Schulinterner Lehrplan zur

Informatik

in der gymnasialen Oberstufe

Städtisches Gymnasium Bad Laasphe

Version 1.6180

vom 25. Februar 2016

Inhaltsverzeichnis

1 Die Informatik-Fachgruppe

Beim Städtischen Gymnasium in Bad Laasphe handelt es sich um eine drei- bis vierzügige Schule in der Kernstadt von Bad Laasphe mit zurzeit ca. 700 Schülern und 60 Lehrern. Das zweite Gymnasium in Bad Laasphe ist eine Privatschule. Das Einzugsgebiet der Schule umfasst einen großen ländlich geprägten Teil des Kreises Siegen-Wittgenstein und einen beträchtlichen Teil jenseits der naheliegenden Landesgrenze zu Hessen. Etwa ein Drittel der Schülerschaft stammt aus Hessen.

Das Fach Informatik wird am Städtischen Gymnasium ab der Jahrgangsstufe 8 im Wahlpflichtbereich II (WP II) zweistündig unterrichtet. 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.

In der Jahrgangsstufe 7 wird ein für alle verpflichtender Kurs zum Umgang mit informatischen Systemen durchgeführt, ein sogenannter *ITG-Kurs* (Informationstechnische Grundbildung), der jedoch nicht unmittelbar dem Fach Informatik zuzuordnen ist.

In der Sekundarstufe II bietet das Städtische-Gymnasium Bad Laasphe in allen Jahrgangsstufen jeweils einen Grundkurs 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. In der Einführungsphase kommt dabei zusätzlich eine didaktische Bibliothek zum Einsatz, welche das Erstellen von graphischen 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 Informatik-Fachschaft des Städtischen Gymnasiums aus zwei Lehrkräften, denen zwei Computerräume mit 17 bzw. 16 Computerarbeitsplätzen und ein Selbstlernzentrum mit 12 Plätzen zur Verfügung stehen. Alle Arbeitsplätze sind an das schulinterne Rechnernetz angeschlossen, so dass Schüler über einen individuell gestaltbaren Zugang zum zentralen Server der Schule alle Arbeitsplätze der drei Räume zum Zugriff auf ihre eigenen Daten, zur Recherche im Internet oder zur Bearbeitung schulischer Aufgaben verwenden können.

Der Unterricht erfolgt im 60-Minuten-Takt. Grundkurse werden in drei Quartalen zweistündig und in einem Quartal dreistündig unterrichtet.

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, Schülern Lerngelegenheiten zu ermöglichen, so dass alle Kompetenzerwartungen des Kernlehrplans von ihnen erfüllt werden können.

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.

Die aufgeführten Beispiele und Materialien haben empfehlenden Charakter.

2.2 Übersicht zu den Unterrichtsvorhaben

Vorhaben	Zeit in h	Thema
Ef-I	5	Einführung in die Nutzung von Informatiksystemen und in grundlegende Begrifflichkeiten
Ef-II	10	Grundlagen der objektorientierten Analyse, Modellierung und Implementierung anhand von statischen Graphikszenen
Ef-III	14	Grundlegung der objektorientierten Programmierung und algorithmischer Grundstrukturen in Java anhand von einfachen Animationen
Ef-IV	14	Modellierung und Implementierung von Klassen- und Objektbeziehungen anhand von graphischen Spielen und Simulationen
Ef-V	5	Such- und Sortieralgorithmen anhand kontextbezogener Beispiele
Ef-VI	8	Geschichte der digitalen Datenverarbeitung und die Grundlagen des Datenschutzes
Ef-Gesamt	**56**	
Q1-I	6	Wiederholung der objektorientierten Modellierung und Programmierung anhand einer kontextbezogenen Problemstellung
Q1-II	15	Modellierung und Implementierung von Anwendungen mit dynamischen, linearen Datenstrukturen
Q1-III	12	Suchen und Sortieren auf linearen Datenstrukturen
Q1-IV	15	Modellierung und Nutzung von relationalen Datenbanken in Anwendungskontexten
Q1-V	8	Sicherheit und Datenschutz in Netzstrukturen
Q1-Gesamt	**56**	
Q2-I	18	Modellierung und Implementierung von Anwendungen mit dynamischen, nichtlinearen Datenstrukturen
Q2-II	15	Endliche Automaten und formale Sprachen
Q2-III	9	Prinzipielle Arbeitsweise eines Computers und Grenzen der Automatisierbarkeit
Q2-IV		Wiederholung und Vertiefung ausgewählter Kompetenzen und Inhalte des ersten Jahrs der Qualifikationsphase
Q2-Gesamt	**42**	

3 Einführungsphase

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

Die Schüler

- 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).

3.1 Unterrichtsvorhaben Ef-I

Einführung in die Nutzung von Informatiksystemen und in grundlegende Begrifflichkeiten

• Leitfragen: – Womit beschäftigt sich die Wissenschaft der Informatik? – Wie kann die in der Schule vorhandene informatische Ausstattung genutzt werden? • Zentrale Kompetenzen: – Argumentieren – Darstellen und Interpretieren • Zeitbedarf: 5 Stunden	• Inhaltsfelder: – Informatiksysteme – Informatik, Mensch und Gesellschaft • Inhaltliche Schwerpunkte: – Einzelrechner – Dateisystem – Internet – Einsatz von Informatiksystemen

Das erste Unterrichtsvorhaben stellt eine allgemeine Einführung in das Fach Informatik dar. Dabei ist zu berücksichtigen, dass für manche Schülerinnen und Schüler in der Einführungsphase der erste Kontakt mit dem Unterrichtsfach Informatik stattfindet, so dass zu Beginn Grundlagen des Fachs behandelt werden müssen.

Zunächst wird auf den Begriff der Information eingegangen und die Möglichkeit der Kodierung in Form von Daten thematisiert. Anschließend wird auf die Übertragung von Daten im Sinne des Sender-Empfänger-Modells eingegangen. Dabei wird eine überblickartige Vorstellung der Kommunikation von Rechnern in Netzwerken erarbeitet.

Des Weiteren soll der grundlegende Aufbau eines Rechnersystems im Sinne der Von-Neumann-Architektur erarbeitet werden und mit dem grundlegenden Prinzip der Datenverarbeitung (Eingabe-Verarbeitung-Ausgabe) in Beziehung gesetzt werden.

Bei der Beschäftigung mit Datenkodierung, Datenübermittlung und Datenverarbeitung ist jeweils ein Bezug zur konkreten Nutzung der informatischen Ausstattung der Schule herzustellen. So wird in die verantwortungsvolle Nutzung dieser Systeme eingeführt.

Unterrichtssequenzen:

1. **Information, deren Kodierung und Speicherung**
   (a) Informatik als Wissenschaft der Verarbeitung von Informationen
   (b) Darstellung von Informationen in Schrift, Bild und Ton
   (c) Speichern von Daten mit informatischen Systemen am Beispiel der Schulrechner
   (d) Vereinbarung von Richtlinien zur Datenspeicherung auf den Schulrechnern (z.B. Ordnerstruktur, Dateibezeichner usw.)

