Schulinternes Curriculum

zum Kernlehrplan für die gymnasiale Oberstufe

Informatik

(Stand: 01.07.2015)

1 Inhalt

2 Die Fachgruppe Informatik des Clara-Schumann-Gymnasiums Bonn

Beim Clara-Schumann-Gymnasium (CSG) handelt es sich um eine vierzügige Schule. Das Fach Informatik wird am CSG ab der Jahrgangsstufe 8 im Differenzierungsbereich vierstündig unterrichtet. In der zweijährigen Laufzeit dieser Kurse wird in altersstufengerechter Weise unter anderem auf ausgewählte Grundlagen der Algorithmik, auf die technische Informatik am Beispiel von Schaltwerken und Schaltnetzen und auf Grundlangen der Programmierung eingegangen. Der Unterricht erfolgt dabei in enger Verzahnung mit Inhalten der Mathematik und Physik und wird zum Teil in Form von Projekten gestaltet.

Organisatorisch wird das Fach Informatik in der Sekundarstufe I den Schülerinnen und Schülern als Alternative zu den Sprachen Latein und Italienisch angeboten.

In der Sekundarstufe II bietet das CSG 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. Die in der Sekundarstufe I verwendete Programmiersprache unterscheidet sich von der aus der Sekundarstufe II.

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

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 CSG aus drei Lehrkräften. Es stehen zwei Computerräume jeweils 15 Computerarbeitsplätzen für die Schüler(innen) zur Verfügung. Alle Arbeitsplätze sind an das schulinterne Schüler-Rechnernetz angeschlossen, so dass Schülerinnen und Schüler über einen individuell gestaltbaren Zugang zum zentralen Server der Schule alle Arbeitsplätze der beiden Räume zum Zugriff auf ihre eigenen Daten, zur Recherche im Internet oder zur Bearbeitung schulischer Aufgaben verwenden können.

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

3 Entscheidungen zum Unterricht

3.1 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“ (Unterkapitel 0) 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“** (Unterkapitel 3.1.3) 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.

3.1.1 Konkretisierte Kompetenzerwartungen gemäß Lehrplan

3.1.1.1 Konkretisierte Kompetenzerwartungen gemäß Lehrplan am Ende der Einführungsphase (EF)

Im Folgenden werden die im Kernlehrplan aufgeführten in der Einführungsphase zu erreichenden/erfüllenden konkretisierten Kompentenzerwartungen aufgeführt.

Inhaltsfeld 1: Daten und ihre Strukturierung

Inhaltlicher Schwerpunkt:

- Objekte und Klassen

Objekte und Klassen

Die Schülerinnen und Schüler
- ermitteln bei der Analyse einfacher Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M),
- modellieren Klassen mit ihren Attributen, ihren Methoden und Assoziationsbeziehungen (M),
- modellieren Klassen unter Verwendung von Vererbung (M),
- ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare Datensammlungen zu (M),
- ordnen Klassen, Attributen und Methoden ihren Sichtbarkeitsbereich zu (M),
- stellen den Zustand eines Objekts dar (D),
- stellen die Kommunikation zwischen Objekten grafisch dar (M),
- stellen Klassen, Assoziations- und Vererbungsbeziehungen in Diagrammen grafisch dar (D),
- dokumentieren Klassen durch Beschreibung der Funktionalität der Methoden (D),
- analysieren und erläutern eine objektorientierte Modellierung (A),
- implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I).

Inhaltsfeld 2: Algorithmen

Inhaltliche Schwerpunkte:

- Analyse, Entwurf und Implementierung einfacher Algorithmen
- Algorithmen zum Suchen und Sortieren

Analyse, Entwurf und Implementierung einfacher Algorithmen

Die Schülerinnen und Schüler
- analysieren und erläutern einfache Algorithmen und Programme (A),
- modifizieren einfache Algorithmen und Programme (I),
- entwerfen einfache Algorithmen und stellen sie umgangssprachlich und grafisch dar (M),

- implementieren Algorithmen unter Verwendung von Variablen und Wertzuweisungen, Kontrollstrukturen sowie Methodenaufrufen (I),
- testen Programme schrittweise anhand von Beispielen (I).

Algorithmen zum Suchen und Sortieren

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

Inhaltsfeld 3: Formale Sprachen und Automaten

Inhaltlicher Schwerpunkt:

- Syntax und Semantik einer Programmiersprache

Syntax und Semantik einer Programmiersprache

Die Schülerinnen und Schüler
- implementieren einfache Algorithmen unter Beachtung der Syntax und Semantik einer Programmiersprache (I),
- interpretieren Fehlermeldungen und korrigieren den Quellcode (I).

Inhaltsfeld 4: Informatiksysteme

Inhaltliche Schwerpunkte:

- Digitalisierung
- Einzelrechner
- Dateisystem
- Internet

Digitalisierung

Die Schülerinnen und Schüler
- stellen ganze Zahlen und Zeichen in Binärcodes dar (D),
- interpretieren Binärcodes als Zahlen und Zeichen (D).

Einzelrechner

Die Schülerinnen und Schüler
- beschreiben und erläutern den strukturellen Aufbau und die Arbeitsweise singulärer Rechner am Beispiel der „Von-Neumann-Architektur" (A).

Dateisystem

Die Schülerinnen und Schüler
- nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung und gemeinsamen Verwendung von Daten unter Berücksichtigung der Rechteverwaltung (K).

Internet

Die Schülerinnen und Schüler
- nutzen das Internet zur Recherche, zum Datenaustausch und zur Kommunikation (K).

Inhaltsfeld 5: Informatik, Mensch und Gesellschaft

Inhaltliche Schwerpunkte:

- Einsatz von Informatiksystemen
- Wirkungen der Automatisierung
- Geschichte der automatischen Datenverarbeitung

Einsatz von Informatiksystemen

Die Schülerinnen und Schüler
- nutzen die im Unterricht eingesetzten Informatiksysteme selbstständig, sicher, zielführend und verantwortungsbewusst (D).

Wirkungen der Automatisierung

Die Schülerinnen und Schüler
- bewerten anhand von Fallbeispielen die Auswirkungen des Einsatzes von Informatiksystemen (A).

Geschichte der automatischen Datenverarbeitung

Die Schülerinnen und Schüler
- erläutern wesentliche Grundlagen der Geschichte der digitalen Datenverarbeitung (A).

3.1.1.2 Konkretisierte Kompetenzerwartungen gemäß Lehrplan am Ende der Qualifikationsphase (Q1 und Q2)

Im Folgenden werden die im Kernlehrplan aufgeführten in der Qualifikationsphase zu erreichenden/erfüllenden konkretisierten Kompetenzerwartungen für den Grundkurs aufgeführt. (Zur Zeit gibt es am Clara-Schumann-Gymnasium keinen Leistungskurs.)

Inhaltsfeld ❶ Daten und ihre Strukturierung

Inhaltliche Schwerpunkte

Objekte und Klassen

Datenbanken

Objekte und Klassen

Die Schülerinnen und Schüler

- ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M),
- stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D),
- modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (M),
- modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M),
- ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu (M),
- verwenden bei der Modellierung geeigneter Problemstellungen Möglichkeiten der Polymorphie (M),
- ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M),
- stellen die Kommunikation zwischen Objekten grafisch dar (D),
- stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D),
- dokumentieren Klassen (D),
- analysieren und erläutern objektorientierte Modellierungen (A),
- implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I).

Datenbanken

Die Schülerinnen und Schüler

- ermitteln für anwendungsbezogene Problemstellungen Entitäten, zugehörige Attribute, Relationen und Kardinalitäten (M),
- stellen Entitäten mit ihren Attributen und die Beziehungen zwischen Entitäten mit Kardinalitäten in einem Entity-Relationship-Diagramm grafisch dar (D),
- modifizieren eine Datenbankmodellierung (M),
- modellieren zu einem Entity-Relationship-Diagramm ein relationales Datenbankschema (M),
- bestimmen Primär- und Sekundärschlüssel (M),

- analysieren und erläutern eine Datenbankmodellierung (A),
- erläutern die Eigenschaften normalisierter Datenbankschemata (A),
- überprüfen Datenbankschemata auf vorgegebene Normalisierungseigenschaften (D),
- überführen Datenbankschemata in die 1. bis 3. Normalform (M).

Inhaltsfeld ❷ Algorithmen

Inhaltliche Schwerpunkte

Analyse, Entwurf und Implementierung von Algorithmen

Algorithmen in ausgewählten informatischen Kontexten

Analyse, Entwurf und Implementierung von Algorithmen

Die Schülerinnen und Schüler

- analysieren und erläutern Algorithmen und Programme (A),
- modifizieren Algorithmen und Programme (I),
- stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D),
- entwickeln iterative und rekursive Algorithmen unter Nutzung der Strategien „Modularisierung" und „Teilen und Herrschen" (M),
- implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I),
- testen Programme systematisch anhand von Beispielen (I).

Algorithmen in ausgewählten informatischen Kontexten

Die Schülerinnen und Schüler

- erläutern Operationen dynamischer (linearer oder nichtlinearer) Datenstrukturen (A),
- implementieren und erläutern iterative und rekursive Such- und Sortierverfahren (I),
- beurteilen die Effizienz von Algorithmen unter Berücksichtigung des Speicherbedarfs und der Zahl der Operationen (A),
- ermitteln Ergebnisse von Datenbankabfragen über mehrere verknüpfte Tabellen (D).

Inhaltsfeld ❸ Formale Sprachen und Automaten

Inhaltliche Schwerpunkte

Syntax und Semantik einer Programmiersprache

Endliche Automaten

Grammatiken regulärer Sprachen

Möglichkeiten und Grenzen von Automaten und formalen Sprachen

Syntax und Semantik einer Programmiersprache

Die Schülerinnen und Schüler

- nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I),
- beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A),
- interpretieren Fehlermeldungen und korrigieren den Quellcode (I),
- analysieren und erläutern die Syntax und Semantik einer Datenbankabfrage (A),
- verwenden die Syntax und Semantik einer Datenbankabfragesprache, um Informationen aus einem Datenbanksystem zu extrahieren (I).

Endliche Automaten

Die Schülerinnen und Schüler

- analysieren und erläutern die Eigenschaften endlicher Automaten einschließlich ihres Verhaltens bei bestimmten Eingaben (A),
- ermitteln die Sprache, die ein endlicher Automat akzeptiert (D),
- entwickeln und modifizieren zu einer Problemstellung endliche Automaten (M),
- stellen endliche Automaten in Tabellen oder Graphen dar und überführen sie in die jeweils andere Darstellungsform (D),
- entwickeln zur Grammatik einer regulären Sprache einen zugehörigen endlichen Automaten (M).

Grammatiken regulärer Sprachen

Die Schülerinnen und Schüler

- analysieren und erläutern Grammatiken regulärer Sprachen (A),

- modifizieren Grammatiken regulärer Sprachen (M),
- ermitteln die formale Sprache, die durch eine Grammatik erzeugt wird (A),
- entwickeln zu einer regulären Sprache eine Grammatik, die die Sprache erzeugt (M),
- entwickeln zur akzeptierten Sprache eines Automaten eine zugehörige Grammatik (M),
- beschreiben an Beispielen den Zusammenhang zwischen Automaten und Grammatiken (D).

Möglichkeiten und Grenzen von Automaten und formalen Sprachen

Die Schülerinnen und Schüler

- zeigen die Grenzen endlicher Automaten und regulärer Grammatiken im Anwendungszusammenhang auf (A).

Inhaltsfeld ❹ Informatiksysteme

Inhaltliche Schwerpunkte

Einzelrechner und Rechnernetzwerke

Nutzung von Informatiksystemen

Sicherheit

Einzelrechner und Rechnernetzwerke

Die Schülerinnen und Schüler

- erläutern die Ausführung eines einfachen maschinennahen Programms sowie die Datenspeicherung auf einer „Von-Neumann-Architektur" (A),
- beschreiben und erläutern Netzwerk-Topologien, die Client-Server-Struktur und Protokolle sowie ein Schichtenmodell in Netzwerken (A).

Nutzung von Informatiksystemen

Die Schülerinnen und Schüler

- nutzen bereitgestellte Informatiksysteme und das Internet reflektiert zur Erschließung, Aufbereitung und Präsentation fachlicher Inhalte (D),

- nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung von Dateien unter Berücksichtigung der Rechteverwaltung (K),
- wenden eine didaktisch orientierte Entwicklungsumgebung zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I).

