Schulinterner Lehrplan für die gymnasiale Oberstufe

Informatik

(Stand: 15.06.2015)

Inhaltsverzeichnis

1 Die Fachgruppe Informatik des Gymnasiums Würselen

Beim Gymnasium Würselen handelt es sich um eine vierzügige Schule im Stadtgebiet von Würselen mit zurzeit ca. 1000 Schülerinnen und Schülern, ca 80 Lehrerinnen und Lehrern.
Das Fach Informatik wird am Gymnasium Würselen ab der Jahrgangsstufe 8 im Wahlpflichtbereich II (WP II) zweistündig unterrichtet und von etwa einem Drittel der Schülerinnen und Schüler besucht. In der zweijährigen Laufzeit dieser Kurse wird in altersstufengerechter Weise unter anderem auf Grundlagen der Algorithmik am Beispiel einer didaktischen Lernumgebung, auf die technische Informatik am Beispiel von Schaltwerken und Schaltnetzen und auf gesellschaftliche Aspekte wie Datensicherheit und Verschlüsselung eingegangen. Angebote der RWTH Aachen im Rahmen des Schülerlabors Infosphere werden regelmäßig besucht.

In der Jahrgangsstufen 5 wird ein für alle verpflichtender Kurs zum Umgang mit informatischen Systemen und unserem LMS Fronter durchgeführt. Ein Mediencurriculum für die gesamte Sekundarstufe I stellt sicher, dass die Kompetenzen im Umgang mit digitalen Werkzeugen durch die Fächer gefördert werden, da dies keine Aufgabe des Fachs Informatik ist.

In der Sekundarstufe II bietet das Gymnasium Würselen für die eigenen Schülerinnen und Schüler in allen Jahrgangsstufen jeweils einen Grundkurs in Informatik an.

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

Der Unterricht der Sekundarstufe II wird auf Basis der Programmiersprache *Java* durchgeführt. In der Einführungsphase kommt dabei zusätzlich eine Lernumgebung wie z.B. *Greenfoot* zum Einsatz, um die Prinzipien der Objektorientierung und die Grundlagen der Algorithmik zu veranschaulichen.

Durch projektartiges Vorgehen, offene Aufgaben und Möglichkeiten, Problemlösungen zu verfeinern oder zu optimieren, entspricht der Informatikunterricht der Oberstufe in besonderem Maße den Erziehungszielen, Leistungsbereitschaft zu fördern, ohne zu überfordern.

Die gemeinsame Entwicklung von Materialien und Unterrichtsvorhaben, die Evaluation von Lehr- und Lernprozessen sowie die stetige Überprüfung und eventuelle Modifikation des schulinternen Curriculums durch die Fachkonferenz Informatik stellen einen wichtigen Beitrag zur Qualitätssicherung und -entwicklung des Unterrichts dar.

Zurzeit besteht die Fachschaft Informatik des Gymnasiums Würselen aus fünf Lehrkräften (zwei davon mit Fakultas für die Sek II), denen ein Computerraum mit 22 Computerarbeitsplätzen und Laptopwagen mit 54 Geräten zur Verfügung stehen. Das Schulgebäude ist mit flächendeckendem WLAN ausgestattet, so dass sowohl

bei der Arbeit im Computerraum als auch mit den mobilen Geräten Zugang zu unserem webbasierten LMS Fronter besteht.

Der Unterricht erfolgt nach dem Doppelstundenprinzip.

2 Entscheidungen zum Unterricht

2.1 Unterrichtsvorhaben

Die Darstellung der Unterrichtsvorhaben im schulinternen Lehrplan besitzt den Anspruch, sämtliche im Kernlehrplan angeführten Kompetenzen abzudecken. Dies entspricht der Verpflichtung jeder Lehrkraft, Schülerinnen und Schülern Lerngelegenheiten zu ermöglichen, so dass alle Kompetenzerwartungen des Kernlehrplans von ihnen erfüllt werden können.

Die entsprechende Umsetzung erfolgt auf zwei Ebenen: der Übersichts- und der Konkretisierungsebene.