   Beispiele:

   (a) Textkodierung: Kodierung und Dekodierung von Texten mit unbekannten Zeichensätzen (z.B. Wingdings)
   (b) Bildkodierung: Kodierung von Bildinformationen in Raster- und Vektorgrafiken

2. **Informations- und Datenübermittlung in Netzen**
   (a) Sender-Empfänger-Modell und seine Bedeutung für die Eindeutigkeit von Kommunikation

(b) Informatische Kommunikation in Rechnernetzen am Beispiel des Schulnetzwerks (z.B. Benutzeranmeldung, Netzwerkordner, Zugriffsrechte, Client-Server)

(c) Grundlagen der technischen Umsetzung von Rechnerkommunikation am Beispiel des Internets (z.B. Netzwerkadresse, Paketvermittlung, Protokoll)

(d) Richtlinien zum verantwortungsvollen Umgang mit dem Internet

Beispiel: Rollenspiel zur Paketvermittlung im Internet

Schüler übernehmen die Rollen von Clients und Routern. Sie schicken spielerisch Informationen auf Karten von einem Schüler-Client zum anderen. Jeder Schüler hat eine Adresse, jeder Router darüber hinaus eine Routingtabelle. Mit Hilfe der Tabelle und einem Würfel wird entschieden, wie ein Paket weiter vermittelt wird.

3. **Aufbau informatischer Systeme**

(a) Identifikation typischer Komponenten informatischer Systeme und anschließende Beschränkung auf das Wesentliche, Herleitung der Von-Neumann-Architektur

(b) Identifikation des EVA-Prinzips (Eingabe-Verarbeitung-Ausgabe) als Prinzip der Verarbeitung von Daten und Grundlage der Von-Neumann-Architektur

Material: Demonstrationshardware

Beispiel: Durch Demontage eines Demonstrationsrechners entdecken Schüler die verschiedenen Hardwarekomponenten eines Informatiksystems. Als Demonstrationsrechner bietet sich ein ausrangierter Schulrechner an.

Die Schüler

- beschreiben und erläutern den Aufbau und die Arbeitsweise singulärer Rechner am Beispiel der Von-Neumann-Architektur (A),
- nutzen die im Unterricht eingesetzten Informatiksysteme selbstständig, sicher, zielführend und verantwortungsbewusst (D),
- nutzen das Internet zur Recherche, zum Datenaustausch und zur Kommunikation (K).

3.2 Unterrichtsvorhaben Ef-II

Grundlagen der objektorientierten Analyse, Modellierung und Implementierung anhand von statischen Graphikszenen

• Leitfrage: Wie lassen sich Gegenstandsbereiche informatisch modellieren und im Sinne einer Simulation informatisch realisieren? • Zentrale Kompetenzen: – Modellieren – Implementieren – Darstellen und Interpretieren • Zeitbedarf: 10 Stunden	• Inhaltsfelder: – Daten und ihre Strukturierung – Formale Sprachen und Automaten • Inhaltliche Schwerpunkte: – Objekte und Klassen – Syntax und Semantik einer Programmiersprache

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

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

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

Da bei der Umsetzung dieser ersten Projekte konsequent auf die Verwendung von Kontrollstrukturen verzichtet wird und der Quellcode aus einer rein linearen Sequenz besteht, ist auf diese Weise eine Fokussierung auf die Grundlagen der Objektorientierung möglich, ohne dass algorithmische Probleme ablenken. Natürlich kann die Arbeit an diesen Projekten unmittelbar zum nächsten Unterrichtsvorhaben führen. Dort stehen unter anderem Kontrollstrukturen im Mittelpunkt.

Material: Stifte-und-Mäuse-Umgebung: http://www.mg-werl.de/sum/

Material: GLOOP-Umgebung: http://www.bezreg-duesseldorf.nrw.de/lerntreffs/informatik/structure/material/sek2/einfuehrungen/gloop.php

Unterrichtssequenzen:

1. **Identifikation von Objekten**
   (a) Am Beispiel eines lebensweltnahen Beispiels werden Objekte im Sinne der Objektorientierten Modellierung eingeführt.
   (b) Objekte werden mit Objektkarten visualisiert und mit sinnvollen Attributen und Fähigkeiten, d.h. Methoden versehen.
   (c) Manche Objekte sind prinzipiell typgleich und werden so zu einer Objektsorte bzw. Objektklasse zusammengefasst.
   (d) Vertiefung: Modellierung weiterer Beispiele ähnlichen Musters
2. **Analyse von Klassen didaktischer Lernumgebungen**

(a) Objektorientierte Programmierung als modularisiertes Vorgehen (Entwicklung von Problemlösungen auf Grundlage vorhandener Klassen)

(b) Teilanalyse der Klassen der didaktischen Lernumgebungen („Stifte und Mäuse oder GLOOP

3. **Implementierung zwei- oder dreidimensionaler, statischer Szenen**

(a) Grundaufbau einer Java-Klasse

(b) Konzeption einer Szene mit Objekten

(c) Deklaration und Initialisierung von Objekten

(d) Methodenaufrufe mit Parameterübergabe zur Manipulation von Objekteigenschaften (z.B. Farbe, Position, usw.)

Die Schüler

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

3.3 Unterrichtsvorhaben Ef-III

Grundlegung der objektorientierten Programmierung und algorithmischer Grundstrukturen in Java anhand von einfachen Animationen

- Leitfrage: Wie lassen sich Animationen und Simulationen optischer Gegenstandsbereiche unter Berücksichtigung von Tastatureingaben realisieren?
- Zentrale Kompetenzen:
  - Argumentieren
  - Modellieren
  - Implementieren
- Zeitbedarf: 14 Stunden

- 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

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

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

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

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

Unterrichtssequenzen:

1. **Bewegungsanimationen am Beispiel einfacher graphischer Objekte**
   (a) Kontinuierliche Verschiebung eines Objekts mit Hilfe einer Schleife (While-Schleife)
   (b) Tastaturabfrage zur Realisierung einer Schleifenbedingung für eine Animationsschleife
   (c) Mehrstufige Animationen mit mehreren sequenziellen Schleifen
   (d) Berechnung von Abständen zwischen Objekten mit Hilfsvariablen
   (e) Meldungen zur Kollision zweier Objekte mit Hilfe von Abstandsberechnungen und Verzweigungen (IF-Anweisungen)
2. **Erstellen und Verwalten größerer Mengen einfacher graphischer Objekte**

(a) Erzeugung von Objekten mit Hilfe von Zählschleifen (FOR-Schleife)

(b) Verwaltung von Objekten in eindimensionalen Feldern (Arrays)

(c) Animation von Objekten, die in eindimensionalen Feldern (Arrays) verwaltet werden

(d) Vertiefung: Verschiedene Feldbeispiele

3. **Modellierung und Animation komplexerer graphisch repräsentierbarer Objekte**

(a) Modellierung eines Simulationsprogramms mit eigenen Klassen, die sich selbst mit Hilfe von einfachen Objekten zeigen mit Hilfe eines Implementationsdiagramms