Sicherheit

Die Schülerinnen und Schüler

- erläutern Eigenschaften, Funktionsweisen und Aufbau von Datenbanksystemen unter dem Aspekt der sicheren Nutzung (A),
- analysieren und erläutern Eigenschaften und Einsatzbereiche symmetrischer und asymmetrischer Verschlüsselungsverfahren (A).

Inhaltsfeld ⑤ Informatik, Mensch und Gesellschaft

Inhaltliche Schwerpunkte

Wirkungen der Automatisierung

Grenzen der Automatisierung

Wirkungen der Automatisierung

Die Schülerinnen und Schüler

- untersuchen und bewerten anhand von Fallbeispielen Auswirkungen des Einsatzes von Informatiksystemen sowie Aspekte der Sicherheit von Informatiksystemen, des Datenschutzes 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).

Grenzen der Automatisierung

Die Schülerinnen und Schüler

- untersuchen und beurteilen Grenzen des Problemlösens mit Informatiksystemen (A).

3.1.2 Übersichtsraster Unterrichtsvorhaben

3.1.2.1 Übersichtsraster Unterrichtsvorhaben in der Einführungsphase

<table>
<tr><th colspan="2">Einführungsphase</th></tr>
<tr>
<td>

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

Thema:

Grundlagen der objektorientierten Programmierung anhand von Beispielkontexten

Zentrale Kompetenzen:

- Modellieren
- Implementieren
- Darstellen und Interpretieren
- Kommunizieren und Kooperieren

Inhaltsfelder:

- Daten und ihre Strukturierung
- Algorithmen
- Formale Sprachen und Automaten
- Informatiksysteme
- Informatik, Mensch und Gesellschaft

Inhaltliche Schwerpunkte:

- Objekte und Klassen
- Analyse, Entwurf und Implementierung einfacher Algorithmen
- Syntax und Semantik einer Programmiersprache
- Dateisystem
- Internet
- Einsatz von Informatiksystemen

U. A. behandelte Fachbegriffe und Aspekte:

- Klassen, Objekte, UML-Klassendiagramme

</td>
<td>

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

Thema:

Grundlagen algorithmischer Grundstrukturen in Java anhand von Schleifen, Arrays und statischen Hilfs-Methoden

Zentrale Kompetenzen:

- Argumentieren
- Modellieren
- Implementieren
- Darstellen und Interpretieren
- Kommunizieren und Kooperieren

Inhaltsfelder:

- Algorithmen
- Formale Sprachen und Automaten
- Informatiksysteme
- Informatik, Mensch und Gesellschaft

Inhaltliche Schwerpunkte:

- Objekte und Klassen
- Analyse, Entwurf und Implementierung einfacher Algorithmen
- Syntax und Semantik einer Programmiersprache
- Dateisystem
- Internet
- Einsatz von Informatiksystemen

U. A. behandelte Fachbegriffe und Aspekte:

- Klassen-Methoden (static)
- Kontrollstrukturen: Schleifen (while, for, do-while),

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

• Methoden, Parameterübergabe, Rückgabewerte • Schutzklassen-Modifier: public, protected, private • Geheimnisprinzip, Kapselung, Getter- und Setter-Methoden • Lokale Variable, Instanzvariable (Attribute) • Vererbung/Generalisierung/ Spezialisierung • Gerichtete Assoziationen • Verzweigungen (if) • Klassenvariable (static), Konstante (final) • Verwendung von Java-Doc, Angabe sinnvoller Kommentare in eigenen Quelltexten, Wahl bedeutungsvoller Namen für Variable, Konstante, Methoden und Klassen, Namenskonventionen für Groß- und Kleinschreibung in Java • Idee, dass jedes Objekt als möglichst aussagekräftiger String repräsentiert werden können soll, Überschreiben der von der Klasse Object geerbten toString-Methode **Zeitbedarf**: 15 Stunden	• Elementare Datentypen und gegenseitige Konvertierungen • Konvertierung von und nach String • die statische Datenstruktur Array • mehrdimensionale Arrays • lineare Suche im Array • Suche nach Minimum/Maximum • Untersuchung von Arrays auf Gleichheit (im Ggs. zum Vergleich der Referenzen) **Zeitbedarf**: 14 Stunden

Einführungsphase	
Unterrichtsvorhaben E-III **Thema:** *Einführung in die Nutzung von Informatiksystemen und in grundlegende Begrifflichkeiten mit besonderem Augenmerk auf die*	*Unterrichtsvorhaben E-IV* **Thema**: *Grundlagen der objektorientierten Programmierung (OOP) mit besonderem Schwerpunkt*

Funktionsweise der JavaVirtualMachine und der Von-Neumann-Architektur

Zentrale Kompetenzen:

- Argumentieren
- Modellieren
- Implementieren
- Darstellen und Interpretieren
- Kommunizieren und Kooperieren

Inhaltsfelder:

- Daten und ihre Strukturierung
- Formale Sprachen und Automaten
- Informatiksysteme
- Informatik, Mensch und Gesellschaft

Inhaltliche Schwerpunkte:

- Objekte und Klasse
- Syntax und Semantik einer Programmiersprache
- Digitalisierung
- Einzelrechner
- Dateisystem
- Internet
- Einsatz von Informatiksystemen

U. A. behandelte Fachbegriffe und Aspekte:

- Binärsystem und p-adische Darstellung von Zahlen
- ASCII-Code und Uni-Code
- Von-Neumann-Architektur
- Kleine grundlegende Einblicke in die JVM und den Garbage-Collector

Zeitbedarf: 9 Stunden

Polymorphismus und Typ-Definitionen

Zentrale Kompetenzen:

- Argumentieren
- 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
- Dateisystem
- Internet
- Einsatz von Informatiksystemen

U. A. behandelte Fachbegriffe und Aspekte:

- Typ (im Gegensatz zum Begriff Klasse), Java-Interface
- Polymorphie,
- spätes Binden
- Type-Casting
- Abstrakte Klassen, Interfaces
- Multiplizitäten, Kardinalitäten

Zeitbedarf: 5 Stunden

Einführungsphase	
Unterrichtsvorhaben E-V **Thema**: *Klassenentwurf , Aspekte des objektorientierten Designs (OOD) und der objektorientierten Analyse (OOA) am Beispiel verschiedener Entwurfsmöglichkeiten eines Integer-Arrays* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Daten und ihre Strukturierung • Daten und ihre Strukturierung • Formale Sprachen und Automaten • Informatiksysteme • Informatik, Mensch und Gesellschaft **Inhaltliche Schwerpunkte:** • Objekte und Klassen • Syntax und Semantik einer Programmiersprache • Dateisystem • Internet • Einsatz von Informatiksystemen **U. A. behandelte Fachbegriffe und As-**	*Unterrichtsvorhaben E-VI* **Thema**: *Aspekte des objektorientierten Designs (OOD) bei der Modellierung und Implementierung von Klassen- und Objektbeziehungen am Beispiel des Design-Patterns Observer und des Model-View-Controller-Paradigmas* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen und Automaten • Internet • Informatik, Mensch und Gesellschaft **Inhaltliche Schwerpunkte:** • Objekte und Klassen • Analyse, Entwurf und Implementierung einfacher Algorithmen • Syntax und Semantik einer Programmiersprache • Informatiksysteme • Dateisystem

pekte: • OOP, OOD und OOA • Objektorientierte Modellierung (OOM) • Referenzen, Vergleich von Arrays, Strings und anderen Objekten, Überschreiben der von der Klasse Object geerbten equals-Methode • Flache und tiefe Kopie, Überschreiben der von der Klasse Object geerbten clone-Methode • Konzepte des Klassenentwurfs (u. a. Kopplung und Kohäsion, Kapselung, Entwurf nach Zuständigkeiten, Vermeiden von Code-Duplizierung) **Zeitbedarf**: 7 Stunden	• Einsatz von Informatiksystemen **U. A. behandelte Fachbegriffe und Aspekte:** • Idee und Grund von/für Design Patterns, • Das Verhaltensmuster Observer (im Gegensatz zu Polling) • Erstellung grafischer Benutzeroberflächen, • Verwendung des GUI-Builders von Netbeans • Trennung von Model und View/Controller (UI-Delegate) **Zeitbedarf**: 8 Stunden

Einführungsphase	
<u>*Unterrichtsvorhaben E-VII*</u> **Thema**: *Entwurf und Analyse von Sortierverfahren auf Arrays* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:**	<u>*Unterrichtsvorhaben E-VIII*</u> **Thema**: *Generische Programmierung* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Daten und ihre Strukturierung

• Algorithmen • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen und Automaten • Informatiksysteme • Informatik, Mensch und Gesellschaft **Inhaltliche Schwerpunkte:** • Objekte und Klassen • Analyse, Entwurf und Implementierung einfacher Algorithmen • Algorithmen zum Suchen und Sortieren • Syntax und Semantik einer Programmiersprache • Dateisystem • Internet • Einsatz von Informatiksystemen **U. A. behandelte Fachbegriffe und Aspekte:** • Sortieren durch Einfügen: Insertionsort • Sortieren durch Auswählen: Minsort, Maxsort • Darstellung mit einem Struktogramm • Effizienzanalyse: Laufzeitanalyse/Zeitkomplexität mithilfe der Gauß'schen Summationsformel, Worst-Case, Best-Case, Average-Case, Platzkomplexität, • Groß-O-Notation (nur für sehr einfache konvergente oder divergente Laufzeitfolgen, kein Limes-Superior) • Implementierung von Insertionsort und (Minsort oder Maxsort)	• Algorithmen • Formale Sprachen und Automaten • Informatiksysteme • Informatik, Mensch und Gesellschaft **Inhaltliche Schwerpunkte:** • Objekte und Klassen • Analyse, Entwurf und Implementierung einfacher Algorithmen • Syntax und Semantik einer Programmiersprache • Dateisystem • Internet • Einsatz von Informatiksystemen **U. A. behandelte Fachbegriffe und Aspekte:** • Der Diamond-Operator (Typ-Inferenz-Operator <>) • Typkonkretisierung beim Erzeugen einer Instanz (mit new) • Typkonkretisierung beim Ableiten einer Klasse oder Implementieren eines Interfaces (mit extends oder implements) • Das Interface ComparableContent aus den Materialien für das Zentralabitur ab Abitur 2017 • Implementieren eigener Klassen, die ComparableContent implementieren • Überlegungen für die Verallgemeinerung von Sortierverfahren.

Zeitbedarf: 6 Stunden	**Zeitbedarf**: 5 Stunden

Einführungsphase	
Unterrichtsvorhaben E-IX **Thema**: *Geschichte der digitalen Datenverarbeitung und die Grundlagen des Datenschutzes* **Zentrale Kompetenzen:** • Argumentieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Informatiksysteme • Informatik, Mensch und Gesellschaft **Inhaltliche Schwerpunkte:** • Dateisystem • Internet • Wirkungen der Automatisierung • Geschichte der automatischen Datenverarbeitung **U. A. behandelte Fachbegriffe und Aspekte:** • Berühmte Namen von Firmen und Personen	

Zeitbedarf: 5 Stunden	

Summe Einführungsphase: 74 h

3.1.2.2 Übersichtsraster Unterrichtsvorhaben in der Qualifikationsphase

Zurzeit wird das Fach Informatik am Clara-Schumann-Gymnasium nur im Grundkurs angeboten.