Im „Übersichtsraster Unterrichtsvorhaben“ (Kapitel 2.1.1) wird die für alle Lehrerinnen und Lehrer gemäß Fachkonferenzbeschluss verbindliche Verteilung der Unterrichtsvorhaben dargestellt. Das Übersichtsraster dient dazu, den Kolleginnen und Kollegen einen schnellen Überblick über die Zuordnung der Unterrichtsvorhaben zu den einzelnen Jahrgangsstufen sowie den im Kernlehrplan genannten Kompetenzen, Inhaltsfeldern und inhaltlichen Schwerpunkten zu verschaffen. Der ausgewiesene Zeitbedarf versteht sich als grobe Orientierungsgröße, die nach Bedarf über- oder unterschritten werden kann.

Während der Fachkonferenzbeschluss zum „Übersichtsraster Unterrichtsvorhaben“ zur Gewährleistung vergleichbarer Standards sowie zur Absicherung von Lerngruppenübertritten und Lehrkraftwechseln für alle Mitglieder der Fachkonferenz Bindekraft entfalten soll, beinhaltet die Ausweisung „konkretisierter Unterrichtsvorhaben“ (Kapitel 2.1.2) Beispiele und Materialien, die empfehlenden Charakter haben. Referendarinnen und Referendaren sowie neuen Kolleginnen und Kollegen dienen diese vor allem zur standardbezogenen Orientierung in der neuen Schule und zur Verdeutlichung von didaktisch-methodischen Zugängen.

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

2.1.1 Übersichtsraster Unterrichtsvorhaben

I) Einführungsphase

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

Thema:
Einführung in die Nutzung von Informatiksystemen und in grundlegende Begrifflichkeiten

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

Inhaltsfelder:
• Informatiksysteme
• Informatik, Mensch und Gesellschaft

Inhaltliche Schwerpunkte:
• Einzelrechner
• Dateisyst
em
• Internet
• Einsatz von Informatiksystemen

Zeitbedarf: 6 Stunden
</td>
<td>
<u>Unterrichtsvorhaben E-II</u>

Thema:
Grundlagen der objektorientierten Analyse, Modellierung und Implementierung mithilfe einer didaktisch orientierten Entwicklungsumgebung (Greenfoot oder andere)

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

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

Inhaltliche Schwerpunkte:
• Objekte und Klassen
• Syntax und Semantik einer Programmiersprache

Zeitbedarf: 8 Stunden
</td>
</tr>
</table>

Einführungsphase	
Unterrichtsvorhaben E-III **Thema**: *Grundlagen der objektorientierten Programmierung und algorithmischer Grundstrukturen in Java anhand eines Spiels in einer didaktisch orientierten Entwicklungsumgebung (Greenfoot oder andere)* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen und Automaten **Inhaltliche Schwerpunkte:** • Objekte und Klassen • Syntax und Semantik einer Programmiersprache • Analyse, Entwurf und Implementierung einfacher Algorithmen **Zeitbedarf**: 20 Stunden	*Unterrichtsvorhaben E-IV* **Thema**: *Modellierung und Implementierung von Klassen- und Objektbeziehungen anhand eines Anwendungsbeispiels* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen und Automaten **Inhaltliche Schwerpunkte:** • Objekte und Klassen • Syntax und Semantik einer Programmiersprache • Analyse, Entwurf und Implementierung einfacher Algorithmen • Datenstruktur Array **Zeitbedarf**: 18 Stunden

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

Thema:
Such- und Sortieralgorithmen anhand kontextbezogener Beispiele

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

Inhaltsfelder:
• Algorithmen

Inhaltliche Schwerpunkte:
• Algorithmen zum Suchen und Sortieren
• Analyse, Entwurf und Implementierung einfacher Algorithmen

Zeitbedarf: 9 Stunden</td>
<td><u>Unterrichtsvorhaben E-VI</u>

Thema:
Geschichte der digitalen Datenverarbeitung und die Grundlagen des Datenschutzes

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

Inhaltsfelder:
• Informatik, Mensch und Gesellschaft
• Informatiksysteme

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

Zeitbedarf: 15 Stunden</td>
</tr>
<tr><td colspan="2">Summe Einführungsphase: 76</td></tr>
</table>