(b) Implementierung eigener Methoden mit und ohne Parameterübergabe

(c) Realisierung von Zustandsvariablen

(d) Thematisierung des Geheimnisprinzips und des Autonomitätsprinzips von Objekten

(e) Animation mit Hilfe des Aufrufs von selbstimplementierten Methoden

(f) Vertiefung: Weitere Projekte

Die Schüler

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

3.4 Unterrichtsvorhaben Ef-IV

Modellierung und Implementierung von Klassen- und Objektbeziehungen anhand von graphischen Spielen und Simulationen

• Leitfrage: Wie lassen sich komplexere Datenflüsse und Beziehungen zwischen Objekten und Klassen realisieren? • Zentrale Kompetenzen: – Argumentieren – Modellieren – Implementieren – Darstellen und Interpretieren • Zeitbedarf: 14 Stunden	• 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

Dieses Unterrichtsvorhaben beschäftigt sich im Schwerpunkt mit dem Aufbau komplexerer Objektbeziehungen. Während in vorangegangenen Unterrichtsvorhaben Objekte nur jeweils solchen Objekten Nachrichten schicken konnten, die sie selbst erstellt haben, soll in diesem Unterrichtsvorhaben diese hierarchische Struktur aufgebrochen werden.

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

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

Unterrichtssequenzen:

1. **Vertiefung des Referenzbegriffs und Einführung des Prinzips der dynamischen Referenzierung**
   (a) Einführung der Objektselektion mit der Maus
   (b) Einführung der Klasse Figur als Oberklasse der sichtbaren Objekte in „Stifte und Mäuse“ (alternativ: Klasse GLObjekt als Oberklasse aller sichtbaren Objekte in „GLOOP“)
   (c) Steuerung einfacher grafischer Objekte über eine Referenz *aktuell*, die jeweils durch eine Klickselektion mit der Maus auf ein neues Objekt gesetzt werden kann.
2. **Entwicklung eines Spiels mit der Notwendigkeit von Kollisionskontrollen zwischen zwei oder mehr graphischen Objekten**
   (a) Modellierung des Spiels ohne Berücksichtigung der Kollision mit Hilfe eines Implementationsdiagramms

(b) Dokumentation der Klassen des Projekts

(c) Implementierung eines Prototypen ohne Kollision

(d) Ergänzung einer Kollisionsabfrage durch zusätzliche Assoziationsbeziehungen in Diagramm, Dokumentation und Quellcode

(e) Verallgemeinerung der neuen Verwendung von Objektreferenzen

(f) Vertiefung: Entwicklung weiterer Spiele und Simulationen mit vergleichbarer Grundmodellierung

3. **Erarbeitung einer Simulation mit graphischen Objekten, die sich durch unterschiedliche Ergänzungen voneinander unterscheiden (Vererbung durch Spezialisierung ohne Überschreiben von Methoden)**

   (a) Analyse und Erläuterung einer Basisversion der graphischen Klasse

   (b) Realisierung von graphischen Erweiterungen zur Basisklasse mit und ohne Vererbung (Implementationsdiagramm und Quellcode)

   (c) Verallgemeinerung und Reflexion des Prinzips der Vererbung am Beispiel der Spezialisierung

4. **Entwicklung einer komplexeren Simulation mit graphischen Elementen, die unterschiedliche Animationen durchführen (Vererbung mit Überschreiben von Methoden)**

   (a) Analyse und Erläuterung einer einfachen graphischen Animationsklasse

   (b) Spezialisierung der Klasse zu Unterklassen mit verschiedenen Animationen durch Überschreiben der entsprechenden Animationsmethode

   (c) Reflexion des Prinzips der späten Bindung

   (d) Vertiefung: Entwicklung eines vergleichbaren Projekts mit einer (abstrakten) Oberklasse

Die Schüler

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

3.5 Unterrichtsvorhaben Ef-V

Such- und Sortieralgorithmen anhand kontextbezogener Beispiele

• Leitfragen: Wie können Objekte bzw. Daten effizient sortiert werden, so dass eine schnelle Suche möglich wird? • Zentrale Kompetenzen: – Argumentieren – Modellieren – Darstellen und Interpretieren • Zeitbedarf: 5 Stunden	• Inhaltsfelder: – Algorithmen • Inhaltliche Schwerpunkte: – Algorithmen zum Suchen und Sortieren – Analyse, Entwurf und Implementierung einfacher Algorithmen

Dieses Unterrichtsvorhaben beschäftigt sich mit der Erarbeitung von Such- und Sortieralgorithmen. Der Schwerpunkt des Vorhabens liegt dabei auf den Algorithmen selbst und nicht auf deren Implementierung in einer Programmiersprache, auf die in diesem Vorhaben vollständig verzichtet werden soll.

Zunächst erarbeiten die Schüler 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 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 Schüler sollen auf diese Weise das *Sortieren durch Vertauschen*, das *Sortieren durch Auswählen* und mindestens einen weiteren Sortieralgorithmus, kennen lernen.

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

Unterrichtssequenzen:

1. **Explorative Erarbeitung eines Sortierverfahrens**
   (a) Sortierprobleme im Kontext informatischer Systeme und im Alltag (z.B. Dateisortierung, Tabellenkalkulation, Telephonbuch, Bundesligatabelle, usw.)
   (b) Vergleich zweier Elemente als Grundlage eines Sortieralgorithmus
   (c) Erarbeitung eines Sortieralgorithmus durch die Schüler

   Beispiel: Sortieren mit Waage

   Die 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 können.

   Material: Sortierverfahren: http://www.matheprisma.uni-wuppertal.de/Module/Sortieren/index.htm

2. **Systematisierung von Algorithmen und Effizienzbetrachtungen**
   (a) Formulierung (falls selbst gefunden) oder Erläuterung von mehreren Algorithmen im Pseudocode (auf jeden Fall: Sortieren durch Vertauschen, Sortieren durch Auswählen)
   (b) Anwendung von Sortieralgorithmen auf verschiedene Beispiele
   (c) Bewertung von Algorithmen anhand der Anzahl der nötigen Vergleiche

(d) Variante des Sortierens durch Auswählen (Nutzung eines einzigen oder zweier Felder bzw. lediglich eines einzigen zusätzlichen Ablageplatzes oder mehrerer neuer Ablageplätze)

(e) Effizienzbetrachtungen an einem konkreten Beispiel bezüglich der Rechenzeit und des Speicherplatzbedarfs

(f) Analyse des weiteren Sortieralgorithmus (sofern nicht in Sequenz 1 und 2 bereits geschehen)

Beispiele: Sortieren durch Auswählen, Sortieren durch Vertauschen, Quicksort bzw. 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.

3. **Binäre Suche auf sortierten Daten**

(a) Suchaufgaben im Alltag und im Kontext informatischer Systeme

(b) Evtl. Simulationsspiel zum effizienten Suchen mit binärer Suche

(c) Effizienzbetrachtungen zur binären Suche

Beispiel: Simulationsspiel zur binären Suche nach Tischtennisbällen

Mehrere Tischtennisbälle sind nummeriert, sortiert und unter Bechern verdeckt. Mit Hilfe der binären Suche kann sehr schnell ein bestimmter Tischtennisball gefunden werden.