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: 03. 07. 2014)

- **Qualifikationsphase I**

Qualifikationsphase I	
Unterrichtsvorhaben Q1-I **Thema**: *Modellierung und Implementierung von Anwendungen mit dynamischen, linearen Datenstrukturen* **Zentrale Kompetenzen:** • Modellieren • Darstellen und Interpretieren • Argumentieren • Implementieren **Inhaltsfelder:** • Daten und ihre Strukturierung • Algorithmen	*Unterrichtsvorhaben Q1-II* **Thema**: *Rekursive Programmierstrategien* **Zentrale Kompetenzen:** • Modellieren • Darstellen und Interpretieren • Argumentieren • Implementieren **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 **U. A. behandelte Fachbegriffe und Aspekte:** • Queue, Stack, List • Entwurfsdiagramme und Implementationsdiagramme **Zeitbedarf**: 14 Stunden	**Inhaltliche Schwerpunkte:** • Objekte und Klassen • Analyse, Entwurf und Implementierung von Algorithmen • Syntax und Semantik einer Programmiersprache • Algorithmen in ausgewählten informatischen Kontexten **U. A. behandelte Fachbegriffe und Aspekte:** • Rekursionsverankerung, rekursiver Ab- und Aufstieg, rekursiver Aufruf, Rekursionstiefe • Untersuchung der Aufrufstruktur, Rekursionsformen (lineare Rek., kaskadenartige Rek., vernestete Rek. verschränkte Rek.) **Zeitbedarf**: 6 Stunden

Qualifikationsphase I	
Unterrichtsvorhaben Q1-III **Thema**: *Bäume (als spezieller Graph und als dynamische, nichtlineare Datenstruktur)* **Zentrale Kompetenzen:** • Modellieren • Darstellen und Interpretieren • Argumentieren • Implementieren **Inhaltsfelder:**	*Unterrichtsvorhaben Q1-IV* **Thema**: *Suchen auf linearen Datenstrukturen* **Zentrale Kompetenzen:** • Modellieren • Darstellen und Interpretieren • Argumentieren • Implementieren **Inhaltsfelder:** • Algorithmen

• Daten und ihre Strukturierung • Algorithmen **Inhaltliche Schwerpunkte:** • Daten und ihre Strukturierung • Objekte und Klassen • Analyse, Entwurf und Implementierung von Algorithmen • Algorithmen in ausgewählten informatischen Kontexten • Analyse, Entwurf und Implementierung von Algorithmen **U. A. behandelte Fachbegriffe und Aspekte:** • Grundlegende Fachbegriffe: Grad, Tiefe, Höhe, Blatt, Inhalt, Teilbaum, Ebene, vollständiger Baum, voller Baum, Knoten, Kanten, … • Rekursive Verarbeitung von Bäumen • Standard-Traversierungen: Pre-Order, In-Order, Post-Order, Level-Order **Zeitbedarf**: 9 Stunden	• Formale Sprachen und Automaten • Informatiksysteme **Inhaltliche Schwerpunkte:** • Algorithmen zum Suchen und Sortieren • Objekte und Klassen • Analyse, Entwurf und Implementierung von Algorithmen • Algorithmen in ausgewählten informatischen Kontexten • Syntax und Semantik einer Programmiersprache • Nutzung von Informatiksystemen **U. A. behandelte Fachbegriffe und Aspekte:** • Lineare vs. binäre Suche auf Arrays • Voraussetzungen für binäre Suche auf Arrays • Lineare Suche auf Listen • Effizienzanalyse **Zeitbedarf**: 2 Stunden

Qualifikationsphase I	
<u>*Unterrichtsvorhaben Q1-V*</u> **Thema**: *Suchen mit binären Suchbäumen* **Zentrale Kompetenzen:** • Modellieren	<u>*Unterrichtsvorhaben Q1-VI*</u> **Thema**: *Sortierverfahren auf Listen* **Zentrale Kompetenzen:** • Modellieren

• Darstellen und Interpretieren • Argumentieren • Implementieren **Inhaltsfelder:** • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen und Automaten • Informatiksysteme **Inhaltliche Schwerpunkte:** • Objekte und Klassen • Objekte und Klassen • Analyse, Entwurf und Implementierung von Algorithmen • Algorithmen in ausgewählten informatischen Kontexten • Syntax und Semantik einer Programmiersprache • Nutzung von Informatiksystemen **U. A. behandelte Fachbegriffe und Aspekte:** • Implementation von Einfügen und Suchen • Laufzeitanalyse von Suche beim optimalen (also vollständigen) Suchbaum im Worst-, Best- und Avg.-Case • Gegenüberstellung zu binärer Suche in Arrays • Implementation und Analyse von BinaryTreeSort **Zeitbedarf**: 14 Stunden	• Darstellen und Interpretieren • Argumentieren • Implementieren • 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 • Algorithmen in ausgewählten informatischen Kontexten • Syntax und Semantik einer Programmiersprache • Nutzung von Informatiksystemen **U. A. behandelte Fachbegriffe und Aspekte:** • Übertragung der Sortierverfahren Insertionsort und Minsort auf Listen • Entwicklung, Analyse und Implementierung BinaryTreeSort • Entwicklung von Merge- und Quicksort auf Listen, das Divide&Conquer-Algorithmenentwurfs-Paradigma • Analyse von Merge und Quicksort, Gegenüberstellung von BinaryTreeSort und Quicksort auf Listen **Zeitbedarf**: 8 Stunden

Qualifikationsphase I	
Unterrichtsvorhaben Q1-VII **Thema**: *Modellierung und Nutzung von relationalen Datenbanken in Anwendungskontexten* **Zentrale Kompetenzen:** • Modellieren • Darstellen und Interpretieren • Argumentieren • Implementieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen und Automaten **Inhaltliche Schwerpunkte:** • Datenbanken • Algorithmen in ausgewählten informatischen Kontexten • Syntax und Semantik einer Programmiersprache **U. A. behandelte Fachbegriffe und Aspekte:** • 1. bis 3. Normalformen und Normalisierung • Entity-Relationship-Diagramme • Entitäten, Attribute, Relationen, Kardinalitäten • Tabelle, Datentyp • Datenbankschemata	

• Schlüsselkandidaten, Primär- und Sekundärschlüsseln • Primärschlüssel, Fremdschlüssel • Annomalien, Redundanz, Konsistenz • Relationenalgebra und RA-Anfragen • SQL-Abfragen (JOIN, UNION, AS, GROUP BY,ORDER BY, ASC, DESC, COUNT, MAX, MIN, SUM, Arithmetische Operatoren: +, -, *, /, (…), Vergleichsoperatoren: =, <>, >, <, >=, <=, LIKE, BETWEEN, IN, IS NULL) • Verwendung SQL zur Erzeugung von Datenbanken (DDL) **Zeitbedarf**: 22 Stunden	

Summe Qualifikationsphase I: 75h

- **Qualifikationsphase II**

Qualifikationsphase II	
<u>*Unterrichtsvorhaben Q2-I*</u> **Thema**: *Endliche Automaten und formale Sprachen* **Zentrale Kompetenzen:** • Modellieren • Darstellen und Interpretieren • Argumentieren • Implementieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Algorithmen • Formale Sprachen und Automaten	<u>*Unterrichtsvorhaben Q2-II*</u> **Thema**: *Grenzen der Automatisierbarkeit* **Zentrale Kompetenzen:** • Modellieren • Darstellen und Interpretieren • Argumentieren • Implementieren • 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 • Endliche Automaten • Grammatiken regulärer Sprachen • Möglichkeiten und Grenzen von Automaten und formalen Sprachen • Nutzung von Informatiksystemen **U. A. behandelte Fachbegriffe und Aspekte:** • DEA, NEA • Alphabet, Phrasenstrukturgrammatik, formale Sprachen, kleenesche Hülle • reguläre Grammatik, reguläre Sprache • Das Pumping-Lemma für reguläre Sprachen **Zeitbedarf**: 14 Stunden	• Informatiksysteme • Informatik, Mensch und Gesellschaft **Inhaltliche Schwerpunkte:** • Objekte und Klassen • Analyse, Entwurf und Implementierung von Algorithmen • Algorithmen in ausgewählten informatischen Kontexten • Syntax und Semantik einer Programmiersprache • Endliche Automaten • Grammatiken regulärer Sprachen • Möglichkeiten und Grenzen von Automaten und formalen Sprachen • Nutzung von Informatiksystemen • Grenzen der Automatisierung **U. A. behandelte Fachbegriffe und Aspekte:** • Die Chomsky-Hierarchie (1h) • Turingmaschine (2h) • Das Wortproblem für reguläre Sprachen • Das Wortproblem für Typ-0-Sprachen • Turing-Berechenbarkeit • Grenzen praktischer Berechenbarkeit bei exponentieller Laufzeit bei Backtracking-Algorithmen **Zeitbedarf**: 16 Stunden

Qualifikationsphase II	
Unterrichtsvorhaben Q2-III **Thema**: *Prinzipielle Arbeitsweise eines Computers* **Zentrale Kompetenzen:** • Implementieren • Argumentieren • Darstellen und Interpretieren **Inhaltsfelder:** • Formale Sprachen und Automaten • Informatiksysteme **Inhaltliche Schwerpunkte:** • Syntax und Semantik einer Programmiersprache • Einzelrechner und Rechnernetzwerk • Nutzung von Informatiksystemen **U. A. behandelte Fachbegriffe und Aspekte:** • Das Von-Neumann-Rechner Modell • Eine Assembler- und Maschinen-Sprache • Simulation und Untersuchung realer Programme **Zeitbedarf**: 7 Stunden	*Unterrichtsvorhaben Q1-IV* **Thema**: *Sicherheit und Datenschutz in Netzstrukturen* **Zentrale Kompetenzen:** • Implementieren • Argumentieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Informatiksysteme • Informatik, Mensch und Gesellschaft **Inhaltliche Schwerpunkte:** • Einzelrechner und Rechnernetzwerke • Nutzung von Informatiksystemen • Sicherheit • Wirkung der Automatisierung **U. A. behandelte Fachbegriffe und Aspekte:** • Das TCP-IP-Referenzmodell • Repräsentation von Information • Symmetrische Verschlüsselungsverfahren (Cäsar, Vigenère, One-Time-Pad) • Perfekte Sicherheit • Asymmetrische Verschlüsselungsverfahren und ein konkretes Beispiel • Datenschutz und Urheberrecht **Zeitbedarf**: 16 Stunden

Qualifikationsphase II	
Unterrichtsvorhaben Q2-V **Thema**: *Wiederholungsphase* **Zentrale Kompetenzen:** • sämtliche Kompetenzen **Inhaltsfelder:** • sämtliche Inhaltsfelder **Inhaltliche Schwerpunkte:** • sämtliche Schwerpunkte, angepasst auf die Bedürfnisse der Lerngruppe **Zeitbedarf**: Bedarfsgerecht	

Summe Qualifikationsphase II: 75h + Wdh.

3.1.3 Konkretisierte Kompetenzerwartungen

3.1.3.1 Konkretisierte Kompetenzerwartungen am Ende der Einführungsphase

Im Folgenden sollen die in Abschnitt 1 aufgeführten Unterrichtsvorhaben konkretisiert werden. Diese Konkretisierung hat vorschlagenden Charakter, ohne die pädagogische Freiheit des Lehrenden einschränken zu wollen.

In der Einführungsphase wird im ersten Halbjahr die didaktische Bibliothek BlueJ verwendet. Innerhalb des zweiten Halbjahres wird auf die Umgebung Netbeans so früh umgestellt, dass der Umgang mit Netbeans mit Beenden der Einführungsphase den Schülerinnen und Schülern für die Erfordernisse in den Stufen Q1 und Q2 hinreichend bekannt ist.

Darüber hinaus **können** weitere didaktische Umgebungen und Software-Produkte wie z. B. GLOOP oder UML-Editoren (wie z. B. UMLet oder StarUML) zusätzlich eingesetzt werden.

Die Von-Neumann-Architektur wird mit MOPS oder JOHNNY analysiert und simuliert.

Bei den Automaten in der Qualifikationsphase wird das Programmpaket ATOCC verwendet.

Die übergeordneten Kompetenzen des Kompetenzbereichs "Kommunizieren und Kooperieren" werden in jedem Unterrichtsvorhaben erworben bzw. vertieft und sind daher nicht jedes Mal erneut aufgeführt.

Kommunizieren und Kooperieren (K)

Schülerinnen und Schüler

- verwenden Fachausdrücke bei der Kommunikation über informatische Sachverhalte,
- kommunizieren und kooperieren in Gruppen und in Partnerarbeit,
- präsentieren Arbeitsabläufe und Arbeitsergebnisse.

Ebenso bieten fast alle Unterrichtsvorhaben, in denen Programme implementiert werden, die Gelegenheit, die folgenden Kompetenzen zu erwerben bzw. zu vertiefen:

Schülerinnen und Schüler

- ermitteln bei der Analyse einfacher Problemstellung Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M),
- dokumentieren Klassen durch Beschreibung der Funktionalität der Methoden (D),
- analysieren und erläutern eine objektorientierte Modellierung (A),
- implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I),
- analysieren und erläutern einfache Algorithmen und Programme (A),
- modifizieren einfache Algorithmen und Programme (I),
- entwerfen einfache Algorithmen und stellen sie umgangssprachlich und grafisch dar (M),
- implementieren Algorithmen unter Verwendung von Variablen und Wertzuweisungen, Kontrollstrukturen sowie Methodenaufrufen (I),
- testen Programme schrittweise anhand von Beispielen (I),
- implementieren einfache Algorithmen unter Beachtung der Syntax und Semantik einer Programmiersprache (I),
- interpretieren Fehlermeldungen und korrigieren den Quellcode (I),

- nutzen das verfügbare Informatiksystem zu strukturierten Verwaltung und gemeinsamen Verwendung von Daten unter Berücksichtigung der Rechteverwaltung (K),
- nutzen das Internet zur Recherche, zum Datenaustausch und zur Kommunikation (K),
- nutzen die im Unterricht eingesetzten Informatiksysteme selbstständig, sicher, zielführend und verantwortungsbewusst (D).

