinsamkeiten zwischen Stack und Queue werden diskutiert. Auch die Queue wird im Anwendungskontext implementiert mit entsprechendem Schwerpunkt auf den Operationen auf der Datenstruktur Queue.

In weiteren Kontexten wird die Verwaltung von Daten in Schlangen, Stapeln oder Listen vertieft. Modellierungen werden dabei in Entwurfs- und Implementationsdiagrammen dargestellt. Dabei bieten sich Anwendungskontexte wie algorithmische Kontexte an.

Zeitbedarf: 20 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. **Die Datenstruktur lineare Liste im Anwendungskontext unter Nutzung der Klasse List** a. Erarbeitung der Vorteile der Klasse List im Gegensatz zu den bereits bekannten linearen Strukturen b. Modellierung und Implementierung einer kontextbezogenen Anwendung unter Verwendung der Klasse List.	Die Schülerinnen und Schüler • erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A), • analysieren und erläutern Algorithmen und Programme (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu (M), • ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • modifizieren Algorithmen und Programme (I),	*Beispiel:* Bibliothek Die in der entworfenen Bibliothek genutzte Datenstruktur zur Verwaltung der Inhalte wird auf durch eine Liste ersetzt (Motivation: Endlichkeit und Speicherbedarf des Arrays). Durch ein lexikographisch korrektes Einfügen kann die dynamische Struktur der Liste voll ausgenutzt werden. *Beispiel:* Rangierbahnhof Im Rangierbahnhof kann äquivalent vorgegangen werden. Die Zusammenstellung eines Zuges nutzt die dynamische Größe und Sortierung einer Liste. *Materialien:* Materialien zum Zentralabitur LINK!
2. Die Datenstruktur Stapel im Anwendungskontext unter Nutzung der Klasse Stack a. Analyse der Problemstellung,		*Beispiel:* Rückgabestapel Die Rückgabe von Büchern wird in das Projekt integriert. Die Rückgabe von Büchern erfolgt in Form eines Stapels (Abbild der Wirklichkeit). Die

Ermittlung von Objekten, ihren Eigenschaften und Operationen b. Erarbeitung der Funktionalität der Klasse Stack *c.* Modellierung und Implementierung der Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse Stack	• implementieren Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • testen Programme systematisch anhand von Beispielen (I), • stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D).	zurückgegebenen Bücher können dann schrittweise wieder in die Bibliothek einsortiert werden. *Beispiel:* Rangierbahnhof Im Rangierbahnhof wird das Abstellgleis durch einen Stapel realisiert. *Materialien:* Material zum Zentralabitur LINK!
3. Die Datenstruktur Schlange im Anwendungskontext unter Nutzung der Klasse Queue a. Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen b. Erarbeitung der Funktionalität der Klasse Queue *c.* Modellierung und Implementierung der Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse Queue		*Beispiel*: Reparatur Die Bücher der Bibliothek werden um ein Attribut erweitert, das den Zustand des Buches angibt. Nach einer bestimmten Anzahl an Verleihungen wird das buch nach der Rückgabe in eine Reparaturwarteschlange eingefügt. Diese Warteschlange kann schrittweise abgearbeitet werden und die Bücher werden wieder in die Bibliothek zurückgestellt. *Beispiel:* Rangierbahnhof Im Rangierbahnhof wird ein Durchgangsgleis durch eine Queue realisiert. *Materialien:* Material zum Zentralabitur LINK!
4. Vertiefung - Anwendungen von Listen, Stapeln oder Schlangen in mindestens einem weiteren Kontext		*Beispiel:* Algorithmik auf Stack/Queue Ein Stack bzw eine Queue wird mit einer Folge von Zufallszahlen gefüllt. Mithilfe zweier weiterer Stacks soll die Zahlenfolge nun sortiert werden. Wiederum ausgehend von drei Stacks ist einer der Stacks mit einem mathematischen Term gefüllt, der nun entsprechend den geltenden

		mathematischen Regeln ausgewertet werden muss. Die Schwierigkeit kann dabei etwa durch Klammern an die Leistungsfähigkeit des Kurses angepasst werden. Die sog. UPN (*Umgekehrt-Polnische-Notation*) bzw. *Postfix-Notation* eines Terms setzt den Operator hinter die Operanden. Um einen Term aus der gewohnten Infixschreibweise in einen Term in UPN umzuwandeln oder um den Wert des Terms zu berechnen, kann ein Stack verwendet werden. *Weitere Beispiele zur Übung und Vertiefung:* Als weitere Übungsbeispiele oder Projekte können etwa die Umsetzung einer Patientenverwaltung in einer Arztpraxis oder die Verwaltung der Kunden in einem Amt („Nummern ziehen“) dienen.