Material: Binäres Suchspiel: http://www.matheprisma.uni-wuppertal.de/Module/BinSuch/index.htm

Die Schüler

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

3.6 Unterrichtsvorhaben Ef-VI

Geschichte der digitalen Datenverarbeitung und die Grundlagen des Datenschutzes

• Leitfrage: Welche Entwicklung durchlief die moderne Datenverarbeitung und welche Auswirkungen ergeben sich insbesondere hinsichtlich neuer Anforderungen an den Datenschutz daraus? • Zentrale Kompetenzen: – Argumentieren – Darstellen und Interpretieren • Zeitbedarf: 8 Stunden	• Inhaltsfelder: – Informatik, Mensch und Gesellschaft – Informatiksysteme • Inhaltliche Schwerpunkte: – Wirkungen der Automatisierung – Geschichte der automatischen Datenverarbeitung – Digitalisierung

Das folgende Unterrichtsvorhaben stellt den Abschluss der Einführungsphase dar. Schüler sollen selbstständig informatische Themenbereiche aus dem Kontext der Geschichte der Datenverarbeitung und insbesondere den daraus sich ergebenen Fragen des Datenschutzes bearbeiten. Diese Themenbereiche werden in Kleingruppen bearbeitet und in Form von Plakatpräsentationen vorgestellt. Schüler 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.

Unterrichtssequenzen:

1. **Selbstständige Erarbeitung von Themen durch die Schüler**
   (a) Mögliche Themen zur Erarbeitung in Kleingruppen:
   - Eine kleine Geschichte der Digitalisierung: vom Morsen zum modernen Digitalcomputer
   - Eine kleine Geschichte der Kryptographie: von Caesar zur Enigma
   - Von Nullen, Einsen und mehr: Stellenwertsysteme und wie man mit ihnen rechnet
   - Kodieren von Texten und Bildern: ASCII, RGB und mehr
   - Auswirkungen der Digitalisierung: Veränderungen der Arbeitswelt und Datenschutz

   (b) Vorstellung und Diskussion durch Schüler

   Beispiel: Ausstellung zu informatischen Themen

   Die Schüler bereiten eine Ausstellung zu informatischen Themen vor. Dazu werden Stellwände und Plakate vorbereitet, die ggf. auch außerhalb des Informatikunterrichts in der Schule ausgestellt werden können.

2. **Vertiefung des Themas Datenschutz**
   (a) Erarbeitung grundlegender Begriffe des Datenschutzen
   (b) Problematisierung und Anknüpfung an die Lebenswelt der Schüler
   (c) Diskussion und Bewertung von Fallbeispielen aus dem Themenbereich „Datenschutz“

   Beispiel: Fallbeispiele aus dem aktuellen Tagesgeschehen: Die Schüler bearbeiten Fallbeispiele aus ihrer eigenen Erfahrungswelt oder der aktuellen Medienberichterstattung.

   Materialien: Materialblatt zum Bundesdatenschutzgesetz

Die 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).

Gesamtzahl der Stunden in der Einführungsphase: 56

4 Qualifikationsphase

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 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).

4.1 Unterrichtsvorhaben Q1-I

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 graphisch darstellen?
- Zentrale Kompetenzen:
  - Argumentieren
  - Modellieren
  - Implementieren
  - Darstellen und Interpretieren
- Zeitbedarf: 6 Stunden

- 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

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

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

Unterrichtssequenzen:

1. **Wiederholung und Erweiterung der objektorientierten Modellierung und Programmierung durch Analyse und Erweiterung eines kontextbezogenen Beispiels**
   (a) Analyse der Problemstellung
   (b) Analyse der Modellierung (Implementationsdiagramm)
   (c) Erweiterung der Modellierung im Implementationsdiagramm (Vererbung, abstrakte Klasse)
   (d) Kommunikation zwischen mindestens zwei Objekten (graphische Darstellung)
   (e) Dokumentation von Klassen
   (f) Implementierung der Anwendung oder von Teilen der Anwendung

   Beispiel: Tannenbaum

Ein Tannenbaum soll mit verschiedenen Arten von Schmuckstücken versehen werden, die durch unterschiedliche geometrische Objekte dargestellt werden. Es gibt Kugeln, Päckchen in der Form von Würfeln und Zuckerringe in Form von Toren.

Ein Prototyp, der bereits mit Kugeln geschmückt werden kann, kann zur Verfügung gestellt werden. Da alle Schmuckstücke über die Funktion des Auf- und Abschmückens verfügen sollen, liegt es nahe, dass entsprechende Methoden in einer gemeinsamen Oberklasse realisiert werden.

Die 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),
- 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 graphisch dar (D),
- dokumentieren Klassen (D),
- stellen die Kommunikation zwischen Objekten grafisch dar (D).

4.2 Unterrichtsvorhaben Q1-II

Modellierung und Implementierung von Anwendungen mit dynamischen, linearen Datenstrukturen

• Leitfrage: Wie können beliebig viele linear angeordnete Daten im Anwendungskontext verwaltet werden? • Zentrale Kompetenzen: – Argumentieren – Modellieren – Implementieren – Darstellen und Interpretieren • Zeitbedarf: 15 Stunden	• 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

Nach Analyse einer Problemstellung in einem geeigneten Anwendungskontext, in dem Daten nach dem First-In-First-Out-Prinzip verwaltet werden, werden der Aufbau von Schlangen am Beispiel dargestellt und die Operationen der Klasse Queue erläutert. Anschließend werden für die Anwendung notwendige Klassen modelliert und implementiert. Eine Klasse für eine den Anforderungen der Anwendung entsprechende Oberfläche sowie die Klasse Queue wird dabei von der Lehrkraft vorgegeben. Anschließend wird die Anwendung modifiziert, um den Umgang mit der Datenstruktur zu üben. Anhand einer Anwendung, in der Daten nach dem Last-In-First-Out-Prinzip verwaltet werden, werden Unterschiede zwischen den Datenstrukturen Schlange und Stapel erarbeitet. Um einfacher an Objekte zu gelangen, die zwischen anderen gespeichert sind, wird die Klasse List eingeführt und in einem Anwendungskontext verwendet. In mindestens einem weiteren Anwendungskontext wird die Verwaltung von Daten in Schlangen, Stapeln oder Listen vertieft. Modellierungen werden dabei in Entwurfs- und Implementationsdiagrammen dargestellt.

Unterrichtssequenzen:

1. **Die Datenstruktur Schlange im Anwendungskontext unter Nutzung der Klasse Queue**
   (a) Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen
   (b) Erarbeitung der Funktionalität der Klasse Queue
   (c) Modellierung und Implementierung der Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse Queue

   Beispiel: Patientenwarteschlange (jeder kennt seinen Nachfolger bzw. alternativ seinen Vorgänger)

   Sobald ein Patient in einer Arztpraxis eintrifft, werden sein Name und seine Krankenkasse erfasst. Die Verwaltung der Patientenwarteschlange geschieht über eine Klasse, die hier als Wartezimmer bezeichnet wird. Wesentliche Operationen sind das „Hinzufügen“ eines Patienten und das „Entfernen“ eines Patienten, wenn er zur Behandlung gerufen wird.