Da in der Einführungsphase das Hauptaugenmerk auf die Einführung der objektorientierten Programmiersprache liegt, werden die oben angegeben Kompetenzbezüge nicht mehr explizit bei den einzelnen Themenblöcken genannt.

Unterrichtsvorhaben EF-I

Thema: *Grundlagen der objektorientierten Programmierung anhand von Beispielkontexten*

Leitfragen: Wie kann ein Problem im informatischen System abgebildet werden? Wie modelliert und implementiert man in der Programmiersprache Java?

Zeitbedarf: 15 Stunden (5 Wochen Einführung)

Vorhabenbezogene Konkretisierung:

Ausgehend von einem vorgegebenen Projekt wird der Unterschied zwischen einer Klasse und einem Objekt dargestellt. Innerhalb dieses Projekts werden eigene Klassen erstellt, u. A. indem Quelltexte kopiert und geringfügig abgeändert werden. Bei der Untersuchung von Objekten zur Laufzeit werden Möglichkeiten der Umgebung BlueJ genutzt. Objekte werden zur Laufzeit von anderen Objekten erstellt und verändert.

Erlerntes wird in verschiedenen vorgegebenen und vorimplementierten Kontexten (z. B. Geometrische Formen, Konto, Ticketautomat o. a.) reproduziert und in freien Aufgabenstellungen reorganisiert.

Unterrichtssequenzen	**zu entwickelnde Kompetenzen**
1. Einführung in Klassen und Objekte mit BlueJ anhand von Beispielkontexten Sichtbarkeit und Kapselung	Die Schülerinnen und Schüler • ermitteln bei der Analyse einfacher Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • modellieren Klassen mit ihren Attributen, ihren Methoden und Assoziationsbeziehungen (M),
2. Vererbung	

	• modellieren Klassen unter Verwendung von Vererbung (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), • stellen den Zustand eines Objekts dar (D), • stellen Klassen, Assoziations- und Vererbungsbeziehungen in Diagrammen grafisch dar (D), • dokumentieren Klassen durch Beschreibung der Funktionalität der Methoden (D), • 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). • nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung und gemeinsamen Verwendung von Daten unter Berücksichtigung der Rechteverwaltung (K). • nutzen das Internet zur Recherche, zum Datenaustausch und zur Kommunikation (K). • nutzen die im Unterricht eingesetzten Informatiksysteme selbstständig, sicher, zielführend und verantwortungsbewusst (D).

Unterrichtsvorhaben EF-II

Thema: Grundlagen algorithmischer Grundstrukturen in Java anhand von Schleifen, Arrays und statischen Hilfs-Methoden

Leitfragen: Wie programmiert man lineare Suche und Such-Varianten? Wie speichert und verwaltet man mehrere Daten des gleichen Typs?

Zeitbedarf: 14 Stunden
(z. B.: 2h statische Variable und Methoden, 6h Arrays, 6h Schleifen)

Vorhabenbezogene Konkretisierung:

Es wird besonders auf Aspekte der imperativen Programmierung (als Überbegriff über prozedurale und objektorientierte Programmierung) anhand von statischen Methoden eingegangen. Die statische Datenstruktur Array wird in Java eingeführt zunächst nur eindimensionale Arrays, anschließend auch vertiefend jedoch nicht ausführlich mehrdimensionale Arrays. Übungen zu for- und while-Schleifen und insbesondere die Entscheidung zwischen diesen beiden Schleifen-Varianten werden trainiert anhand von Aufgabenstellung wie linearer Sucher, Bestimmung kleinster/größter Werte, Erzeugung von Teilarrays und Test zweier Arrays auf Gleichheit vertieft. Für den Vergleich der Daten wird auch auf Konvertierung von Datentypen eingegangen.

Unterrichtssequenzen	**zu entwickelnde Kompetenzen**
1. Statische Methoden und Variable 2. Arrays 3. Schleifen (for- und while) und lineare Suche in Arrays	Die Schülerinnen und Schüler • ermitteln bei der Analyse einfacher Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • modellieren Klassen mit ihren Attributen, ihren Methoden und Assoziationsbeziehungen (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare Datensammlungen zu (M), • ordnen Klassen, Attributen und Methoden ihren Sichtbarkeitsbereich zu (M), • stellen den Zustand eines Objekts dar (D), • stellen Klassen, Assoziations- und Vererbungsbeziehungen in Diagrammen grafisch dar (D), • dokumentieren Klassen durch Beschreibung der Funktionalität der Methoden (D), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I). • analysieren und erläutern einfache Algorithmen und Programme (A), • modifizieren einfache Algorithmen und

	Programme (I), • entwerfen einfache Algorithmen und stellen sie umgangssprachlich und grafisch dar (M), • implementieren Algorithmen unter Verwendung von Variablen und Wertzuweisungen, Kontrollstrukturen sowie Methodenaufrufen (I), • testen Programme schrittweise anhand von Beispielen (I). • implementieren einfache Algorithmen unter Beachtung der Syntax und Semantik einer Programmiersprache (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I). • nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung und gemeinsamen Verwendung von Daten unter Berücksichtigung der Rechteverwaltung (K). • nutzen das Internet zur Recherche, zum Datenaustausch und zur Kommunikation (K). • nutzen die im Unterricht eingesetzten Informatiksysteme selbstständig, sicher, zielführend und verantwortungsbewusst (D).

Unterrichtsvorhaben EF-III

Thema: *Einführung in die Nutzung von Informatiksystemen und in grundlegende Begrifflichkeiten mit besonderem Augenmerk auf die Funktionsweise der JavaVirtualMachine und der Von-Neumann-Architektur*

Leitfragen: Wie wird Information in Computern digital repräsentiert? Wie funktionieren Computer? Wie funktionieren Emulatoren und virtuelle Maschinen?

Zeitbedarf: 9 Stunden
(Z. B.: 4h Zahlen und Zeichen + 3h Von-Neumann-Architektur + 2h JVM und GC)

Vorhabenbezogene Konkretisierung:

Zunächst wird auf Informationsdarstellung in Stellenwertsystemen und Zeichendarstellung mit dem ASCII-Code und Uni-Code eingegangen. Anschließend wird die generelle Funktionsweise eines Von-Neumann-Rechners anhand eines Simulationsprogramms erarbeitet. Schließlich wird die JVM kurz vorgestellt. Hier werden Experimente der Speicherverwaltung mit dem Garbage-Collector durchgeführt, indem große Arrays angelegt und wieder entfernt werden.

Unterrichtssequenzen	**zu entwickelnde Kompetenzen**
1. Repräsentation, Information, Interpretation von Information p-Adische Darstellung von Zahlen, ASCII-Code, Uni-Code	Die Schülerinnen und Schüler • 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), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I). • interpretieren Fehlermeldungen und korrigieren den Quellcode (I). • stellen ganze Zahlen und Zeichen in Binärcodes dar (D), • interpretieren Binärcodes als Zahlen und Zeichen (D). • beschreiben und erläutern den strukturellen Aufbau und die Arbeits-weise singulärer Rechner am Beispiel der „Von-Neumann-Architektur“ (A). • nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung und gemeinsamen Verwendung von Daten unter Berücksichtigung der Rechteverwaltung (K). • nutzen das Internet zur Recherche, zum Datenaustausch und zur Kommunikation (K). • nutzen die im Unterricht eingesetzten Informatiksysteme selbstständig, sicher,
2. Von-Neumann-Architektur	
3. Untersuchung von JVM und Garbage-Collection	

	zielführend und verantwortungsbewusst (D).

Unterrichtsvorhaben EF-IV

Thema: *Grundlagen der objektorientierten Programmierung (OOP) mit besonderem Schwerpunkt Polymorphismus und Typ-Definitionen*

Leitfragen: Was bedeutet Polymorphismus? Wo wird Polymorphismus benötigt?

Zeitbedarf: 5 Stunden

Vorhabenbezogene Konkretisierung:

Es wird der Begriff des Typs erarbeitet. Die beiden in Java gegebenen Möglichkeiten Klassen und Interfaces, Typen in Java zu definieren, werden untersucht und anhand von Anwendungsbeispielen eine Auswahl zwischen abstrakten Klassen und Interfaces getroffen. Zudem wird der Begriff des Polymorphismus anhand eines Arrays verschiedener geometrischer Figuren erarbeitet, die alle einen gemeinsamen Obertyp haben. Das Array wird als animierte Schlange graphisch dargestellt.

Unterrichtssequenzen	**zu entwickelnde Kompetenzen**
1. Typen, Interfaces und Polymorphie 2. Verwendung von allgemeineren Typen als Basistyp eines Arrays.	Die Schülerinnen und Schüler • ermitteln bei der Analyse einfacher Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • modellieren Klassen mit ihren Attributen, ihren Methoden und Assoziationsbeziehungen (M), • modellieren Klassen unter Verwendung von Vererbung (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare Datensammlungen zu (M), • ordnen Klassen, Attributen und Methoden ihren Sichtbarkeitsbereich zu (M),

	• stellen den Zustand eines Objekts dar (D), • stellen die Kommunikation zwischen Objekten grafisch dar (M), • stellen Klassen, Assoziations- und Vererbungsbeziehungen in Diagrammen grafisch dar (D), • dokumentieren Klassen durch Beschreibung der Funktionalität der Methoden (D), • analysieren und erläutern eine objektorientierte Modellierung (A), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I). • implementieren einfache Algorithmen unter Beachtung der Syntax und Semantik einer Programmiersprache (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I). • nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung und gemeinsamen Verwendung von Daten unter Berücksichtigung der Rechteverwaltung (K). • nutzen das Internet zur Recherche, zum Datenaustausch und zur Kommunikation (K). • nutzen die im Unterricht eingesetzten Informatiksysteme selbstständig, sicher, zielführend und verantwortungsbewusst (D).

Unterrichtsvorhaben EF-V

Thema: *Klassenentwurf , Aspekte des objektorientierten Designs (OOD) und der objektorientierten Analyse (OOA) am Beispiel verschiedener Entwurfsmöglichkeiten eines Integer-Arrays*

Leitfragen: Wie verteilt man Funktionalitäten und Methoden auf verschiedene Klassen in Projekten, die mehrere Klassen beinhalten? Wann sollte man weitere Klassen definieren anstatt alles in der gleichen Klasse zu platzieren? Wie kann man größere Datenmengen kapseln? Wie kopiert man ein Objekt?

Zeitbedarf: 7 Stunden

Vorhabenbezogene Konkretisierung:

Es werden die Phasen objektorientierten Analyse (OOA), objektorientierten Designs (OOD) und der objekt-orientierten Analyse (OOA) als mögliche Teile der objektorientierten Modellierung (OOM) unterschieden. Es wird exemplarisch mit dem Datentyp Integer gearbeitet, da dieser gut dargestellt werden kann. Für Integer-Arrays werden Klassen erstellt, die verschiedene Hilfsmethoden anbieten, wie Minimum/Maximum-Suche. Die beiden konkurrierenden Möglichkeiten, dass eine Instanz der selbst erstellten Klasse IntegerArray ein Java-Array verwaltet, indem es eine eigene Kopie erstellt und die Referenz darauf nicht preisgibt oder indem es eine Referenz auf ein bereits bestehendes Integer-Array entgegennimmt und daher nicht für die Konsistenz der Inhalte des Arrays verantwortlich sein kann, da die Referenz nach außen bekannt ist, werden mit Vor- und Nachteilen untersucht. Dabei wird die grafische Ausgabe des IntegerArrays teilweise implementiert und in ein vom Lehrer vorgegebenes Framework eingefügt. Dieses Framework kann anschließend auch bei der Untersuchung von Sortierverfahren wiederverwendet werden.