Unterrichtsvorhaben Q1-III:

Thema: Suchen und Sortieren auf linearen Datenstrukturen

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

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

Es werden Sortierverfahren entwickelt und implementiert (ebenfalls für lineare Listen und Felder). Die Implementationen eines Sortierverfahrens wird analysiert und erläutert.

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

Die Rekursion wird in diesem Kursabschnitt nicht explizit geübt oder thematisiert, sondern intuitiv genutzt. In der Folgenden Sequenz wird die Rekursion genauer betrachtet. Beispiele aus dieser Sequenz können dabei als Motivation und Anschauung dienen.

Zeitbedarf: 16 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. **Suchen von Daten in Listen und Arrays** a. Lineare Suche in Listen und in Arrays b. Binäre Suche in Arrays als Beispiel für intuitiv rekursives Problemlösen *c.* Untersuchung der beiden Suchverfahren hinsichtlich ihrer Effizienz (Laufzeitverhalten, Speicherbedarf)	Die Schülerinnen und Schüler • analysieren und erläutern Algorithmen und Programme (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • beurteilen die Effizienz von Algorithmen unter Berücksichtigung des Speicherbedarfs und der Zahl der Operationen (A), • entwickeln iterative und intuitiv rekursive Algorithmen unter Nutzung der Strategien „Modularisierung“ und „Teilen und Herrschen“ (M), • modifizieren Algorithmen und	*Beispiel:* Adressbuch Für ein Adressverwaltungsprogramm soll eine Methode zum Suchen einer Adresse oder Telefonnummer geschrieben werden. *Beispiel*: Bundesjugendspiele Die Teilnehmer an Bundesjugendspielen nehmen an drei Disziplinen teil und erreichen dort Punktzahlen. Diese werden in einer Wettkampfkarte eingetragen und an das Wettkampfbüro gegeben. Zur Vereinfachung sollte sich das Modell auf die drei Disziplinen „Lauf”, „Sprung“ und „Wurf“ beschränken. Im Wettkampfbüro wird das Ergebnis erstellt. Das Programm soll dafür zunächst den Besten einer Disziplin heraussuchen können und später das gesamte Ergebnis nach gewissen Kriterien

	Programme (I), • implementieren iterative Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • implementieren und erläutern iterative Such- und Sortierverfahren (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • testen Programme systematisch anhand von Beispielen (I), • stellen iterative Algorithmen umgangssprachlich und grafisch dar (D).	sortieren können.
2. **Sortieren in Listen und Arrays - Entwicklung und Implementierung von iterativen und intuitiv rekursiven Sortierverfahren** a. Entwicklung und Implementierung eines einfachen Sortierverfahrens für eine Liste b. Implementierung eines einfachen Sortierverfahrens für ein Feld *c.* Kennenlernen eines rekursiven Sortierverfahren für ein Feld (z.B. Sortieren durch Mischen)		*Beispiel:* Adressbuch (s.o.) *Beispiel*: Bundesjugendspiele (s.o.)
3. **Untersuchung der Effizienz von Sortierverfahren** a. Grafische Veranschaulichung der Sortierverfahren b. Untersuchung der Anzahl der Vergleichsoperationen und des Speicherbedarf bei beiden Sortierverfahren *c.* Beurteilung der Effizienz der beiden Sortierverfahren		*Beispiel:* Adressverwaltung (s.o.) *Beispiel*: Bundesjugendspiele (s.o.)