   Die Simulationsanwendung stellt eine GUI zur Verfügung, legt ein Wartezimmer an und steuert die Abläufe. Wesentlicher Aspekt des Projektes ist die Modellierung des Wartezimmers mit Hilfe der Klasse Queue.

   Anschließend wird der Funktionsumfang der Anwendung erweitert: Patienten können sich zusätzlich in die Warteschlange zum Blutdruckmessen einreihen. Objekte werden von zwei Schlangen verwaltet.

2. **Die Datenstruktur Stapel im Anwendungskontext unter Nutzung der Klasse Stack**

(a) Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen

(b) Erarbeitung der Funktionalität der Klasse Stack

(c) Modellierung und Implementierung der Anwendung unter Verwendung eines oder mehrerer Objekte der Klasse Stack

Beispiele:

(a) Heftstapel
In einem Heftstapel soll das Heft einer Schülerin gefunden werden.

(b) Kisten stapeln
In einem Stapel nummerierter Kisten soll eine bestimmte Kiste gefunden und an einen Kunden geliefert werden. Dazu müssen Kisten auf verschiedene Stapel gestapelt und wieder zurückgestellt werden.

3. **Die Datenstruktur lineare Liste im Anwendungskontext unter Nutzung der Klasse List**

(a) Erarbeitung der Vorteile der Klasse List im Gegensatz zu den bereits bekannten linearen Strukturen

(b) Modellierung und Implementierung einer kontextbezogenen Anwendung unter Verwendung der Klasse List.

Beispiel: Abfahrtslauf

Bei einem Abfahrtslauf kommen die Skifahrer nacheinander an und werden nach ihrer Zeit in eine Rangliste eingeordnet. Diese Rangliste wird in einer Anzeige ausgegeben. Ankommende Abfahrer müssen an jeder Stelle der Struktur, nicht nur am Ende oder Anfang eingefügt werden können.

4. **Vertiefung - Anwendungen von Listen, Stapeln oder Schlangen in mindestens einem weiteren Kontext**

Beispiele:

(a) Skispringen
Ein Skispringen hat folgenden Ablauf: Nach dem Sprung erhält der Springer eine Punktzahl und wird nach dieser Punktzahl in eine Rangliste eingeordnet. Die besten 30 Springer qualifizieren sich für den zweiten Durchgang. Sie starten in umgekehrter Reihenfolge gegenüber der Platzierung auf der Rangliste. Nach dem Sprung erhält der Springer wiederum eine Punktzahl und wird nach der Gesamtpunktzahl aus beiden Durchgängen in die endgültige Rangliste eingeordnet.

(b) Terme in Postfix-Notation
Die sog. UPN (Umgekehrte-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.

(c) Rangierbahnhof
Auf einem Güterbahnhof gibt es drei Gleise, die nur zu einer Seite offen sind. Waggons können also von einer Seite auf das Gleis fahren und nur rückwärts wieder hinausfahren. Die Waggons tragen Nummern, wobei die Nummer jedoch erst eingesehen werden kann, wenn der Waggon der vorderste an der offenen Gleisseite ist. (Zwischen den Waggons herumzuturnen, um die anderen Waggonnummern zu lesen, wäre zu gefährlich.) Zunächst stehen alle Waggons unsortiert auf einem Gleis. Ziel ist es, alle Waggons in ein anderes Gleis zu fahren, so dass dort die Nummern der Waggons vom Gleisende aus aufsteigend in richtiger Reihenfolge sind. Zusätzlich zu diesen beiden Gleisen gibt es ein Abstellgleis, das zum Rangieren benutzt werden kann.

(d) Autos an einer Ampel zur Zufahrtsregelung
Es soll eine Ampel zur Zufahrtsregelung in Java simuliert werden. An einem geradlinigen, senkrecht von unten nach oben verlaufenden Straßenstück, das von Autos nur einspurig in eine Richtung befahren werden kann, ist ein Haltepunkt markiert, an dem die Ampel steht. Bei einem Klick auf eine Schaltfläche mit der Aufschrift „Heranfahren“ soll ein neues Auto an den Haltepunkt heranfahren bzw. bis an das letzte Auto, das vor dem Haltepunkt wartet. Grünphasen der Ampel werden durch einen Klick auf eine Schaltfläche mit der Aufschrift „Weiterfahren“ simuliert. In jeder Grünphase darf jeweils nur ein Auto weiterfahren. Die anderen Autos rücken nach.

Material: Lineare Datenstrukturen: http://www.matheprisma.uni-wuppertal.de/Module/LinDatSt/index.htm

Die 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),
- interpretieren Fehlermeldungen und korrigieren den Quellcode (I),
- testen Programme systematisch anhand von Beispielen (I),
- stellen lineare und nichtlineare Strukturen graphisch dar und erläutern ihren Aufbau (D).

4.3 Unterrichtsvorhaben Q1-III

Suchen und Sortieren auf linearen Datenstrukturen

• Leitfrage: Wie kann man gespeicherte Informationen günstig (wieder-)finden? • Zentrale Kompetenzen: – Argumentieren – Modellieren – Implementieren – Darstellen und Interpretieren • Zeitbedarf: 12 Stunden	• Inhaltsfelder: – Algorithmen – Formale Sprachen und Automaten • Inhaltliche Schwerpunkte: – Analyse, Entwurf und Implementierung von Algorithmen – Algorithmen in ausgewählten informatischen Kontexten – Syntax und Semantik einer Programmiersprache

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

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

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

Unterrichtssequenzen:

1. **Suchen von Daten in Listen und Arrays**
   (a) Lineare Suche in Listen und in Arrays
   (b) Binäre Suche in Arrays als Beispiel für rekursives Problemlösen
   (c) Untersuchung der beiden Suchverfahren hinsichtlich ihrer Effizienz (Laufzeitverhalten, Speicherbedarf)

   Beispiele

   (a) Karteiverwaltung
   Für ein Adressverwaltungsprogramm soll eine Methode zum Suchen einer Adresse geschrieben werden.

   (b) Bundesjugendspiele
   Die Teilnehmer an Bundesjugendspielen nehmen an drei Disziplinen teil und erreichen dort Punktzahlen. Diese werden in einer Wettkampfkarte eingetragen und an das Wettkampfbüro gegeben. Zur Vereinfachung sollte sich das Modell auf die drei Disziplinen „Lauf“, „Sprung“ und „Wurf“ beschränken.
   Im Wettkampfbüro wird das Ergebnis erstellt. Das Programm soll dafür zunächst den Besten einer Disziplin heraussuchen können und später das gesamte Ergebnis nach gewissen Kriterien sortieren können.