Unterrichtssequenzen	**zu entwickelnde Kompetenzen**
1. Phase den Softwareentwicklung	Die Schülerinnen und Schüler • ermitteln bei der Analyse einfacher Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • modellieren Klassen mit ihren Attributen, ihren Methoden und Assoziationsbeziehungen (M), • modellieren Klassen unter Verwendung von Vererbung (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare Datensammlungen zu (M), • ordnen Klassen, Attributen und Methoden ihren Sichtbarkeitsbereich zu (M), • stellen den Zustand eines Objekts dar (D), • stellen die Kommunikation zwischen Objekten grafisch dar (M), • stellen Klassen, Assoziations- und Vererbungsbeziehungen in Diagrammen gra-
2. Untersuchung und Gegenüberstellung verschiedener Designs einer Klasse zur Kapselung von Arrays und nach Kriterien der objektorientierten Programmierung beurteilt	

	fisch dar (D), • dokumentieren Klassen durch Beschreibung der Funktionalität der Methoden (D), • analysieren und erläutern eine objektorientierte Modellierung (A), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I). • interpretieren Fehlermeldungen und korrigieren den Quellcode (I). • nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung und gemeinsamen Verwendung von Daten unter Berücksichtigung der Rechteverwaltung (K). • nutzen das Internet zur Recherche, zum Datenaustausch und zur Kommunikation (K). • nutzen die im Unterricht eingesetzten Informatiksysteme selbstständig, sicher, zielführend und verantwortungsbewusst (D).

Unterrichtsvorhaben EF-VI

Thema: *Aspekte des objektorientierten Designs (OOD) bei der Modellierung und Implementierung von Klassen- und Objektbeziehungen am Beispiel des Design-Patterns Observer und des Model-View-Controller-Paradigmas*

Leitfragen: Wie kommunizieren Objekte miteinander? Gibt es vordefinierte objektorientierte Standard-Designs?

Zeitbedarf: 8 Stunden

Vorhabenbezogene Konkretisierung:

Die Änderungen an einem Integer-Array werden nicht automatisch in der Visualisierung des Integer-Arrays übernommen. Die Benachrichtigung über Änderungen kann über das Observer-Pattern oder über Polling erfolgen. Am Beispiel des Observerpatterns wird grundsätzlich die Idee der Sammlung verschiedener Design Patterns vermittelt. Das Integer-Array wird in einer mit einem GUI-Builder erzeugten GUI dargestellt.

Unterrichtssequenzen	**zu entwickelnde Kompetenzen**
1. Das Observer-Pattern 2. Darstellen eines Inter-Arrays in einer graphischen Benutzeroberfläche	Die Schülerinnen und Schüler • ermitteln bei der Analyse einfacher Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • modellieren Klassen mit ihren Attributen, ihren Methoden und Assoziationsbeziehungen (M), • modellieren Klassen unter Verwendung von Vererbung (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare Datensammlungen zu (M), • ordnen Klassen, Attributen und Methoden ihren Sichtbarkeitsbereich zu (M), • stellen den Zustand eines Objekts dar (D), • stellen die Kommunikation zwischen Objekten grafisch dar (M), • stellen Klassen, Assoziations- und Vererbungsbeziehungen in Diagrammen grafisch dar (D), • dokumentieren Klassen durch Beschreibung der Funktionalität der Methoden (D), • analysieren und erläutern eine objektorientierte Modellierung (A), • 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). • nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung und gemeinsamen Verwendung von Daten unter Berücksichtigung der Rechteverwaltung (K). • nutzen das Internet zur Recherche, zum Datenaustausch und zur Kommunikation

	(K). • nutzen die im Unterricht eingesetzten Informatiksysteme selbstständig, sicher, zielführend und verantwortungsbewusst (D).

Unterrichtsvorhaben EF-VII

Thema: *Entwurf und Analyse von Sortierverfahren auf Arrays*

Leitfragen: Wie kann man Daten in einem Array sortieren? Wie kann man die Effizienz eines Sortierverfahrens beurteilen? Welche Fälle treten auf?

Zeitbedarf: 6 Stunden
(z. B.: 4h Theorie + 2h Implementierung)

Vorhabenbezogene Konkretisierung:

Nach der Entwicklung verbalen Beschreibung mehrerer Sortierverfahren anhand von Karten, werden die Sortierverfahren Insertionsort und Minsort formal gefasst, als **Struktogramm** dargestellt und schließlich für ein Integer-Array implementiert. Dabei ist zusätzlich auch eine Implementierung unter Verwendung der visualisierten Klasse Interarray aus Unterrichtsvorhaben VII möglich. Anschließend werden die Sortierverfahren auf Laufzeit- und Platzkomplexität untersucht unter Beachtung des Worst-, Average- und des Bestcase. Eine Einschätzung der Größenordnung erfolgt mittels Groß-O-Notation, die anhand sehr einfacher Beispiele mittels Limes eingeführt wird. Die O-Notation wird schließlich verwendet, um die Ordnung sämtlicher bereits behandelte Algorithmen (Suche, Minimum/Maximum, Sortieren) zu klassifizieren.

Unterrichtssequenzen	**zu entwickelnde Kompetenzen**
1. Entwicklung, Analyse (Zeitkomplexität & Platzkomplexität) und Implementation von Insertionsort und Minsort	Die Schülerinnen und Schüler • ermitteln bei der Analyse einfacher Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • modellieren Klassen mit ihren Attributen, ihren Methoden und Assoziationsbeziehungen (M), • modellieren Klassen unter Verwendung von Vererbung (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare Datensammlun-
2. O-Notation Übungsfunktionen und Analyse bereits bekannter linearer Suche	

	gen zu (M), • ordnen Klassen, Attributen und Methoden ihren Sichtbarkeitsbereich zu (M), • stellen den Zustand eines Objekts dar (D), • stellen die Kommunikation zwischen Objekten grafisch dar (M), • stellen Klassen, Assoziations- und Vererbungsbeziehungen in Diagrammen grafisch dar (D), • dokumentieren Klassen durch Beschreibung der Funktionalität der Methoden (D), • analysieren und erläutern eine objektorientierte Modellierung (A), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I). • analysieren und erläutern einfache Algorithmen und Programme (A), • modifizieren einfache Algorithmen und Programme (I), • entwerfen einfache Algorithmen und stellen sie umgangssprachlich und grafisch dar (M), • implementieren Algorithmen unter Verwendung von Variablen und Wertzuweisungen, Kontrollstrukturen sowie Methodenaufrufen (I), • testen Programme schrittweise anhand von Beispielen (I). • analysieren Such- und Sortieralgorithmen und wenden sie auf Beispiele an (D), • entwerfen einen weiteren Algorithmus zum Sortieren (M), • beurteilen die Effizienz von Algorithmen am Beispiel von Sortierverfahren hinsichtlich Zeitaufwand und Speicherplatzbedarf (A). • implementieren einfache Algorithmen unter Beachtung der Syntax und Semantik einer Programmiersprache (I), • interpretieren Fehlermeldungen und kor-

	rigieren den Quellcode (I). • nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung und gemeinsamen Verwendung von Daten unter Berücksichtigung der Rechteverwaltung (K). • nutzen das Internet zur Recherche, zum Datenaustausch und zur Kommunikation (K). • nutzen die im Unterricht eingesetzten Informatiksysteme selbstständig, sicher, zielführend und verantwortungsbewusst (D).

Unterrichtsvorhaben EF-VIII

Thema: *Generische Programmierung*

Leitfragen: Wie lässt sich eine Datensammlung auf Array-Basis typsicher gestalten? Wie lässt sich die Sortierung beliebiger Daten in einem Array implementieren?

Zeitbedarf: 5 Stunden

Vorhabenbezogene Konkretisierung:

Es wird das Konzept der Generics in Java anhand einer Datensammlung, die auf Arrays beruht eingeführt, einer Klasse ObjectArray, die über Methoden verfügt, die über ein einfaches Array hinausgehen. Da Typsicherheit bei Lesenden Zugriffen nicht gewährleistet ist, werden Generics als Lösung gefunden. Anhand des generischen Interfaces ComparableContent aus den Materialien für das Zentralabitur ab 2017 werden generische Klassen für Adressen implementiert, die angeordnet werden können und auf die somit die zuvor entwickelten und untersuchten Sortierverfahren angewendet werden können.

Unterrichtssequenzen	**zu entwickelnde Kompetenzen**
1. Generics vs. Vererbung Konkretisierung mit new	Die Schülerinnen und Schüler • ermitteln bei der Analyse einfacher Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • modellieren Klassen mit ihren Attributen, ihren Methoden und Assoziationsbeziehungen (M),
2. Konkretisierung mit implements/extends	

	• modellieren Klassen unter Verwendung von Vererbung (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), • stellen den Zustand eines Objekts dar (D), • stellen Klassen, Assoziations- und Vererbungsbeziehungen in Diagrammen grafisch dar (D), • dokumentieren Klassen durch Beschreibung der Funktionalität der Methoden (D), • analysieren und erläutern eine objektorientierte Modellierung (A), • 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). • nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung und gemeinsamen Verwendung von Daten unter Berücksichtigung der Rechteverwaltung (K). • nutzen das Internet zur Recherche, zum Datenaustausch und zur Kommunikation (K). • nutzen die im Unterricht eingesetzten Informatiksysteme selbstständig, sicher, zielführend und verantwortungsbewusst (D).

Unterrichtsvorhaben EF-IX

Thema: 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?*

Zeitbedarf: 5 Stunden

(Z. B.: 3h Geschichte + 2h Datenschutz)

Vorhabenbezogene Konkretisierung:

Das folgende Unterrichtsvorhaben stellt den Abschluss der Einführungsphase dar. Schülerinnen und 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. Die Schülerinnen und Schüler sollen dabei mit Unterstützung des Lehrenden selbstständige Recherchen zu ihren Themen anstellen und auch eine sinnvolle Eingrenzung ihres Themas vornehmen.

Anschließend wird auf den Aspekt des Datenschutzes eingegangen. Dazu werden ausgewählte Auszüge des Bundesdatenschutzgesetzes betrachtet. Es steht keine formale juristische Bewertung von Beispielsituationen im Vordergrund, die im Rahmen eines Informatikunterrichts auch nicht geleistet werden kann, sondern vielmehr eine persönliche Einschätzung praktischer Beispiele im Geiste des Datenschutzgesetzes.

Themenbeispiele:

- Geschichte der Computermodelle
- Datenspeicher (Festplatte, Datasette, Diskette, Lochkarte etc.)
- Datenübertragung (Bluetooth, Handynetz-Generationen, Festnetz etc.)

Unterrichtssequenzen	zu entwickelnde Kompetenzen
1. Selbstständige Erarbeitung von The-men durch die Schülerinnen und Schüler (a) Mögliche Themen zur Erarbeitung in Kleingruppen: – Geschichte der Computermodelle – Datenspeicher (Festplatte, Datasette, Diskette, Lochkarte etc.) – Datenübertragung (Bluetooth, Handy-	Die Schülerinnen und Schüler • bewerten anhand von Fallbeispielen die Auswirkungen des Einsatzes von Informatiksystemen (A), • erläutern wesentliche Grundlagen der Geschichte der digitalen Datenverarbei-tung (A),

netz-Generationen, Festnetz etc.) “ (b) Vorstellung und Diskussion durch Schülerinnen und Schüler	• nutzen das Internet zur Recherche, zum Datenaustausch und zur Kommunikation. (K). • nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung und gemeinsamen Verwendung von Daten unter Berücksichtigung der Rechteverwaltung (K).
2. Vertiefung des Themas Datenschutz (a) Erarbeitung grundlegender Begriffe des Datenschutzen (b) Problematisierung und Anknüpfung an die Lebenswelt der Schülerinnen und Schüler (c) Diskussion und Bewertung von Fallbeispielen aus dem Themenbereich „Datenschutz“	

3.1.3.2 Konkretisierte Kompetenzerwartungen am Ende der 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 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).

Lernmittel / Materialien:

Die üblichen Lernmittel und Materialien Arbeitsblätter, Tafelbilder, Folien, etc. werden passend zum Unterricht eingesetzt und werden im Folgenden nicht extra erwähnt. Lediglich besondere Lernmittel und Materialien werden aufgeführt.

Unterrichtsvorhaben Q1-I

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

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

Zeitbedarf: 14 Stunden (z. B. Queue 4 + Stack 4 + List 6)

Vorhabenbezogene Konkretisierung:

Anhand einer Anwendung, in der Daten nach dem First-In-First-Out-Prinzip verwaltet werden, werden die Operationen der Datenstruktur Schlange thematisiert und die entsprechende Abiturklasse Queue verwendet.

Nach Analyse einer Problemstellung in einem geeigneten Anwendungskontext, in dem Daten nach dem Last-In-First-Out-Prinzip verwaltet werden, werden der Aufbau von Stapeln am Beispiel dargestellt und die Operationen der Klasse Stack anhand der Abiturklasse erläutert. Anschließend werden für die Anwendung notwendige Klassen modelliert und implementiert.