Unterrichtsvorhaben Q1-IV:

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

Leitfragen: *Was ist Rekursion? 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:

Rekursion wird thematisiert und als Konzept an einigen einfachen Beispielen geübt. Dazu bieten sich mathematische Funktionen und Anwendungsbeispiele (etwa die Türme von Hanoi) an.

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

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

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

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

Zu einer Problemstellung in einem entsprechenden Anwendungskontext werden die Operationen der Datenstruktur Suchbaum thematisiert und unter der Verwendung der Klasse BinarySearchTree (der Materialien für das Zentralabitur in NRW) weitere Klassen oder

Methoden in diesem Anwendungskontext modelliert und implementiert. Auch in diesem Kontext werden grafische Darstellungen der Bäume verwendet.

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

Zeitbedarf: 24 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. **Erörterung der Rekursion und Analyse von Baumstrukturen in verschiedenen Kontexten** a. Grundlegende Begriffe (Grad, Tiefe, Höhe, Blatt, Inhalt, Teilbaum, Ebene, Vollständigkeit) b. Aufbau und Darstellung von binären Bäumen anhand von Baumstrukturen in verschiedenen Kontexten	Die Schülerinnen und Schüler • entwickeln und implementieren rekursive Algorithmen • 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	*Beispiel:* Expertensystem Ein Binärbaum wird zur Umsetzung eines einfachen Expertensystems (in Anlehnung an den Akinator o.ä.) genutzt. *Beispiel*: Morsebaum Der Binärbaum wird zu Kodierung und Dekodierung von Morsecode genutzt. *Beispiel*: Suchbäume (zur sortierten Speicherung von Daten) Alle Inhalte, die nach einer Ordnung vor dem Inhalt im aktuellen Teilbaum stehen, sind in dessen linkem Teilbaum, alle die nach dem Inhalt im aktuellen Teilbaum stehen, sind in dessen rechtem Teilbaum. (Dies gilt für alle Teilbäume.) *Beispiel*: Huffman Code

	Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu (M), • modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M), • verwenden bei der Modellierung geeigneter Problemstellungen die Möglichkeiten der Polymorphie (M), • entwickeln iterative und rekursive Algorithmen unter Nutzung der Konstruktionsstrategien „Modularisierung“ und „Teilen und Herrschen“ (M), • implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • modifizieren Algorithmen und Programme (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • testen Programme systematisch anhand von Beispielen (I),	Ein Binärbaum wird genutzt, um einen Text mit dem Huffman Code zu kodieren und ggf. zu dekodieren. *Beispiel:* Termbäume Der Binärbaum wird zur Darstellung und korrekten Auswertung mathematischer Terme genutzt.
***2.* Die Datenstruktur Binärbaum im Anwendungskontext unter Nutzung der Klasse BinaryTree** a. Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen im Anwendungskontext b. Modellierung eines Entwurfsdiagramms und Entwicklung eines Implementationsdiagramms c. Erarbeitung der Klasse BinaryTree und beispielhafte Anwendung der Operationen d. Implementierung der Anwendung oder von Teilen der Anwendung		*Beispiele:* s.o. Alle obigen Beispiele können auch in diesem Abschnitt verwendet werden.
3. Die Datenstruktur binärer Suchbaum im Anwendungskontext unter Verwendung der Klasse BinarySearchTree a. Analyse der Problemstellung,		*Beispiel*: Beliebige Suchbäume In einem binären *Suchbaum* werden Daten bezüglich einer Ordnungsrelation geordnet abgespeichert (etwa Zahlen bzgl. ihrer Größe oder Strings bzgl. ihrer lexikographischen Ordnung). Alle Daten, die nach dieser Ordnung