2. **Sortieren in Listen und Arrays - Entwicklung und Implementierung von iterativen und rekursiven Sortierverfahren**

(a) Entwicklung und Implementierung eines einfachen Sortierverfahrens für eine Liste

(b) Implementierung eines einfachen Sortierverfahrens für ein Feld

(c) Entwicklung eines rekursiven Sortierverfahren für ein Feld (z.B. Sortieren durch Mischen)

Beispiele

(a) Karteiverwaltung (s.o.)

(b) Bundesjugendspiele (s.o.)

3. **Untersuchung der Effizienz der Sortierverfahren „Sortieren durch direktes Einfügen“ und „Quicksort“ auf linearen Listen**

(a) Graphische Veranschaulichung der Sortierverfahren

(b) Untersuchung der Anzahl der Vergleichsoperationen und des Speicherbedarf bei beiden Sortierverfahren

(c) Beurteilung der Effizienz der beiden Sortierverfahren

Beispiel 1: Karteiverwaltung (s.o.)

Beispiel 2: Bundesjugendspiele (s.o.)

Die Schüler

- analysieren und erläutern Algorithmen und Programme (A),
- beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A),
- beurteilen die Effizienz von Algorithmen unter Berücksichtigung des Speicherbedarfs und der Zahl der Operationen (A),
- entwickeln iterative und rekursive Algorithmen unter Nutzung der Strategien „Modularisierung“ und „Teilen und Herrschen“ (M),
- modifizieren Algorithmen und Programme (I),
- implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I),
- implementieren und erläutern iterative und rekursive Such- und Sortierverfahren (I),
- nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I),
- interpretieren Fehlermeldungen und korrigieren den Quellcode (I),
- testen Programme systematisch anhand von Beispielen (I),
- stellen iterative und rekursive Algorithmen umgangssprachlich und graphisch dar (D).

4.4 Unterrichtsvorhaben Q1-IV

Modellierung und Nutzung von relationalen Datenbanken in Anwendungskontexten

• Leitfragen: – Wie können Fragestellungen mit Hilfe einer Datenbank beantwortet werden? – Wie entwickelt man selbst eine Datenbank für einen Anwendungskontext? • Zentrale Kompetenzen: – Argumentieren – Modellieren – Implementieren – Darstellen und Interpretieren • Zeitbedarf: 15 Stunden	• 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

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

Unterrichtssequenzen:

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 mehreren Tabellen 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

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)

Beispiele:

(a) Fahrradverleih
Der Fahrradverleih „BTR (BikesToRent)“ verleiht unterschiedliche Typen von Fahrrädern diverser Firmen an seine Kunden. Die Kunden sind bei „BTR“ registriert (Name, Adresse, Telefon). „BTR“ kennt von den Fahrradfirmen den Namen und die Telefonnummer. Kunden von „BTR“ können CityBikes, Treckingräder und Mountainbikes ausleihen.

(b) Reederei
Die Datenverwaltung einer Reederei soll in einem Datenbanksystem umgesetzt werden. Ausgehend von der Modellierung soll mit Hilfe eines ER-Modells und eines Datenbankschemas dieser erste Entwurf normalisiert und in einem Datenbanksystem umgesetzt werden. Es schließen sich diverse SQL-Abfragen an, wobei auf die Relationenalgebra eingegangen wird.

(c) Buchungssystem
In dem Online-Buchungssystem einer Schule können die Lehrer Medienräume, Beamer, Laptops, Kameras, usw. für einen bestimmten Zeitpunkt buchen, der durch Datum und die Schulstunde festgelegt ist.
Dazu ist die Datenbank zu modellieren, ggf. zu normalisieren und im Datenbanksystem umzusetzen. Weiter sollen sinnvolle Abfragen entwickelt werden. Unter http://mrbs.sourceforge.net/ (abgerufen: 30.03. 2014) findet man ein freies Online-Buchungssystem inklusive Demo, an Hand derer man erläutern kann, worum es in dem Projekt geht.

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

3. *Material:* http://www.oberstufeninformatik.de/Datenbanken/index.html

Die 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 graphisch dar (D),
- überprüfen Datenbankschemata auf vorgegebene Normalisierungseigenschaften (D).

4.5 Unterrichtsvorhaben Q1-V

Sicherheit und Datenschutz in Netzstrukturen

• Leitfragen: – Wie werden Daten in Netzwerken übermittelt? – Was sollte man in Bezug auf die Sicherheit beachten? • Zentrale Kompetenzen: – Argumentieren – Darstellen und Interpretieren • Zeitbedarf: 8 Stunden	• Inhaltsfelder: – Informatiksysteme – Informatik, Mensch und Gesellschaft • Inhaltliche Schwerpunkte: – Einzelrechner und Rechnernetzwerke – Sicherheit – Nutzung von Informatiksystemen, Wirkungen der Automatisierung

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 kryptographische Verfahren analysiert und erläutert. Fallbeispiele zur Datenschutzproblematik und zum Urheberrecht runden das Unterrichtsvorhaben ab.

Unterrichtssequenzen:

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

   Material: Cäsar-Verfahren: http://www.matheprisma.uni-wuppertal.de/Module/Caesar/index.htm

   Material: RSA-Verfahren: http://www.matheprisma.uni-wuppertal.de/Module/RSA/index.htm

2. **Fallbeispiele zur Datenschutzproblematik und zum Urheberrecht**

Die 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).

Gesamtzahl der Stunden in der Qualifikationsphase 1: 56

4.6 Unterrichtsvorhaben Q2-I

Modellierung und Implementierung von Anwendungen mit dynamischen, nichtlinearen Datenstrukturen

- Leitfragen:
  - Wie können Daten im Anwendungskontext mit Hilfe binärer Baumstrukturen verwaltet werden?
  - Wie kann dabei der rekursive Aufbau der Baumstruktur genutzt werden?
  - Welche Vor- und Nachteile haben Suchbäume für die geordnete Verwaltung von Daten?
- Zentrale Kompetenzen:
  - Argumentieren
  - Modellieren
  - Implementieren
  - Darstellen und Interpretieren
- Zeitbedarf: 18 Stunden

- 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

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 graphischer Darstellungen von Binärbäumen erläutert.

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

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

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

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

Unterrichtssequenzen:

1. **Analyse von Baumstrukturen in verschiedenen Kontexten**

(a) Grundlegende Begriffe (Grad, Tiefe, Höhe, Blatt, Inhalt, Teilbaum, Ebene, Vollständigkeit)

(b) Aufbau und Darstellung von binären Bäumen anhand von Baumstrukturen in verschiedenen Kontexten

Beispiele:

(a) Termbaum: Der Aufbau von Termen wird mit Hilfe von binären Baumstrukturen verdeutlicht.

(b) Ahnenbaum: Die binäre Baumstruktur ergibt sich daraus, dass jede Person genau einen Vater und eine Mutter hat.

(c) 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.)

(d) Entscheidungsbäume
Um eine Entscheidung zu treffen, werden mehrere Fragen mit ja oder nein beantwortet. Die Fragen, die möglich sind, wenn die Antwort auf eine Frage mit „ja“ beantwortet wird, befinden sich im linken Teilbaum, die Fragen, die möglich sind, wenn die Antwort „nein“ lautet, stehen im rechten Teilbaum.