II) Qualifikationsphase (Q1 und Q2) - Grundkurs

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

Thema:
Suchen und Sortieren in Arrays

Zentrale Kompetenzen:

Argumentieren
Modellieren
Implementieren
Darstellen und Interpretieren
Kommunizieren und Kooperieren

Inhaltsfelder:

Algorithmen
Formale Sprachen und Automaten

Inhaltliche Schwerpunkte:

Analyse, Entwurf und Implementierung von Algorithmen
Algorithmen in ausgewählten informatischen Kontexten
Syntax und Semantik einer Programmiersprache

Zeitbedarf: 12 Stunden
</td>
<td>
<u>Unterrichtsvorhaben Q1-II</u>

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

Zentrale Kompetenzen:

Argumentieren
Modellieren
Implementieren
Darstellen und Interpretieren
Kommunizieren und Kooperieren

Inhaltsfelder:

Daten und ihre Strukturierung
Algorithmen
Formale Sprachen und Automaten

Inhaltliche Schwerpunkte:

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

Zeitbedarf: 30 Stunden
</td>
</tr>
</table>

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

Unterrichtsvorhaben Q1-III

Thema:
Endliche Automaten und formale Sprachen

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

Inhaltsfelder:
- Endliche Automaten und formale Sprachen

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

Zeitbedarf: 10 Stunden

</td>
<td>

Unterrichtsvorhaben Q1-IV

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

Zentrale Kompetenzen:
- Argumentieren
- Modellieren
- Implementieren
- Darstellen und Interpretieren
- Kommunizieren und Kooperieren

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

Inhaltliche Schwerpunkte:
- Objekte und Klassen
- Analyse, Entwurf und Implementierung von Algorithmen
- Algorithmen in ausgewählten informatischen Kontexten
- Syntax und Semantik einer Programmiersprache

Zeitbedarf: 22 Stunden

</td>
</tr>
<tr><td colspan="2">Summe Qualifikationsphase 1: 74 Stunden</td></tr>
</table>

Qualifikationsphase 2	
Unterrichtsvorhaben Q2-I **Thema:** *Sicherheit und Datenschutz in Netzstrukturen* **Zentrale Kompetenzen:** • Argumentieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Informatiksysteme • Informatik, Mensch und Gesellschaft **Inhaltliche Schwerpunkte:** • Einzelrechner und Rechnernetzwerke • Sicherheit • Nutzung von Informatiksystemen, Wirkungen der Automatisierung **Zeitbedarf**: 22 Stunden	*Unterrichtsvorhaben Q2-II* **Thema:** *Modellierung und Nutzung von relationalen Datenbanken in Anwendungskontexten* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen und Automaten • Informatik, Mensch und Gesellschaft **Inhaltliche Schwerpunkte:** • Datenbanken • Algorithmen in ausgewählten informatischen Kontexten • Syntax und Semantik einer Programmiersprache • Sicherheit **Zeitbedarf**: 16 Stunden

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

<u>Unterrichtsvorhaben Q2-III</u>

Thema:
Prinzipielle Arbeitsweise eines Computers und Grenzen der Automatisierbarkeit

Zentrale Kompetenzen:
- Argumentieren
- Kommunizieren und Kooperieren

Inhaltsfelder:
- Informatiksysteme
- Informatik, Mensch und Gesellschaft

Inhaltliche Schwerpunkte:
- Einzelrechner und Rechnernetzwerke
- Grenzen der Automatisierung

Zeitbedarf: 8 Stunden

</td>
<td>

<u>Unterrichtsvorhaben Q2-IV</u>

Thema:
Wiederholung der objektorientierten Modellierung und Programmierung

Zentrale Kompetenzen:
- Argumentieren
- Modellieren
- Implementieren
- Darstellen und Interpretieren
- Kommunizieren und Kooperieren

Inhaltsfelder:
- Daten und ihre Strukturierung
- Algorithmen
- Formale Sprachen und Automaten
- Informatiksysteme