Ermittlung von Objekten, ihren Eigenschaften und Operationen b. Modellierung eines Entwurfsdiagramms und Entwicklung eines Implementationsdiagramm, c. grafische Darstellung eines binären Suchbaums und Erarbeitung der Struktureigenschaften d. Erarbeitung der Klasse BinarySearchTree und Einführung des Interface Item zur Realisierung einer geeigneten Ordnungsrelation **e.** Implementierung der Anwendung oder von Teilen der Anwendung inklusive einer sortierten Ausgabe des Baums	• stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D), • stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D).	vor der Wurzel im aktuellen Teilbaum stehen, sind in dessen linkem Teilbaum, alle die nach dem Namen im aktuellen Teilbaum stehen, sind in dessen rechtem Teilbaum. (Dies gilt für alle Teilbäume.) Folgende Funktionalitäten werden benötigt: • Einfügen der Informatiker-Daten in den Baum • Suchen nach einem Informatiker über den Schlüssel Name • Ausgabe des kompletten Datenbestands in nach Namen sortierter Reihenfolge
4. **Übung und Vertiefungen der Verwendung von Binärbäumen oder binären Suchbäumen anhand weiterer Problemstellungen**		*Beispiel:* Codierungsbäume, Ahnenbaum, Termbaum (s.o.) *Beispiel*: Buchindex Es soll eine Anwendung entwickelt werden, die anhand von Stichworten und zugehörigen Seitenzahlen ein Stichwortregister erstellt. Da die Stichwörter bei der Analyse des Buches häufig gesucht werden müssen, werden sie in der Klasse Buchindex als Suchbaum (Objekt der Klasse BinarySearchTree) verwaltet. Alle Inhalte, die nach einer Ordnung vor dem

		Inhalt im aktuellen Teilbaum stehen, sind in dessen linkem Teilbaum, alle die nach dem Inhalt im aktuellen Teilbaum stehen, sind in dessen rechtem Teilbaum. (Dies gilt für alle Teilbäume.) *Beispiel*: Entscheidungsbäume (s.o. Expertensystem)

Unterrichtsvorhaben Q1-V:

Thema: Sicherheit und Datenschutz in Netzstrukturen

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

Vorhabenbezogene Konkretisierung:

In dieser Sequenz werden der Datenbankzugriff aus dem Netz, Topologien von Netzwerken, eine Client-Server-Struktur, das TCP/IP-Schichtenmodell sowie Sicherheitsaspekte beim Zugriff auf Datenbanken und verschiedene symmetrische und asymmetrische kryptografische Verfahren analysiert und erläutert. Fallbeispiele zur Datenschutzproblematik und zum Urheberrecht runden das Unterrichtsvorhaben ab.

Zeitbedarf: 10 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Daten in Netzwerken und Sicherheitsaspekte in Netzen sowie beim Zugriff auf Datenbanken	Die Schülerinnen und Schüler • beschreiben und erläutern Topologien, die Client-Server-Struktur	*Materialien:*

a. Beschreibung eines Datenbankzugriffs im Netz anhand eines Anwendungskontextes und einer Client-Server-Struktur zur Klärung der Funktionsweise eines Datenbankzugriffs b. Netztopologien als Grundlage von Client-Server-Strukturen und TCP/IP-Schichtenmodell als Beispiel für eine Paketübermittlung in einem Netz c. Vertraulichkeit, Integrität, Authentizität in Netzwerken sowie symmetrische und asymmetrische kryptografische Verfahren (Cäsar-, Vigenère-, RSA-Verfahren) als Methoden Daten im Netz verschlüsselt zu übertragen	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).	

Unterrichtsvorhaben Q2-I:

Thema: Endliche Automaten und formale Sprachen

Leitfragen: *Wie kann man (endliche) Automaten genau beschreiben? Wie können endliche Automaten (in alltäglichen Kontexten oder zu informatischen Problemstellungen) modelliert werden? Wie können Sprachen durch Grammatiken beschrieben werden? Welche Zusammenhänge gibt es zwischen formalen Sprachen, endlichen Automaten und regulären Grammatiken?*

Vorhabenbezogene Konkretisierung:

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

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

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

Zeitbedarf: 20 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien oder Materialien
1. Endliche Automaten a. Vom Automaten in den Schülerinnen und Schülern bekannten Kontexten zur formalen Beschreibung eines endlichen Automaten b. Untersuchung, Darstellung und Entwicklung endlicher Automaten	Die Schülerinnen und Schüler • analysieren und erläutern die Eigenschaften endlicher Automaten einschließlich ihres Verhaltens auf bestimmte Eingaben (A), • analysieren und erläutern Grammatiken regulärer Sprachen (A), • zeigen die Grenzen endlicher Automaten und regulärer Grammatiken im Anwendungszusammenhang auf (A), • ermitteln die formale Sprache, die durch eine Grammatik erzeugt wird (A), • entwickeln und modifizieren zu einer Problemstellung endliche Automaten (M), • entwickeln und modifizieren zu einer Problemstellung endliche Automaten (M), • entwickeln zur akzeptierten Sprache eines Automaten die zugehörige Grammatik (M), • entwickeln zur Grammatik einer regulären Sprache einen zugehörigen endlichen Automaten (M), • modifizieren Grammatiken regulärer	*Beispiele:* Inselspringen, (Piratenspiel), Getränkeautomat, Zustandsänderung eines Objekts „Auto“, Akzeptor für bestimmte Zahlen, Akzeptor für Teilwörter in längeren Zeichenketten, Akzeptor für Terme
***2.* Untersuchung und Entwicklung von Grammatiken regulärer Sprachen** a. Erarbeitung der formalen Darstellung regulärer Grammatiken b. Untersuchung, Modifikation und Entwicklung von Grammatiken c. Entwicklung von endlichen Automaten zum Erkennen regulärer Sprachen die durch Grammatiken gegeben werden d. Entwicklung regulärer Grammatiken zu endlichen Automaten		*Beispiele:* reguläre Grammatik für Wörter mit ungerader Parität, Grammatik für Wörter, die bestimmte Zahlen repräsentieren,
***3.* Grenzen endlicher Automaten**		

	Sprachen (M), • entwickeln zu einer regulären Sprache eine Grammatik, die die Sprache erzeugt (M), • stellen endliche Automaten in Tabellen oder Graphen dar und überführen sie in die jeweils andere Darstellungsform (D), • ermitteln die Sprache, die ein endlicher Automat akzeptiert (D). • beschreiben an Beispielen den Zusammenhang zwischen Automaten und Grammatiken (D).	

Unterrichtsvorhaben Q2-II:

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:

Die Schüler analysieren die Probleme dateibasierter Datenhaltung und entwickeln Anforderungen an ein Datenbank Managementsystem. Anhand dieser Anforderungen werden in verschiedenen Anwendungskontexten ER-Diagramme entworfen, welche die Notwendigkeiten der Datenbank im Kontext abbilden. Die Begriffe Entität (stark/schwach), Attribut (ein-/mehrwertig), Schlüssel (primär, Diskriminator) und verschiedene Beziehungstypen sowie deren Kardinalitäten werden dabei berücksichtigt.

Zur Umsetzung der Datenbanken werden die Datenbanken dann in das relationale Datenbankenmodell überführt. Zur Abfrage von Informationen wird zunächst die relationale Algebra verwendet. Es können schrittweise die grundlegenden, zusammengesetzten und

erweiterten Operatoren eingeführt werden. Mithilfe der relationalen Algebra können nun gewünschte Daten aus der Datenbank extrahiert werden.

Die Abfragen werden anschließen in SQL umgesetzt. Dazu werden die der relationalen Algebra entsprechenden Befehle in SQL genutzt.

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

Zeitbedarf: 20 Stunden

Sequenzierung des Unterrichtsvorhabens

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. **Nutzung eines DBMS und Modellierung einer Datenbank in einem DBMS durch ER-Diagramme** a. Notwendigkeit eines DBMS, Abgrenzung zu dateibasierter Datenhaltung • Analyse von Problemen in der Datenhaltung • Formulierung von Anforderungen an ein DBMS b. ER- Diagramme • Begriffe der Entität, Attribut, Relation, Kardinalität, Schlüssel • Modellierung einer Datenbank unter Berücksichtigung der oben genannten Begriffe	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),	*Beispiel:* Schule, Universität, Bank, Firma Datenbanken aus beliebigen Bereichen können in dieser Reihe verwendet werden. Praktikable Beispiele sind etwa die Datenhaltung in einer Schule, in der Schüler, Lehrer und weiteres Personal entsprechend verwaltet werden muss.. Man kann die Datenbank um Klassen, Fächer und Unterricht erweitern. Äquivalent lässt sich die Datenverwaltung einer Universität darstellen. Ebenso kann die Datenhaltung einer Bank durch die Verwaltung von Kunden, Konten in verschiedenen Ausprägungen, Krediten, Zweigstellen usw. umgesetzt werden. In einem ähnlichen Szenario kann die Verwaltung einer Firma mit ihren Kunden, Aufträgen, Zulieferern etc. als Grundlage einer