(e) Codierungsbäume für Codierungen, deren Alphabet aus genau zwei Zeichen besteht
Morse hat Buchstaben als Folge von Punkten und Strichen codiert. Diese Codierungen können in einem Binärbaum dargestellt werden, so dass ein Übergang zum linken Teilbaum einem Punkt und ein Übergang zum rechten Teilbaum einem Strich entspricht. Wenn man im Gesamtbaum startet und durch Übergänge zu linken oder rechten Teilbäumen einen Pfad zum gewünschten Buchstaben sucht, erhält man die Morsecodierung des Buchstabens.

2. **Die Datenstruktur Binärbaum im Anwendungskontext unter Nutzung der Klasse BinaryTree**

(a) Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen im Anwendungskontext

(b) Modellierung eines Entwurfsdiagramms und Entwicklung eines Implementationsdiagramms

(c) Erarbeitung der Klasse BinaryTree und beispielhafte Anwendung der Operationen

(d) Implementierung der Anwendung oder von Teilen der Anwendung

(e) Traversierung eines Binärbaums im Pre-, In- und Postorderdurchlauf

Beispiel: Informatikerbaum als binärer Baum

In einem „binären Baum“ werden die Namen und die Geburtsdaten von Informatikern lexikographisch geordnet abgespeichert. Alle Namen, die nach dieser Ordnung vor dem Namen im aktuellen Teilbaum stehen, sind in dessen linkem Teilbaum, alle die nach dem Namen im aktuellen Teilbaum stehen, sind in dessen rechtem Teilbaum. (Dies gilt für alle Teilbäume.)

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*

3. **Die Datenstruktur binärer Suchbaum im Anwendungskontext unter Verwendung der Klasse BinarySearchTree**

(a) Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen

(b) Modellierung eines Entwurfsdiagramms und Entwicklung eines Implementationsdiagramms, graphische Darstellung eines binären Suchbaums und Erarbeitung der Struktureigenschaften

(c) Erarbeitung der Klasse BinarySearchTree und Einführung des Interface Item zur Realisierung einer geeigneten Ordnungsrelation

(d) Implementierung der Anwendung oder von Teilen der Anwendung inklusive einer sortierten Ausgabe des Baums

Beispiel: Informatikerbaum als Suchbaum (s. o.)

4. **Übung und Vertiefungen der Verwendung von Binärbäumen oder binären Suchbäumen anhand weiterer Problemstellungen**

   Beispiele:

   (a) *Codierungsbäume (s.o.) oder Huffman-Codierung*

   (b) *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.

   (c) *Entscheidungsbäume (s.o.)*

   (d) *Termbaum (s.o.)*

   (e) *Ahnenbaum (s.o.)*

Die 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),
- 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 graphisch dar und erläutern ihren Aufbau (D),
- stellen iterative und rekursive Algorithmen umgangssprachlich und graphisch dar (D).

4.7 Unterrichtsvorhaben Q2-II

Endliche Automaten und formale Sprachen

• Leitfragen: – Wie kann man (endliche) Automaten genau beschreiben? – Wie können endliche Automaten (in alltäglichen Kontexten oder zu informatischen Problemstellungen) modelliert werden? – Wie können Sprachen durch Grammatiken beschrieben werden? – Welche Zusammenhänge gibt es zwischen formalen Sprachen, endlichen Automaten und regulären Grammatiken? • Zentrale Kompetenzen: – Argumentieren – Modellieren – Darstellen und Interpretieren • Zeitbedarf: 15 Stunden	• Inhaltsfelder: – Endliche Automaten und formale Sprachen • Inhaltliche Schwerpunkte: – Endliche Automaten – Grammatiken regulärer Sprachen – Möglichkeiten und Grenzen von Automaten und formalen Sprachen

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.

Unterrichtssequenzen:

1. **Endliche Automaten**
   (a) Vom Automaten in den Schülern bekannten Kontexten zur formalen Beschreibung eines endlichen Automaten
   (b) Untersuchung, Darstellung und Entwicklung endlicher Automaten

   Beispiele: Cola-Automat, Geldspielautomat, Roboter, Zustandsänderung eines Objekts „Auto“, Akzeptor für bestimmte Zahlen, Akzeptor für Teilwörter in längeren Zeichenketten, Akzeptor für Terme

   Material: http://www.oberstufeninformatik.de/theorie/index.html

2. **Untersuchung und Entwicklung von Grammatiken regulärer Sprachen**
   (a) Erarbeitung der formalen Darstellung regulärer Grammatiken

(b) Untersuchung, Modifikation und Entwicklung von Grammatiken

(c) Entwicklung von endlichen Automaten zum Erkennen regulärer Sprachen, die durch Grammatiken gegeben werden

(d) Entwicklung regulärer Grammatiken zu endlichen Automaten

Beispiele: reguläre Grammatik für Wörter mit ungerader Parität, Grammatik für Wörter, die bestimmte Zahlen repräsentieren, Satzgliederungsgrammatik

3. **Grenzen endlicher Automaten**

Beispiele: Klammerausdrücke, $a^n b^n$ im Vergleich zu $(ab)^n$

Die Schüler

- analysieren und erläutern die Eigenschaften endlicher Automaten einschließlich ihres Verhaltens auf bestimmte Eingaben (A),
- analysieren und erläutern Grammatiken regulärer Sprachen (A),
- zeigen die Grenzen endlicher Automaten und regulärer Grammatiken im Anwendungszusammenhang auf (A),
- ermitteln die formale Sprache, die durch eine Grammatik erzeugt wird (A),
- entwickeln und modifizieren zu einer Problemstellung endliche Automaten (M),
- entwickeln und modifizieren zu einer Problemstellung endliche Automaten (M),
- entwickeln zur akzeptierten Sprache eines Automaten die zugehörige Grammatik (M),
- entwickeln zur Grammatik einer regulären Sprache einen zugehörigen endlichen Automaten (M),
- modifizieren Grammatiken regulärer Sprachen (M),
- 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).

4.8 Unterrichtsvorhaben Q2-III

Prinzipielle Arbeitsweise eines Computers und Grenzen der Automatisierbarkeit

• Leitfragen: – Was sind die strukturellen Hauptbestandteile eines Computers? – Wie kann man sich die Ausführung eines maschinenahen Programms mit diesen Komponenten vorstellen? – Welche Möglichkeiten bieten Informatiksysteme und wo liegen ihre Grenzen? • Zentrale Kompetenzen: – Argumentieren • Zeitbedarf: 9 Stunden	• Inhaltsfelder: – Informatiksysteme – Informatik, Mensch und Gesellschaft • Inhaltliche Schwerpunkte: – Einzelrechner und Rechnernetzwerke – Grenzen der Automatisierung

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.