Um einfacher an Objekte zu gelangen, die zwischen anderen gespeichert sind, wird die Klasse List gemäß der Abiturklasse 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.

Besondere Lernmittel / Materialien:

- die IDE Netbeans

Unterrichtssequenzen	**zu entwickelnde Kompetenzen**
1. Die Datenstruktur Schlange im Anwendungskontext unter Nutzung der Klasse Queue • Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen • Erarbeitung der Funktionalität der Klasse Queue • Modellierung und Implementierung der Anwendung unter Verwendung der Klasse Queue. • Implementierung der Klasse Queue	Die Schülerinnen und Schüler • ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D), • modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu (M), • ordnen Klassen, Attributen und Methoden
2. Die Datenstruktur Stapel im Anwendungskontext unter Nutzung der Klasse Stack • Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und	

Operationen • Erarbeitung der Funktionalität der Klasse Stack • Modellierung und Implementierung der Anwendung unter Verwendung der Klasse Stack. • Implementierung der Klasse Stack	ihre Sichtbarkeitsbereiche zu (M), • stellen die Kommunikation zwischen Objekten grafisch dar (D), • stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D), • dokumentieren Klassen (D), • analysieren und erläutern objektorientierte Modellierungen (A), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I). • analysieren und erläutern Algorithmen und Programme (A), • modifizieren Algorithmen und Programme (I), • stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D), • implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • testen Programme systematisch anhand von Beispielen (I). • erläutern Operationen dynamischer (linearer oder/und nicht-linearer) Datenstrukturen (A), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I),
3. Die Datenstruktur lineare Liste im Anwendungskontext unter Nutzung der Klasse List • Erarbeitung der Vorteile der Klasse List im Gegensatz zu den bereits bekannten linearen Strukturen • Modellierung und Implementierung einer kontextbezogenen Anwendung unter Verwendung der Klasse List. • Implementierung ausgewählter Metoden der Klasse List	
4. Vertiefung / Anwendung einer linearen Datenstruktur im Anwendungskontext.	zusätzlich: Die Schülerinnen und Schüler • modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M), • verwenden bei der Modellierung geeigneter Problemstellungen Möglichkeiten der Polymorphie (M),

Unterrichtsvorhaben Q1-II

Thema: *Rekursive Programmierstrategien*

Leitfragen: Was ist eine rekursive Methode? Wie schnell wächst die Anzahl der Methoden-Inkarnationen bei verschiedenen Rekursionstypen? Wie sieht die Aufrufstruktur (z. B. Baumartig) einer rekursiven Methode aus?

Zeitbedarf: 6 Stunden

Vorhabenbezogene Konkretisierung:

Zunächst wird das Programmierparadigma Rekursion vorgestellt. Anschließend werden einige kleinere Algorithmen rekursiv formuliert, wie das schnelle Potenzieren oder die Maximumsuche. Anhand einiger weiterer Beispiele werden die Aufrufstrukturen linear-rekursive Methoden, baumartiger Rekursion und vernesteter Rekursion gezeichnet. Abschließend wird die exponentielle Zunahme der Anzahl der Methoden-Inkarnationen mit wachsender Rekursionstiefe sowohl durch Laufzeittests als auch durch das Zeichnen der Aufrufstrukturen untersucht.

Lernmittel / Materialien:

- die IDE Netbeans

Unterrichtssequenzen	**zu entwickelnde Kompetenzen**
1. Entwickeln eigener rekursiver Methoden 2. Untersuchung von Aufstruktur und Auswirkungen auf die Anzahl der Methodeninkarnationen	Die Schülerinnen und Schüler • modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu (M), • ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M), • dokumentieren Klassen (D), • analysieren und erläutern objektorientierte Modellierungen (A), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I). • stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D), • implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • testen Programme systematisch anhand

	von Beispielen (I). • beurteilen die Effizienz von Algorithmen unter Berücksichtigung des Speicherbedarfs und der Zahl der Operationen (A), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I),

Unterrichtsvorhaben Q1-III

Thema: *Bäume (als spezieller Graph und als dynamische, nichtlineare Datenstruktur)*

Leitfragen: Wie können Daten im Anwendungskontext mit Hilfe binärer Baumstrukturen verwaltet werden? Wie kann dabei der rekursive Aufbau der Baumstruktur genutzt werden?

Zeitbedarf: 9 Stunden
(z. B. 2h (Grundbegriffe) + 7h (Traversierung und Programmierung, etc.))

Vorhabenbezogene Konkretisierung:

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

Anschließend werden für eine Problemstellung in einem der Anwendungskontexte Klassen modelliert und implementiert. Dabei werden die Operationen der Datenstruktur Binärbaum thematisiert und die entsprechende Klasse BinaryTree der 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.

Lernmittel / Materialien:

- die IDE Netbeans

Unterrichtssequenzen	zu entwickelnde Kompetenzen
1. Analyse von Baumstrukturen in verschiedenen Kontexten	Die Schülerinnen und Schüler

• Grundlegende Begriffe (Grad, Tiefe, Höhe, Blatt, Inhalt, Teilbaum, Ebene, Vollständigkeit) • Aufbau und Darstellung von binären Bäumen anhand von Baumstrukturen in verschiedenen Kontexten	• ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D), • modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (M), • modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie/oder lineare und nichtlineare Datensammlungen zu (M), • verwenden bei der Modellierung geeigneter Problemstellungen Möglichkeiten der Polymorphie (M), • ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M), • stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D), • dokumentieren Klassen (D), • analysieren und erläutern objektorientierte Modellierungen (A), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I). • analysieren und erläutern Algorithmen und Programme (A), • modifizieren Algorithmen und Programme (I), • stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D), • erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A), • implementieren und erläutern iterative und rekursive Such- und Sortierverfahren (I), • implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • testen Programme systematisch anhand von Beispielen und mit Hilfe von Testanwendungen (I).
2. Die Datenstruktur Binärbaum im Anwendungskontext unter Nutzung der Klasse BinaryTree • Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen im Anwendungskontext • Erarbeitung der Klasse BinaryTree und beispielhafte Anwendung der Operationen • Implementierung einer Anwendung unter Verwendung der Klasse • Traversierung eines Binärbaums im Pre-, In-, Post- und Levelorder-durchlauf • Implementierung ausgewählter Standardoperationen der Klasse BinaryTree	

Unterrichtsvorhaben Q1-IV

Thema: *Suchen auf linearen Datenstrukturen*

Leitfragen: Wie können Daten effizient auf in linearen Datenstrukturen gesucht werden? Was ist der Vorteil von wahlfreiem gegenüber sequentiellen Zugriff?

Zeitbedarf: 2 Stunden

Vorhabenbezogene Konkretisierung:

Anhand von sortierten Beispieldaten werden die lineare und binäre Suche untersucht und die Laufzeit in Worst-, Best- und Avg-Case analysiert. Anschließend wird der Fall von nicht-sortierten Datensätzen sowie die Suche auf Listen untersucht.

Lernmittel / Materialien:

- die IDE Netbeans

Unterrichtssequenzen	**zu entwickelnde Kompetenzen**
1. Suche in sortierten Daten in Arrays 2. Suche in nicht angeordneten oder nicht sortierten Daten in Arrays und Suche in Listen	Die Schülerinnen und Schüler • stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu (M), • ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M), • dokumentieren Klassen (D), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I). • analysieren und erläutern Algorithmen und Programme (A), • modifizieren Algorithmen und Programme (I), • stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D), • implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • testen Programme systematisch anhand von Beispielen (I). • implementieren und erläutern iterative und rekursive Such- und Sortierverfahren (I),

	• beurteilen die Effizienz von Algorithmen unter Berücksichtigung des Speicherbedarfs und der Zahl der Operationen (A), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung von Dateien unter Berücksichtigung der Rechteverwaltung (K), • wenden eine didaktisch orientierte Entwicklungsumgebung zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I).

Unterrichtsvorhaben Q1-V

Thema: *Suchen mit binären Suchbäumen*

Leitfragen: Wie kann dabei der rekursive Aufbau der Baumstruktur genutzt werden? Welche Vor- und Nachteile haben Suchbäume für die geordnete Verwaltung von Daten?

Zeitbedarf: 14 Stunden
(z. B. 2h (Suchen und Aufbau) + 6h (ausgewählte Implementierung und Verwendung) + 6h (theoretische und empirische Laufzeitanalyse in verschiedenen Fällen)

Vorhabenbezogene Konkretisierung:

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 weitere Klassen oder Methoden in diesem Kontext modelliert und implementiert. Die Suchbäume werden wie zuvor auch grafisch dargestellt. Nach einer Verwendung der Klasse BinarySearchTree mit Klassen, die das generische Interface ComparableContent verwenden, wird eine theoretische Laufzeit-Analyse bei Suchanfragen im optimalen Suchbaum und entarteten Suchbaum durchgeführt. Anschließend wird durch Simulation die Ordnung der Such-Anfragen im zufälligen Baum ermittelt. Insgesamt werden daher die folgenden 9 Fälle unterschieden:

Best-Case-Suchanfrage, Avg.-Case-Suchanfrage und Worst-Case-Suchanfrage jeweils beim optimalen Suchbaum, beim Avg-Case-Baum (zufälligen Baum) und beim entarteten Baum

Lernmittel / Materialien:

- die IDE Netbeans

Unterrichtssequenzen	**zu entwickelnde Kompetenzen**
1. Suche in Suchbäumen 2. Ausgewählte Implementierungen und Entwickeln von Klassen, die das genirische Interface ComparableContent verwenden 3. Aufwandsanalyse des Suchens: Worst-, Avg.- und Best-Case-Suchanfrage in Worst-, Avg.- und Best-Case Suchbaum	Die Schülerinnen und Schüler • ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D), • modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (M), • modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu (M), • verwenden bei der Modellierung geeigneter Problemstellungen Möglichkeiten der Polymorphie (M), • ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M), • stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D), • dokumentieren Klassen (D), • analysieren und erläutern objektorientierte Modellierungen (A), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I). • analysieren und erläutern Algorithmen und Programme (A), • modifizieren Algorithmen und Programme (I),

	• stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D), • implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • testen Programme systematisch anhand von Beispielen (I). • erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A), • implementieren und erläutern iterative und rekursive Such- und Sortierverfahren (I), • beurteilen die Effizienz von Algorithmen unter Berücksichtigung des Speicherbedarfs und der Zahl der Operationen (A), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • nutzen bereitgestellte Informatiksysteme und das Internet reflektiert zur Erschließung, Aufbereitung und Präsentation fachlicher Inhalte (D), • nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung von Dateien unter Berücksichtigung der Rechteverwaltung (K), • wenden eine didaktisch orientierte Entwicklungsumgebung zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I).

Unterrichtsvorhaben Q1-VI

Thema: *Sortierverfahren auf Listen*

Leitfragen: Wie kann eine Liste zeiteffizient sortiert werden?

Zeitbedarf: 8 Stunden

(z. B.: 2h (Implementation der bereits bekannten Sortierverfahren) + 1h (BinaryTreeSort) + 3 (Entwicklung und Analyse von Merge- und Quicksort auf Listen) + 2h das Divide&Conquer-Paradigma)

Vorhabenbezogene Konkretisierung:

Zunächst werden die bereits aus der EF bekannten Sortierverfahren auf Arrays, Insertionsort und Minsort, auf Listen übertragen und Auswirkungen der Veränderung der Datenstruktur auf die Zeit- und Platzkomplexität untersucht. Anschließend werden die Ergebnisse aus dem vorhergehenden Unterrichtsvorhaben verwendet, um direkt das Verfahren BinaryTreeSort, bei dem sämtliche Elemente aus der Liste entnommen und in einen BinaryTree eingefügt werden, zu entwickeln, die Zeit- & Platzkomplexität zu analysieren und es zu implementieren. Schließlich wird das Divide- & Conquer-Paradigma am Beispiel von Quick- und Mergesort erarbeitet. Die Verfahren werden bzgl. Aufwand durch Argumentation am Rekursionsbaum (=baumartige rekursive Aufrufstruktur) analysiert und dabei auf Parallelen von Quicksort und BinaryTreeSort zurückgegriffen. Das zugrundeliegende Algorithmenentwurfsparadigma Divide&Conquer (Teile und herrsche) wird an diesen beiden Beispiel-Algorithmen als verallgemeinerter Ansatz herausgestellt.