Inhaltliche Schwerpunkte:
- Objekte und Klassen
- Analyse, Entwurf und Implementierung von Algorithmen
- Syntax und Semantik einer Programmiersprache
- Nutzung von Informatiksystemen

Zeitbedarf: 8 Stunden

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

2.1.2 Konkretisierte Unterrichtsvorhaben

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

Hinweis:

Verbindliche Festlegungen der Fachkonferenz:

Kompetenzorientierte Kernlehrpläne sind curriculare Vorgaben, bei denen die erwarteten Lernergebnisse in Form von fachbezogenen Kompetenzen beschrieben und fachdidaktisch begründeten Kompetenzbereichen sowie Inhaltsfeldern zugeordnet sind. Diese zu entwickelnden Kompetenzen sind in den folgenden konkretisierten Unterrichtsvorhaben in der jeweils zweiten Tabellenspalte aufgeführt.

Die Mitglieder der Fachkonferenz haben sich darauf verständigt, in ihrem Unterricht geeignete Lerngelegenheiten anzubieten, so dass die Schülerinnen und Schüler diese Kompetenzen im Rahmen der angegebenen Unterrichtsvorhaben erwerben oder vertiefen können.

Unterrichtliche Anregungen:

Die Themen, Leitfragen und Ausführungen unter der Überschrift *Vorhabenbezogene Konkretisierung*, sowie die angeführten Beispiele, Medien und Materialien sind Vorschläge bzw. Hilfen für die Lehrkräfte des Gymnasiums Würselen und zeigen auf, wie die curricularen Vorgaben umgesetzt und sequenziert werden können.

In diesen Bereichen sind Abweichungen von den vorgeschlagenen Vorgehensweisen möglich.

I) Einführungsphase

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

Die Schülerinnen und Schüler

- verwenden Fachausdrücke bei der Kommunikation über informatische Sachverhalte (K),
- präsentieren Arbeitsabläufe und -ergebnisse (K),
- kommunizieren und kooperieren in Gruppen und in Partnerarbeit (K),
- nutzen das verfügbare Informatiksystem zur strukturierten Verwaltung und gemeinsamen Verwendung von Daten unter Berücksichtigung der Rechteverwaltung (K).

In der Einführungsphase wird eine Entwicklungsumgebung wie z.B. *Greenfoot* verwendet. Das Installationspaket für *Greenfoot* findet sich unter www.greenfoot-center.de

Unterrichtsvorhaben EF-I

Thema: Einführung in die Nutzung von Informatiksystemen und in grundlegende Begrifflichkeiten

Leitfragen: *Womit beschäftigt sich die Wissenschaft der Informatik? Wie kann die in der Schule vorhandene informatische Ausstattung genutzt werden?*

Vorhabenbezogene Konkretisierung:

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

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

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

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

Zeitbedarf: 6 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. Information, deren Kodierung und Speicherung (a) Informatik als Wissenschaft der Verarbeitung von Informationen (b) Darstellung von Informationen in Schrift, Bild und Ton (c) Speichern von Daten mit	Die Schülerinnen und Schüler • beschreiben und erläutern den Aufbau und die Arbeitsweise singulärer Rechner am Beispiel der „Von-Neumann-Architektur“ (A), • nutzen die im Unterricht eingesetzten Informatiksysteme selbstständig,	*Beispiel*: Textkodierung Kodierung und Dekodierung von Texten mit unbekannten Zeichensätzen (z.B. Wingdings) *Beispiel*: Bildkodierung Kodierung von Bildinformationen in Raster- und Vektorgrafiken