c. Vertiefung an einem weiteren Datenbankbeispiel	• modifizieren eine Datenbankmodellierung (M), • modellieren zu einem Entity-Relationship-Diagramm ein relationales Datenbankschema (M), • bestimmen Primär- und Sekundärschlüssel (M), • überführen Datenbankschemata in vorgegebene Normalformen (M), • verwenden die Syntax und Semantik einer Datenbankabfragesprache, um Informationen aus einen Datenbanksystem zu extrahieren (I), • ermitteln Ergebnisse von Datenbankabfragen über mehrere verknüpfte Tabellen (D), • stellen Entitäten mit ihren Attributen und die Beziehungen zwischen Entitäten in einem Entity-Relationship-Diagramm grafisch dar (D), • überprüfen Datenbankschemata auf vorgegebene Normalisierungseigenschaften (D).	Datenbank gewählt werden.
***2.* Modellierung von relationalen Datenbanken** a. Entwicklung einer relationalen Dantebank aus einem ER-Diagramm inklusive Bestimmung der Schlüssel b. Entwicklung von Anfragen an eine relationale Datenbank mithilfe der relationalen Algebra • Operatoren der relationalen Algebra werden erarbeitet und geübt • Anfragen an die entwickelte relationale Datenbank werden entwickelt c. 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) d. 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)		

Unterrichtsvorhaben Q2-III:

Thema: Prinzipielle Arbeitsweise eines Computers und Grenzen der Automatisierbarkeit

Leitfragen: *Was sind die strukturellen Hauptbestandteile eines Computers und wie kann man sich die Ausführung eines maschinenahen Programms mit diesen Komponenten vorstellen? Welche Möglichkeiten bieten Informatiksysteme und wo liegen ihre Grenzen?*

Vorhabenbezogene Konkretisierung:

Anhand einer von-Neumann-Architektur und einem maschinennahen Programm wird die prinzipielle Arbeitsweise von Computern verdeutlicht.

Ausgehend von den prinzipiellen Grenzen endlicher Automaten liegt die Frage nach den Grenzen von Computern bzw. nach Grenzen der Automatisierbarkeit nahe. Mit Hilfe einer entsprechenden Java-Methode wird plausibel, dass es unmöglich ist, ein Informatiksystem zu entwickeln, dass für jedes beliebige Computerprogramm und jede beliebige Eingabe entscheidet ob das Programm mit der Eingabe terminiert oder nicht (Halteproblem). Anschließend werden Vor- und Nachteile der Grenzen der Automatisierbarkeit angesprochen und der Einsatz von Informatiksystemen hinsichtlich prinzipieller Möglichkeiten und prinzipieller Grenzen beurteilt.

Zeitbedarf: 12 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien oder Materialien
1. Von-Neumann-Architektur und die Ausführung maschinennaher Programme a. prinzipieller Aufbau einer von Neumann-Architektur mit CPU, Rechenwerk, Steuerwerk, Register und Hauptspeicher b. einige maschinennahe Befehlen und ihre Repräsentation in einem Binär-Code, der in einem Register gespeichert werden kann c. Analyse und Erläuterung der Funktionsweise eines einfachen maschinennahen Programms	Die Schülerinnen und Schüler • erläutern die Ausführung eines einfachen maschinennahen Programms sowie die Datenspeicherung auf einer „Von-Neumann-Architektur“ (A), • untersuchen und beurteilen Grenzen des Problemlösens mit Informatiksystemen (A).	*Beispiel:* Addition von 4 zu einer eingegeben Zahl mit einem Rechnermodell
***2.* Grenzen der Automatisierbarkeit** a. Vorstellung des Halteproblems b. Unlösbarkeit des Halteproblems **c.** Beurteilung des Einsatzes von Informatiksystemen hinsichtlich prinzipieller Möglichkeiten und prinzipieller Grenzen		*Beispiel:* Halteproblem

2.2 Grundsätze der fachmethodischen und fachdidaktischen Arbeit

In Absprache mit der Lehrerkonferenz sowie unter Berücksichtigung des Schulprogramms hat die Fachkonferenz Informatik 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 27 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 Schülerinnen und Schüler.