Unterrichtssequenzen:

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 Befehle und ihre Repräsentation in einem Binär-Code, der in einem Register gespeichert werden kann
   (c) Analyse und Erläuterung der Funktionsweise eines einfachen maschinennahen Programms

   Beispiel: Addition von 4 zu einer eingegeben Zahl mit einem Rechnermodell

   Material: Modellrechner: http://www.oberstufeninformatik.de/dc/index.html

2. **Grenzen der Automatisierbarkeit**
   (a) Vorstellung und Unlösbarkeit des Halteproblems
   (b) Beurteilung des Einsatzes von Informatiksystemen hinsichtlich prinzipieller Möglichkeiten und prinzipieller Grenzen

Die 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).

4.9 Unterrichtsvorhaben Q2-IV

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

Gesamtzahl der Stunden in der Qualifikationsphase 2: 42

5 Grundsätze der methodischen und didaktischen Arbeit

In Absprache mit der Lehrerkonferenz sowie unter Berücksichtigung des Schulprogramms hat die Fachkonferenz Informatik des Städtischen 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.
3. Die Unterrichtsgestaltung ist auf die Ziele und Inhalte abgestimmt.
4. Medien und Arbeitsmittel sind schülernah gewählt.
5. Die Schüler erreichen einen Lernzuwachs.
6. Der Unterricht fördert eine aktive Teilnahme der Schüler.
7. Der Unterricht fördert die Zusammenarbeit zwischen den Schülern und bietet ihnen Möglichkeiten zu eigenen Lösungen.
8. Der Unterricht berücksichtigt die individuellen Lernwege der einzelnen Schüler.
9. Die Schüler erhalten Gelegenheit zu selbstständiger Arbeit und werden dabei unterstützt.
10. Der Unterricht fördert strukturierte und funktionale Partner- bzw. Gruppenarbeit.
11. Der Unterricht fördert strukturierte und funktionale Arbeit im Plenum.
12. Die Lernumgebung ist vorbereitet; der Ordnungsrahmen wird eingehalten.
13. Die Lehr- und Lernzeit wird intensiv für Unterrichtszwecke genutzt.
14. Es herrscht ein positives pädagogisches Klima im Unterricht.

Fachliche Grundsätze:

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

6 Grundsätze der Leistungsbewertung und -rückmeldung

Auf der Grundlage von §13 - §16 der APO-GOSt sowie Kapitel 3 des Informatik-Kernlehrplans für die gymnasiale Oberstufe hat die Fachkonferenz des Städtischen Gymnasiums Bad Laasphe 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.

Klausuren

Instrumente:

Phase	Anzahl der Klausuren	Dauer einer Klausur
Einführungsphase Ef.1	1 Klausur	1,5 Unterrichtsstunden (90 min.)
Einführungsphase Ef.2	2 Klausuren	1,5 Unterrichtsstunden (90 min.)
Grundkurs Q1.1	2 Klausuren	2,0 Unterrichtsstunden (120 min.)
Grundkurs Q1.2	2 Klausuren	2,0 Unterrichtsstunden (120 min.)
Grundkurs Q2.1	2 Klausuren	2,0 Unterrichtsstunden (135 min.)
Grundkurs Q2.2	1 Klausur unter Abiturbedingungen	3,0 Unterrichtsstunden (180 min.)

Anstelle einer Klausur kann gemäß dem Beschluss der Lehrerkonferenz in Q1.2 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.

Da in der Qualifikationsphase die Note „ausreichend minus“ im Unterschied zur Einführungsphase ein Defizit darstellt, erfolgt die Zuordnung der Prozentpunkte zu den Noten in der Einführungsphase etwas anders.

Spätestens ab der Qualifikationsphase orientiert sich die Zuordnung der Hilfspunktsumme zu den Notenstufen an dem Zuordnungsschema des Zentralabiturs. 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.

Einführungsphase		**Qualifikationssphase**		
Prozentpunkte	Note	Prozentpunkte	Punkte	Note
95–100	1	95–100	15	1+
90–94	1−	90–94	14	1
85–89	2+	85–89	13	1−
80–84	2	80–84	12	2+
75–79	2−	75–79	11	2
70–74	3+	70–74	10	2−
65–69	3	65–69	9	3+
60–64	3−	60–64	8	3
55–59	4+	55–59	7	3−
50–54	4	50–54	6	4+
45–49	4−	45–49	5	4
37–44	5+	39–44	4	4−
30–36	5	33–38	3	5+
22–29	5−	27–32	2	5
0–21	6	20–26	1	5−
		0–19	0	6

Sonstige Mitarbeit

Den Schülern werden die Kriterien zum Beurteilungsbereich der „sonstigen Mitarbeit“ zu Beginn des Schuljahres genannt.

Leistungsaspekte

- Mündliche und sonstige Leistungen
  - Beteiligung am Unterrichtsgespräch
  - Die Beratung anderer Mitschüler bzw. Teams
  - Zusammenfassungen zur Vor- und Nachbereitung des Unterrichts
  - Präsentation von Arbeitsergebnissen
  - Referate
  - Mitarbeit in Partner-/Gruppenarbeitsphasen
  - Die selbstständige und intensive Auseinandersetzung mit schriftlichen Informationen und Aufgabenstellungen
- Praktische Leistungen am Computer
  - Implementierung, Test und Anwendung von Informatiksystemen
- Sonstige schriftliche Leistungen
  - Lernerfolgsüberprüfung durch kurze schriftliche Übungen
    In Kursen, in denen höchstens 50% der Kursmitglieder eine Klausur schreiben, finden schriftliche Übungen mindestens einmal pro Kurshalbjahr statt, in anderen Kursen entscheidet über die Durchführung die Lehrkraft.
    Schriftliche Übungen dauern ca. 20 Minuten und umfassen den Stoff der letzten ca. 4–6 Stunden.
  - Bearbeitung von schriftlichen Aufgaben im Unterricht
  - Bearbeitung von Aufgaben im eingenverantwortlichen Unterricht (SB-Aufgaben bei Unterrichtsausfall)

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,
- die Quantität 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.

Bei Projektarbeit darüber hinaus auf

- die Dokumentation des Arbeitsprozesses,
- den Grad der Selbstständigkeit,
- die Reflexion des eigenen Handelns und
- die Aufnahme von Beratung durch die Lehrkraft.

Grundsätze der Leistungsrückmeldung und Beratung

Die Grundsätze der Leistungsbewertung werden zu Beginn eines jeden Halbjahres den 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.

Die Leistungsrückmeldung kann

- durch ein Gespräch mit dem Schüler,
- durch einen Feedbackbogen,
- durch die schriftliche Begründung einer Note oder
- durch eine individuelle Lern-/Förderempfehlung

erfolgen.

Leistungsrückmeldungen erfolgen auch in der Einführungsphase im Rahmen der kollektiven und individuellen Beratung zur Wahl des Faches als fortgesetztes Grundkursfach in der Qualifikationsphase.

7 Qualitätssicherung und Evaluation

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

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

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

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

(Genehmigt in der Fachkonferenz vom 02.06.2014)

(Ergänzt um die Präzisierung der Klausur-Arbeitszeiten in der Fachkonferenz vom 25.02.2016)

♢