informatischen Systemen am Beispiel der Schulrechner (d) Vereinbarung von Richtlinien zur Datenspeicherung auf den Schulrechnern (z.B. Ordnerstruktur, Dateibezeichner usw.)	sicher, zielführend und verantwortungsbewusst (D), • nutzen das Internet zur Recherche, zum Datenaustausch und zur Kommunikation (K).	
2. Informations- und Datenübermittlung in Netzen (a) „Sender-Empfänger-Modell“ und seine Bedeutung für die Eindeutigkeit von Kommunikation (b) Informatische Kommunikation in Rechnernetzen am Beispiel des Schulnetzwerks (z.B. Benutzeranmeldung, Netzwerkordner, Zugriffsrechte, Client-Server) (c) Grundlagen der technischen Umsetzung von Rechnerkommunikation am Beispiel des Internets (z.B. Netzwerkadresse,		*Beispiel*: Rollenspiel zur Paketvermittlung im Internet Schülerinnen und Schüler übernehmen die Rollen von Clients und Routern. Sie schicken spielerisch Informationen auf Karten von einem Schüler-Client zum anderen. Jede Schülerin und jeder Schüler hat eine Adresse, jeder Router darüber hinaus eine Routingtabelle. Mit Hilfe der Tabelle und einem Würfel wird entschieden, wie ein Paket weiter vermittelt wird.

Paketvermittlung, Protokoll) (d) Richtlinien zum verantwortungsvollen Umgang mit dem Internet		
3. Aufbau informatischer Systeme (a) Identifikation typischer Komponenten informatischer Systeme und anschließende Beschränkung auf das Wesentliche, Herleitung der „Von-Neumann-Architektur“ (b) Identifikation des EVA-Prinzips (Eingabe-Verarbeitung-Ausgabe) als Prinzip der Verarbeitung von Daten und Grundlage der „Von-Neumann-Architektur“		*Material*: Demonstrationshardware Durch Demontage eines Demonstrations-rechners entdecken Schülerinnen und Schüler die verschiedenen Hardwarekomponenten eines Informatiksystems. Als Demonstrationsrechner bietet sich ein ausrangierter Schulrechner an.

Unterrichtsvorhaben EF-II

Thema: Grundlagen der objektorientierten Analyse, Modellierung und Implementierung anhand einer Entwicklungsumgebung wie z.B. *Greenfoot*.

Leitfrage: *Wie lassen sich Gegenstandsbereiche informatisch modellieren und im Sinne einer Simulation informatisch realisieren?*

Vorhabenbezogene Konkretisierung:

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

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

Im Anschluss wird mit der Realisierung erster Projekte mit Hilfe einer didaktischen Programmierumgebung wie z.B. *Greenfoot* begonnen. Die von der Bibliothek vorgegebenen Klassen werden von Schülerinnen und Schülern in Teilen analysiert und entsprechende Objekte anhand einfacher Problemstellungen erprobt. Dazu muss der grundlegende Aufbau einer Java-Klasse thematisiert und zwischen Deklaration, Initialisierung und Methodenaufrufen unterschieden werden.

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

Zeitbedarf: 4 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. Identifikation von Objekten	Die Schülerinnen und Schüler	*Beispiel:* Vogelschwarm

(a) Am Beispiel eines lebensweltnahen Beispiels werden Objekte im Sinne der Objektorientierten Modellierung eingeführt. (b) Objekte werden mit Objektkarten visualisiert und mit sinnvollen Attributen und „Fähigkeiten“, d.h. Methoden versehen. (c) Manche Objekte sind prinzipiell typgleich und werden so zu einer Objektsorte bzw. Objektklasse zusammengefasst. (d) Vertiefung: Modellierung weiterer Beispiele ähnlichen Musters	• ermitteln bei der Analyse einfacher Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • modellieren Klassen mit ihren Attributen, ihren Methoden und Assoziationsbeziehungen (M), • stellen die Kommunikation zwischen Objekten grafisch dar (M), • implementieren einfache Algorithmen unter Beachtung der Syntax und Semantik einer Programmiersprache (I), • stellen den Zustand eines Objekts dar (D).	Schülerinnen und Schüler betrachten einen Vogelschwarm als Menge gleichartiger Objekte, die in einer Klasse mit Attributen und Methoden zusammengefasst werden können. *Materialien:* Ergänzungsmaterialien zum Lehrplannavigator - Allgemeine Objektorientierung (Download EF-II.1)
2. Analyse von Klassen didaktischer Lernumgebungen (a) Objektorientierte Programmierung als modularisiertes Vorgehen (Entwicklung von Problemlösungen auf Grundlage vorhandener Klassen)		*Materialien*: *Greenfoot* (oder andere)