Lernmittel / Materialien:

- die IDE Netbeans

Unterrichtssequenzen	**zu entwickelnde Kompetenzen**
1. Übertragung der von Arrays bekannten Sortierverfahren (Insertionsort und Minsort) auf Listen 2. Entwicklung, Analyse und Implementation von BinaryTreeSort 3. Das Divide- & Conquer-Paradigma am Beispiel von Merge- und Quicksort	Die Schülerinnen und Schüler • stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D), • modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (M), • modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu (M), • verwenden bei der Modellierung geeigneter

4. Laufzeitanalyse von Merge- und Quicksort	Problemstellungen Möglichkeiten der Polymorphie (M), • ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M), • stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D), • dokumentieren Klassen (D), • analysieren und erläutern objektorientierte Modellierungen (A), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I). • analysieren und erläutern Algorithmen und Programme (A), • modifizieren Algorithmen und Programme (I), • stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D), • entwickeln iterative und rekursive Algorithmen unter Nutzung der Strategien „Modularisierung“ und „Teilen und Herrschen“ (M), • implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • testen Programme systematisch anhand von Beispielen (I). • erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A), • implementieren und erläutern iterative und rekursive Such- und Sortierverfahren (I), • beurteilen die Effizienz von Algorithmen unter Berücksichtigung des Speicherbedarfs und der Zahl der Operationen (A), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • interpretieren Fehlermeldungen und korri-

	gieren den Quellcode (I), • nutzen bereitgestellte Informatiksysteme und das Internet reflektiert zur Erschließung, Aufbereitung und Präsentation fachlicher Inhalte (D), • nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung von Dateien unter Berücksichtigung der Rechteverwaltung (K), • wenden eine didaktisch orientierte Entwicklungsumgebung zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I).

Unterrichtsvorhaben Q1-VII

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?*

Zeitbedarf: 22 Stunden

(z. B.: 10h Nutzung + 10h Entwurf + 2h Einblick in JDBC-Anbindung)

Vorhabenbezogene Konkretisierung:

Ausgehend von einer vorhandenen Datenbank entwickeln Schülerinnen und Schüler für sie relevante Fragestellungen, die mit dem vorhandenen Datenbestand beantwortet werden sollen. Zur Beantwortung dieser Fragestellungen wird die vorgegebene Datenbank von den Schülerinnen und Schülern analysiert und die notwendigen Grundbegriffe für Datenbanksysteme sowie die erforderlichen SQL-Abfragen werden erarbeitet.

In anderen Anwendungskontexten müssen Datenbanken erst noch entwickelt werden, um Daten zu speichern und Informationen für die Beantwortung von möglicherweise auftretenden Fragen zur Verfügung zu stellen. Dafür ermitteln Schülerinnen und Schüler in den Anwendungssituationen Entitäten, zugehörige Attribute, Relationen und Kardinalitäten und stellen diese in Entity-Relationship-Modellen dar. Entity-Relationship-Modelle werden interpretiert und erläutert, modifiziert und in Datenbankschemata überführt. Mit Hilfe von SQL-Anweisungen können anschließend im Kontext relevante Informationen aus der Datenbank extrahiert werden.

Ein Entity-Relationship-Diagramm kann auch verwendet werden, um die Entitäten inklusive ihrer Attribute und Relationen in einem vorgegebenen Datenbankschema darzustellen.

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

Lernmittel / Materialien:

- die IDE Netbeans
- Datenbankserver (z. B. XAMPP)
- SQL-Tutorials (z. B. die Seite SQL-Zoo, VideoCenter-Datenbank http://dokumentation.videocenter.schule.de/old/video/index.html)

Unterrichtssequenzen	**zu entwickelnde Kompetenzen**
1. Nutzung von relationalen Datenbanken • 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 • Relationenalgebra (RA) und RA-Anfragen-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: +, -, *, /, (…), Vergleichsoperato-	• ermitteln für anwendungsbezogene Problemstellungen Entitäten, zugehörige Attribute, Relationen und Kardinalitäten (M), • stellen Entitäten mit ihren Attributen und die Beziehungen zwischen Entitäten mit Kardinalitäten in einem Entity-Relationship-Diagramm grafisch dar (D), • modifizieren eine Datenbankmodellierung (M), • modellieren zu einem Entity-Relationship-Diagramm ein relationales Datenbankschema (M), • bestimmen Primär- und Sekundärschlüssel (M), • analysieren und erläutern eine Datenbankmodellierung (A), • erläutern die Eigenschaften normalisierter Datenbankschemata (A), • überprüfen Datenbankschemata auf vorgegebene Normalisierungseigenschaften (D), • überführen Datenbankschemata in die 1. bis 3. Normalform (M). • ermitteln Ergebnisse von Datenbankabfragen über mehrere verknüpfte Tabellen (D). • nutzen die Syntax und Semantik einer Pro-

ren: =, <>, >, <, >=, <=, LIKE, BETWEEN, IN, IS NULL)	grammiersprache bei der Implementierung und zur Analyse von Programmen (I), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • analysieren und erläutern die Syntax und Semantik einer Datenbankabfrage (A), • verwenden die Syntax und Semantik einer Datenbankabfragesprache, um Informationen aus einen Datenbanksystem zu extrahieren (I). • nutzen bereitgestellte Informatiksysteme und das Internet reflektiert zur Erschließung, Aufbereitung und Präsentation fachlicher Inhalte (D), • nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung von Dateien unter Berücksichtigung der Rechteverwaltung (K), • wenden eine didaktisch orientierte Entwicklungsumgebung zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I), • erläutern Eigenschaften, Funktionsweisen und Aufbau von Datenbanksystemen unter dem Aspekt der sicheren Nutzung (A),
2. Modellierung von relationalen Datenbanken • 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 • Entwicklung einer Datenbank aus einem Datenbankentwurf: Modellierung eines relationalen Datenbankschematas zu einem Entity-Relationship-Diagramm inklusive der Bestimmung von Primär- und Sekundärschlüsseln Redundanz, Konsistenz und Normalformen: Untersuchung einer Datenbank hinsichtlich Konsistenz und Redundanz in einer Anwendungssituation unter dem Aspekt der sicheren Nutzung. Überprüfung von Datenbankschemata hinsichtlich der 1. bis 3. Normalform und Normalisierung (um Redundanzen zu vermeiden und Konsistenz zu gewährleisten)	
3. Daten-Abfrage aus einer Datenbank mit Java und JDBC	

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 regulären Sprachen, endlichen Automaten und regulären Grammatiken?

Zeitbedarf: 18 Stunden

(z. B. 6h Endliche Automaten mit Darstellungen und Programmierung + 6h Grammatiken und Sprachen und Überführungen + 6h nicht-reguläre Sprachen und Grenzen von endlichen Automaten)

Vorhabenbezogene Konkretisierung:

Zunächst werden deterministische endliche Automaten (DEA) und nichtdeterministische endliche Automaten (NEA) in Beispielkontexten vorgestellt und anschließend formal definiert. Die Übergangsfunktion wird mittels Transitionstabelle und Wertetabelle dargestellt, eine Darstellungsform des Automaten als Transitionsgraph wird verwendet. Zudem wird ein DEA als Java-Programm implementiert, jedoch kein NEA.

Anschließend werden formale Sprachen und phrasenstruktur-Grammatiken (Typ-0) definiert. Anschließend werden reguläre Grammatiken definiert und in Beispielen untersucht sowie die Überführungen

- reguläre Grammatik <-> NEA
- NEA <-> DEA

erarbeitet. Reguläre Ausdrücke als weitere Darstellung regulärer Sprachen sind möglich, aber nicht obligatorisch.

Schließlich wird gezeigt, dass es nicht-reguläre Sprachen gibt und es werden Grenzen des Akzeptierens mit endlichen Automaten aufgezeigt. In diesem Zusammenhang wird das Pumping-Lemma auf einige Beispiel-Sprachen angewendet und seine Begründung visuell am Automaten nachvollzogen.

Lernmittel / Materialien:

- eine didaktische Entwicklungsumgebung ATOCC
- die DIE Netbeans

Unterrichtssequenzen	**zu entwickelnde Kompetenzen**
1. DEAs und NEAs	Die Schülerinnen und Schüler • modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (M), • modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu (M),
2. Formale Sprachen, Typ-0-Grammatiken, reguläre Grammatiken und Sprachen, Umwandlungen der Darstellungsformen ineinander (Automat, Grammatik, reg. Ausdrücke)	
3. Nicht-reguläre Sprachen, Grenzen endlicher Akzeptoren	

	• verwenden bei der Modellierung geeigneter Problemstellungen Möglichkeiten der Polymorphie (M), • ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M), • stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D), • dokumentieren Klassen (D), • analysieren und erläutern objektorientierte Modellierungen (A), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I). • testen Programme systematisch anhand von Beispielen (I). • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • analysieren und erläutern die Eigenschaften endlicher Automaten einschließlich ihres Verhaltens bei bestimmten Eingaben (A), • ermitteln die Sprache, die ein endlicher Automat akzeptiert (D), • entwickeln und modifizieren zu einer Problemstellung endliche Automaten (M), • stellen endliche Automaten in Tabellen oder Graphen dar und überführen sie in die jeweils andere Darstellungsform (D), • entwickeln zur Grammatik einer regulären Sprache einen zugehörigen endlichen Automaten (M). • analysieren und erläutern Grammatiken regulärer Sprachen (A), • modifizieren Grammatiken regulärer Sprachen (M),

	• ermitteln die formale Sprache, die durch eine Grammatik erzeugt wird (A), • entwickeln zu einer regulären Sprache eine Grammatik, die die Sprache erzeugt (M), • entwickeln zur akzeptierten Sprache eines Automaten eine zugehörige Grammatik (M), • beschreiben an Beispielen den Zusammenhang zwischen Automaten und Grammatiken (D). • zeigen die Grenzen endlicher Automaten und regulärer Grammatiken im Anwendungszusammenhang auf (A). • nutzen bereitgestellte Informatiksysteme und das Internet reflektiert zur Erschließung, Aufbereitung und Präsentation fachlicher Inhalte (D), • nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung von Dateien unter Berücksichtigung der Rechteverwaltung (K), • wenden eine didaktisch orientierte Entwicklungsumgebung zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I).

Unterrichtsvorhaben Q2-II

Thema: *Grenzen der Automatisierbarkeit*

Leitfragen: *Welche weiteren Sprach-Typen werden in der Chomsky-Hierarchie unteerschieden? Was ist das Wortproblem? Wie wirkt sich das Ergänzen eines endlichen Automaten um eine potentiell-unendliche Liste/Band auf die Mächtigkeit des Akzeptierens von Sprachen aus? Ist das Wortproblem für Typ-0-Sprachen entscheidbar und was ist Berechenbarkeit? Welche Möglichkeiten bieten Informatiksysteme und wo liegen ihre Grenzen?*

Zeitbedarf: 16 Stunden
(z. B.: 2h Chomsky-Hierarchie mit Beispielen + 2h Turing-Maschine als Akzeptor + 6h Berechenbarkeit mit TMs und nicht-berechenbare Probleme + 6h Grenzen praktischer „Berechenbarkeit“ bei Backtracking-Algorithmen und zugehörigen harten Problemen)

Vorhabenbezogene Konkretisierung:

Nachdem im vorhergehenden Unterrichtsvorhaben festgestellt wurde, dass es Sprachen gibt, die nicht regulär sind, wird nun die Chomsky-Hierarchie vorgestellt und an einigen ausgewählten Beispiel-Sprachen untersucht. Um auch kompliziertere Sprachen erkennen zu können, wird die Endlichkeit der Automaten aufgehoben, indem das Modell der Turing-Maschine (TM) betrachtet wird und einige TMs zu einigen Beispiel-Sprachen aufgestellt werden.

Die Semi-Entscheidbarkeit wird an einem Sprachbeispiel entdeckt und anschließend die Berechenbarkeit mit TMs definiert. Schließlich wird anhand eines konkreten nicht-berechenbaren Problems (z. B. Halteproblem), dessen nicht-berechenbarkeit beweisen wird gezeigt, dass es nicht-berechenbare Funktionen gibt, sowie der Begriff „intuitive Berechenbarkeit“ und die Church’sche These vorgestellt.

Eine Grenze „praktischer Berechenbarkeit“ wird schließlich anhand eines Problems, für das nur ein exponentieller Algorithmus, der das Backtracking-Paradigma verwendet, bekannt ist, als weitere Grenze von Informatiksystemen und als Begründung für die Bedeutung der Suche nach approximierenden Algorithmen (für NP-harte Probleme, der Begriff wird aber nicht definiert) untersucht.