3.) Die Unterrichtsgestaltung ist auf die Ziele und Inhalte abgestimmt.

4.) Medien und Arbeitsmittel sind lernernah gewählt.

5.) Die Schülerinnen und Schüler erreichen einen Lernzuwachs.

6.) Der Unterricht fördert und fordert eine aktive Teilnahme der Lernenden.

7.) Der Unterricht fördert die Zusammenarbeit zwischen den Lernenden und bietet ihnen Möglichkeiten zu eigenen Lösungen.

8.) Der Unterricht berücksichtigt die individuellen Lernwege der einzelnen Schülerinnen und Schüler.

9.) Die Lernenden erhalten Gelegenheit zu selbstständiger Arbeit und werden dabei unterstützt.

10.) Der Unterricht fördert strukturierte und funktionale Einzel-, Partner- bzw. Gruppenarbeit sowie Arbeit in kooperativen Lernformen.

11.) Der Unterricht fördert strukturierte und funktionale Arbeit im Plenum.

12.) Die Lernumgebung ist vorbereitet; der Ordnungsrahmen wird eingehalten.

13.) Die Lehr- und Lernzeit wird intensiv für Unterrichtszwecke genutzt.

14.) Es herrscht ein positives pädagogisches Klima im Unterricht.

Fachliche Grundsätze:

15.) Der Informatikunterricht ist problemorientiert und wenn möglich an Unterrichtsvorhaben und Kontexten ausgerichtet.

16.) Der Informatikunterricht ist kognitiv aktivierend und verständnisfördernd.

17.) Der Informatikunterricht unterstützt durch seine praktische Ausrichtung Lernprozesse bei Schülerinnen und Schülern.

18.) Der Informatikunterricht ist kumulativ, d.h. er knüpft soweit möglich an die Vorerfahrungen und das Vorwissen der Lernenden an und ermöglicht den Erwerb von Kompetenzen.

19.) Der Informatikunterricht fördert vernetzendes Denken und zeigt dazu eine über die verschiedenen Organisationsebenen bestehende Vernetzung von informatischen Konzepten und Prinzipien mithilfe von Basiskonzepten auf.

20.) Der Informatikunterricht folgt dem Prinzip der Exemplarizität und gibt den Lernenden die Gelegenheit, Strukturen und Gesetzmäßigkeiten möglichst anschaulich in den ausgewählten Problemen zu erkennen.

21.) Der Informatikunterricht bietet nach Erarbeitungsphasen auch Phasen der Metakognition, in denen zentrale Aspekte von zu erlernenden Kompetenzen reflektiert werden.

22.) Im Informatikunterricht wird auf eine angemessene Fachsprache geachtet. Schülerinnen und Schüler werden zu regelmäßiger, sorgfältiger und selbstständiger Dokumentation der erarbeiteten Unterrichtsinhalte angehalten.

23.) Der Informatikunterricht ist in seinen Anforderungen und im Hinblick auf die zu erreichenden Kompetenzen und deren Teilziele für die Schülerinnen und Schüler transparent.

24.) Im Informatikunterricht werden Diagnoseinstrumente zur Feststellung des jeweiligen Kompetenzstandes der Schülerinnen und Schüler durch die Lehrkraft, aber auch durch den Lernenden selbst eingesetzt.

25.) Der Informatikunterricht bietet immer wieder auch Phasen der Übung und des Transfers auf neue Aufgaben und Problemstellungen.

26.) Der Informatikunterricht bietet die Gelegenheit zum regelmäßigen wiederholenden Üben sowie zu selbstständigem Aufarbeiten von Unterrichtsinhalten.