(b) Teilanalyse der Klassen einer didaktischen Lernumgebung wie z.B. *Greenfoot*		
3. Implementierung dreidimensionaler, statischer Szenen (a) Grundaufbau einer *Java*-Klasse (b) Konzeption einer Szene mit Kamera, Licht und sichtbaren Objekten (c) Deklaration und Initialisierung von Objekten (d) Methodenaufrufe mit Parameterübergabe zur Manipulation von Objekteigenschaften (Position, Drehung)		*Beispiel:* Igel-Szenario Schülerinnen und Schüler benutzen ein Szenario, erzeugen Objekte durch Verwendung der *Greenfoot*-Entwicklungs-umgebung und rufen die vorhandenen Methoden auf. *Materialien:* Igel-Szenario, Download auf der Lernplattform Fronter

Unterrichtsvorhaben EF-III

Thema: Grundlagen der objektorientierten Programmierung und algorithmischer Grundstrukturen in *Java* anhand von einfachen Animationen

Leitfragen: *Wie werden die Fähigkeiten von Objekten durch Implementierung geeigneter Methoden erweitert?*

Vorhabenbezogene Konkretisierung:
Der Schwerpunkt dieses Unterrichtsvorhabens liegt auf der Entwicklung eines Verständnisses für die Grundlagen der objektorientierten Programmierung (Klasse, Objekt, Attribute, Methoden) und des Kennenlernens algorithmischer Grundstrukturen (Verzweigungen, Schleifen).

Für die Implementierung wird eine grafische Entwicklungsumgebung (z.B. Greenfoot) verwendet.

Sind anhand einfacher Beispiele wie etwa der Entwicklung eines einfachen Computerspiels im Schwerpunkt Schleifen und Verzweigungen eingeführt worden, sollen diese Konzepte an weiteren Beispielprojekten eingeübt werden. Auch die Erzeugung größerer Mengen grafischer Objekte und deren Verwaltung in einem Feld kann ein Anlass zur Verwendung von Kontrollstrukturen sein.

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

Die Reihe kann durch Videotutorials in unserem Greenfoot-Channel auf Youtube gestützt werden.
Methodisch kann dabei das „Flipped Classroom“-Konzept zum Einsatz kommen.

Zeitbedarf: 20 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Bewegungsanimationen am Beispiel einfacher grafischer Objekte (a) Steuerung von Bildschirmobjekten durch Aufruf geeigneter Methoden (b) if-Anweisung und boolesche Methoden zur Überprüfung von (Spiel-)Situationen (c) Tastaturabfrage und Steuerung von Bildschirmobjekten (d) Modellierung und Implementierung eigener Methoden (e) Erstellung eigener Klassen (f) Interaktion zwischen Objekten (g) evtl. Verwalten und Anzeigen eines Punktestands	Die Schülerinnen und Schüler • analysieren und erläutern einfache Algorithmen und Programme (A), • entwerfen einfache Algorithmen und stellen sie umgangssprachlich und grafisch dar (M), • ermitteln bei der Analyse einfacher Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • modellieren Klassen mit ihren Attributen, ihren Methoden und Assoziationsbeziehungen (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare Datensammlungen zu (M), • ordnen Klassen, Attributen und	*Beispiel*: Trick-the-Turtle Die Schülerinnen und Schüler realisieren auf der Basis des Trick-the-Turtle-Szenarios ein Spiel, bei dem eine Turtle durch den Benutzer gesteuert Salat fressen und sich bewegenden Schlangen ausweichen soll. *Materialien:* Trick-the-Turtle-Szenarion : Download auf der Lernplattform Fronter Lernvideos des Greenfoot-Channels

	Methoden ihren Sichtbarkeitsbereich zu (M), • modifizieren einfache Algorithmen und Programme (I), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I), • implementieren Algorithmen unter Verwendung von Variablen und Wertzuweisungen, Kontrollstrukturen sowie Methodenaufrufen (I), • implementieren einfache Algorithmen unter Beachtung der Syntax und Semantik einer Programmiersprache (I), • testen Programme schrittweise anhand von Beispielen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I).	

Unterrichtsvorhaben EF-IV

Thema: Modellierung und Implementierung von Algorithmen an zweidimensionalen Arrays.