Lernmittel / Materialien:

- eine didaktische Entwicklungsumgebung ATOCC

Unterrichtssequenzen	**zu entwickelnde Kompetenzen**
1. (a) Weitere Sprach-Typen gemäß Chomsky-Hierarchie (b) Akzeptieren von Sprachen mittels TM / Definieren einer TM	Die Schülerinnen und Schüler • ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (M), • modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu (M), • verwenden bei der Modellierung geeigneter Problemstellungen Möglichkeiten der Polymorphie (M), • ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M),
2. Turingberechenbarkeit und Church’sche These	
3. Grenzen praktischer Berechenbarkeit am Beispiel eines Problems, für das der beste bekannte Algorithmus das Backtracking-Paradigma verwendet und exponentielle Laufzeit hat.	

	• stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D), • dokumentieren Klassen (D), • analysieren und erläutern objektorientierte Modellierungen (A), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I). • analysieren und erläutern Algorithmen und Programme (A), • modifizieren Algorithmen und Programme (I), • stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D), • entwickeln iterative und rekursive Algorithmen unter Nutzung der Strategien „Modularisierung“ und „Teilen und Herrschen“ (M), • implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • testen Programme systematisch anhand von Beispielen (I). • beurteilen die Effizienz von Algorithmen unter Berücksichtigung des Speicherbedarfs und der Zahl der Operationen (A), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • analysieren und erläutern die Eigenschaften endlicher Automaten einschließlich ihres Verhaltens bei bestimmten Eingaben (A), • entwickeln und modifizieren zu einer Problemstellung endliche Automaten (M), • analysieren und erläutern Grammatiken regulärer Sprachen (A),

	• ermitteln die formale Sprache, die durch eine Grammatik erzeugt wird (A), • entwickeln zur akzeptierten Sprache eines Automaten eine zugehörige Grammatik (M), • beschreiben an Beispielen den Zusammenhang zwischen Automaten und Grammatiken (D). • zeigen die Grenzen endlicher Automaten und regulärer Grammatiken im Anwendungszusammenhang auf (A). • wenden eine didaktisch orientierte Entwicklungsumgebung zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I). • untersuchen und beurteilen Grenzen des Problemlösens mit Informatiksystemen (A).

Unterrichtsvorhaben Q2-III

Thema: *Prinzipielle Arbeitsweise eines Computers*

Leitfragen: *Was sind die strukturellen Hauptbestandteile eines Computers und wie kann man sich die Ausführung eines maschinenahen Programms mit diesen Komponenten vorstellen?*

Vorhabenbezogene Konkretisierung:

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

Zeitbedarf: 7 Stunden
(z. B.: 3h Von-Neumann-Architektur und Hardware-Komponenten + 3h Assembler-Programmierung + 1h Untersuchung von realen Programmen)

Lernmittel / Materialien:

- MOPS oder JOHNNY
- Ein Diskeditor/Hexeditor

Unterrichtssequenzen	**zu entwickelnde Kompetenzen**
1. Prinzipieller Aufbau einer von Neumann-Architektur mit CPU, Rechenwerk, Steuerwerk, Register und Hauptspeicher	Die Schülerinnen und Schüler • nutzen die Syntax und Semantik einer Pro-

2. (a) Maschinennahe Befehlen (Assembler) und ihre Repräsentation in einem Binär-Code (Maschinensprache) (b) Analyse und Erläuterung der Funktionsweise eines einfachen maschinennahen Programms, Schleifen und Verzweigungen	grammiersprache bei der Implementierung und zur Analyse von Programmen (I), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • erläutern die Ausführung eines einfachen maschinennahen Programms sowie die Datenspeicherung auf einer „Von-Neumann-Architektur“ (A), • nutzen bereitgestellte Informatiksysteme und das Internet reflektiert zur Erschließung, Aufbereitung und Präsentation fachlicher Inhalte (D), • nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung von Dateien unter Berücksichtigung der Rechteverwaltung (K), • wenden eine didaktisch orientierte Entwicklungsumgebung zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I).
3. Untersuchung eines realen Programms mit einem Diskeditor	

Unterrichtsvorhaben Q2-IV

Thema: *Sicherheit und Datenschutz in Netzstrukturen*

Leitfragen: *Wie werden Daten in Netzwerken übermittelt? Was sollte man in Bezug auf die Sicherheit beachten? Welche Möglichkeiten der Verschlüsselung gibt es?*

Vorhabenbezogene Konkretisierung:

Zunächst werden Topologien von Netzwerken, eine Client-Server-Struktur, das TCP/IP-Referenzmodell sowie Sicherheitsaspekte und verschiedene symmetrische und asymmetrische kryptografische Verfahren analysiert und erläutert. Fallbeispiele zur Datenschutzproblematik und zum Urheberrecht runden das Unterrichtsvorhaben ab.

Zeitbedarf: 16 Stunden

(Z. B.: 4 Netzwerke + 3h symm. Verschlüsselung + 6h asymm. Verschlüsselung + 3h Datenschutz und Urheberrecht)

Lernmittel / Materialien:

- Whireshark
- Crypttool
- *Ergänzungsmaterialien zum Lehrplannavigator Unterrichtsvorhaben Q1 5* - Datenschutz beim Videocenter, Materialblatt-Datenschutzgesetz (Download Q1-V.2)
- Materialblatt zum Bundesdatenschutzgesetz (Download EF-VI.1)

Unterrichtssequenzen	**zu entwickelnde Kompetenzen**
1. Informationsübertragung in Netzwerken (a) Netzwerk-Toplologien und eine Client-Server-Struktur vs. Peer-to-Peer-Kommunikation (b) Das TCP/IP-Referenzmodell und das POP3-Protokoll als Beispiel	Die Schülerinnen und Schüler • beschreiben und erläutern Netzwerk-Topologien, die Client-Server-Struktur und Protokolle sowie ein Schichtenmodell in Netzwerken (A). • nutzen bereitgestellte Informatiksysteme und das Internet reflektiert zur Erschließung, Aufbereitung und Präsentation fachlicher Inhalte (D), • nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung von Dateien unter Berücksichtigung der Rechteverwaltung (K), • wenden eine didaktisch orientierte Entwicklungsumgebung zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I). • erläutern Eigenschaften, Funktionsweisen und Aufbau von Datenbanksystemen unter dem Aspekt der sicheren Nutzung (A), • analysieren und erläutern Eigenschaften und Einsatzbereiche symmetrischer und asymmetrischer Verschlüsselungsverfahren (A). • untersuchen und bewerten anhand von Fallbeispielen Auswirkungen des Einsatzes von Informatiksystemen sowie Aspekte der Sicherheit von Informatiksystemen, des Datenschutzes und des Urheberrechts (A), • untersuchen und bewerten Problemlagen,
2. (a) Verschlüsselung und Kryptoanalyse Symmetrische und asymmetrische kryptografische Verfahren (Cäsar-, Vigenère-, RSA-Verfahren) als Methoden Daten im Netz verschlüsselt zu übertragen (b) Signatur und Zertifizierung	
3. Fallbeispiele zur Datenschutzproblematik und zum Urheberrecht Sicherheitsproblematik und Datenschutz bei Zugriff auf Datenbanken über das Internet	

	die sich aus dem Einsatz von Informatiksystemen ergeben, hinsichtlich rechtlicher Vorgaben, ethischer Aspekte und gesellschaftlicher Werte unter Berücksichtigung unterschiedlicher Interessenlagen (A).

3.2 Grundsätze der fachmethodischen und fachdidaktischen Arbeit

3.3 Grundsätze der Leistungsbewertung und Leistungsrückmeldung

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

Am Clara-Schumann-Gymnasium liegt zudem ein Leistungskonzept für das Fach Informatik in Form eines separaten Dokuments vor.

3.3.1 Beurteilungsbereich Klausuren

Verbindliche Absprachen:

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

Instrumente:

- Einführungsphase: 1 Klausur je Halbjahr
  Dauer der Klausur: 2 Unterrichtsstunden

- Grundkurse Q 1: 2 Klausuren je Halbjahr
  Dauer der Klausuren: 2 Unterrichtsstunden
- Grundkurse Q 2.1: 2 Klausuren
  Dauer der Klausuren: 3 Unterrichtsstunden

- Grundkurse Q 2.2: 1 Klausur unter Abiturbedingungen
- Anstelle einer Klausur kann gemäß dem Beschluss der Lehrerkonferenz in Q 1.2 eine Facharbeit geschrieben werden.

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

Kriterien

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

Spätestens ab der Qualifikationsphase orientiert sich die Zuordnung der Hilfspunktsummenanteile 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.

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

3.3.2 Beurteilungsbereich Sonstige Mitarbeit

Der Beurteilungsbereich „Mitarbeit im Unterricht“ erfasst die Qualität und Kontinuität der Beiträge, die die Schülerinnen und Schüler im Unterricht erbringen. Diese Beiträge sollen unterschiedliche mündliche und schriftliche Formen in enger Bindung an die Aufgabenstellung, die inhaltliche Reichweite und das Anspruchsniveau der jeweiligen Unterrichtseinheit umfassen.

Bei den mündlichen Leistungen im Unterricht sind zu bewerten:

- Beteiligung am Unterrichtsgespräch
- Zusammenfassungen zur Vor- und Nachbereitung des Unterrichts
- Präsentation von Arbeitsergebnissen
- Mitarbeit in Partner- und Gruppenarbeitsphasen

Neben der Richtigkeit, Vollständigkeit und Komplexität der Gedankengänge sind die der Altersstufe angemessene sprachliche Darstellung und die Verwendung der Fachsprache von Bedeutung.

Bei der Unterrichtsgestaltung sind den Schülerinnen und Schülern hinreichend Möglichkeiten zur Mitarbeit zu eröffnen, z.B. durch

- praktische Leistungen am Computer als Werkzeug im Unterricht (z. B. Implementierung, Test und Anwendung von Informatiksystemen),
- Protokolle und Referate,
- Projektarbeit,
- Lernerfolgsüberprüfungen und schriftliche Übungen.
- Bearbeitung von schriftlichen Aufgaben im Unterricht

Kriterien

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

Die Bewertungskriterien stützen sich auf

- die Qualität der Beiträge,
- die Quantität der Beiträge und
- die Kontinuität der Beiträge.

Besonderes Augenmerk ist dabei auf

- die sachliche Richtigkeit,
- die angemessene Verwendung der Fachsprache,
- die Darstellungskompetenz,

- die Komplexität und den Grad der Abstraktion,
- die Selbstständigkeit im Arbeitsprozess,
- die Präzision und
- die Differenziertheit der Reflexion zu legen.

Bei Gruppenarbeiten auch auf

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

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 Schülerinnen und Schülern transparent gemacht. Leistungsrückmeldungen **können** erfolgen

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

- zu Eltern- oder Schülersprechtagen.

Die Leistungsrückmeldung **kann**

- durch ein Gespräch mit der Schülerin oder 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 Informatik als fortgesetztes Grund- oder Leistungskursfach in der Qualifikationsphase.

4 Entscheidungen zu fach- und unterrichtsübergreifenden Fragen

Zusammenarbeit mit anderen Fächern

Im Informatikunterricht werden Kompetenzen anhand informatischer Inhalte in verschiedenen Anwendungskontexten erworben, in denen Schülerinnen und Schülern aus anderen Fächern Kenntnisse mitbringen können. Diese können insbesondere bei der Auswahl und Bearbeitung von Softwareprojekten berücksichtigt werden und in einem hinsichtlich der informatischen Problemstellung angemessenem Maß in den Unterricht Eingang finden. Aufgrund der engen Verwandtschaft der Informatik zur Mathematik wird diese inhaltlich fachliche Nähe in besonderem Umfang im Unterricht berücksichtigt, um den Schülerinnen und Schülern ein stimmiges Gesamtbild der Informatik zu vermitteln und bei ihnen nicht eine einseitig in Richtung Softwareentwicklung und IT verzerrte Vorstellung zu erzeugen.

Vorbereitung auf die Erstellung der Facharbeit

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

5 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 trägt durch diesen Prozess zur Qualitätsentwicklung und damit zur Qualitätssicherung des Faches bei.

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

Das schulinterne Curriculum (siehe 2.1) ist zunächst bis 2017 für den ersten Durchgang durch die gymnasiale Oberstufe nach Erlass des Kernlehrplanes verbindlich. Spätestens 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 Fachkonferenz Informatik auf der Grundlage ihrer Unterrichtserfahrungen eine Gesamtsicht des schulinternen Curriculums vornehmen und ggf. eine Beschlussvorlage für die erste Fachkonferenz des folgenden Schuljahres erstellen.