2.3 Grundsätze der Leistungsbewertung und Leistungsrückmeldung

Zur Leistungsbewertung liegt ein Konzept der Fachschaft Informatik vor.

2.4 Lehr- und Lernmittel

Es wurde bisher kein Lehrwerk in der Sekundarstufe II eingeführt.

Die Schülerinnen und Schüler arbeiten die im Unterricht behandelten Inhalte in häuslicher Arbeit nach.

In der Sekundarstufe I werden Logo, der JavaHamster, HTML und Visual Basic eingesetzt. Diese Sprachen oder Umgebungen können vom Lehrer durch gleichwertige Sprachen oder Umgebungen ersetzt werden, die es ermöglichen die gleichen Ziele zu erreichen. Dabei ist darauf zu achten, dass vorrangig freie, plattformunabhängige Software eingesetzt wird, um ein häusliches Arbeiten für die Schüler zu erleichtern. In der Sekundarstufe II wird als Programmiersprache Java eingesetzt. Die Programmierumgebungen sind, unter den oben genannten Gesichtspunkten, vom Lehrer frei wählbar. Bisher sind Geany, BlueJ und Eclipse im Einsatz.

Unterstützende Materialien sind z. B. über die angegebenen Links bei den konkretisierten Unterrichtsvorhaben des MSW angegeben. Diese findet man unter:

http://www.standardsicherung.schulministerium.nrw.de/lehrplaene/lehrplannavigator-s-ii/

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

Durch die unterschiedliche Belegung von Fächern können Schülerinnen und Schüler Aspekte aus anderen Kursen mit in den Informatikunterricht einfließen lassen. Es wird Wert darauf gelegt, dass in bestimmten Fragestellungen die Expertise einzelner Schülerinnen und Schüler gesucht wird, die aus einem von ihnen belegten Fach genauere Kenntnisse mitbringen und den Unterricht dadurch bereichern.

Vorbereitung auf die Erstellung der Facharbeit

Um eine einheitliche Grundlage für die Erstellung und Bewertung der Facharbeiten im zweiten Halbjahr der Jahrgangsstufe Q1 zu gewährleisten, finden im Vorfeld jeweils zum Jahresende der Q1.1 Methodentage zur Facharbeit statt, in deren Rahmen Schulungen durch Fachlehrer stattfinden.

4 Qualitätssicherung und Evaluation

Evaluation des schulinternen Curriculums

Das schulinterne Curriculum stellt keine starre Größe dar, sondern ist als „lebendes Dokument“ zu betrachten. Dementsprechend werden die Inhalte stetig überprüft, um ggf. Modifikationen vornehmen zu können. Die Fachkonferenz trägt durch diesen Prozess zur Qualitätsentwicklung und damit zur Qualitätssicherung des Faches Informatik bei.
Die Evaluation erfolgt jährlich. Zu Schuljahresbeginn werden die Erfahrungen des vergangenen Schuljahres in der Fachschaft gesammelt, bewertet und eventuell notwendige Konsequenzen und Handlungsschwerpunkte formuliert. Folgendes Raster kann dabei als Hilfe dienen

Kriterien		**Ist-Zustand Auffälligkeiten**	**Änderungen/ Konsequenzen/ Perspektivplanung**	**Wer** (Verantwortlich)	**Bis wann** (Zeitrahmen)
Funktionen					
Fachvorsitz					
Stellvertreter					
Sonstige Funktionen (im Rahmen der schulprogrammatischen fächerübergreifenden Schwerpunkte)					
Ressourcen					
Personell	Fachlehrer/in				
	Lerngruppen				
	Lerngruppengröße				
	…				
räumlich	Fachraum				

	Bibliothek				
	…				
materiell / sachlich	Lehrwerke				
	Fachzeitschriften				
	…				
zeitlich	Abstände Fachteamarbeit				
	Dauer Fachteamarbeit				
	…				
Unterrichtsvorhaben					
Leistungsbewertung/ Einzelinstrumente					
Leistungsbewertung/Grundsätze					