Leitfrage: *Wie können zweidimensionale Arrays durch geschachtelte Schleifen im Rahmen eines größeren Softwareprojekts manipuliert werden?*

Vorhabenbezogene Konkretisierung:
Dieses Unterrichtsvorhaben beschäftigt sich im Schwerpunkt mit den algorithmischen Besonderheiten im Umgang mit zweidimensionalen Arrays.

Als mögliche Anwendung wird Conways Game of Life oder ein Programm zur Bildverarbeitung modelliert und implementiert.
Am Beispiel des Bildverarbeitungsprogramms:
Auf der Basis der Analyse von bekannten Bildverarbeitungsprogrammen werden zunächst typische Funktionalitäten einer solchen Software heraus gearbeitet.
Die Schülerinnen und Schüler erhalten dann ein BlueJ-Projekt, in dem die GUI des Bildverarbeitungsprogramms bereits vorbereitet ist. Die Menueinträge lassen sich je nach gewünschten Funktionalitäten dabei verändern. Ebenfalls vorbereitet ist der Menupunkt „Datei öffnen“, bei dem ein Datei-Öffnen-Dialog erscheint und eine ausgewählte Grafikdatei (.bmp, .png, .jpg“ in ein zweidimensionales Array importiert wird.
In einem Stationlernen sollen sich die Schülerinnen und Schüler zunächst mit dem Aufbau des Projekts anhand von UML-Diagrammen beschäftigen. Dabei werden dann grundlegende Strukturen geklärt wie z.B. der Aufbau der Klasse Pixel und die Bedeutung von Farbwerten im RGB-Farbmodell.
In weiteren vorbereitenden Stationen wird die Datenstruktur zweidim. Array eingeführt und eingeübt.
In der Hauptphase des Projekts werden wiederum anhand verschiedener Stationen die notwendigen Algorithmen zur Bildverarbeitung (z.B. Umwandlung in Graustufen, Schärfen, Weichzeichnen, Drehen,...) erarbeitet und sollen von den Schülern zunächst in Struktogrammen modelliert und anschließend implementiert werden. Dazu werden die vorhandenen Methoden der zur Verfügung gestellten Klassen benutzt.

Zeitbedarf: 18 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
In dieser Sequenz wird die Modellierung eines Problems unter Verwendung der Datenstruktur zweidimensionales Array behandelt. Zur Auswahl stehen beispielsweise Conways Game of Life oder ein Projekt zur Bildverarbeitung, das im folgenden beispielhaft aufgeführt ist, aber auch durch Game of Life ersetzt werden kann: **1. Entwicklung eines Bildverarbeitungsprogramms als Stationenlernen zur Manipulation zweidimensionaler Arrays** (a) Digitale Bilder als zweidimensionales Arrays vom Typ Pixel, RGB-Farbsystem, Graustufen (b) Kennenlernen der Datenstruktur zweidimensionales Array (c) Umsetzung von Algorithmen zur	Die Schülerinnen und Schüler • analysieren und erläutern eine objektorientierte Modellierung (A), • stellen die Kommunikation zwischen Objekten grafisch dar (M), • ermitteln bei der Analyse einfacher Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • modellieren Klassen mit ihren Attributen, ihren Methoden und Assoziationsbeziehungen (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare Datensammlungen zu (M), • ordnen Klassen, Attributen und Methoden ihren Sichtbarkeitsbereich	*Beispiel*: BlueJ (Download *bluej.org)* Basisversion des BlueJ-Projekts „Bildverarbeitung“, Download auf der Lernplattform Fronter Beschreibung der Stationen, Stationenlaufzettel als Download auf Fronter

Bildverarbeitung (Umwandlung in Graustufen, Schärfen, Weichzeichnen, …) in Java-Code. (d) Verschachtelte for-Schleifen	zu (M), • modellieren Klassen unter Verwendung von Vererbung (M), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I), • testen Programme schrittweise anhand von Beispielen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • modifizieren einfache Algorithmen und Programme (I), • stellen Klassen, Assoziations- und Vererbungsbeziehungen in Diagrammen grafisch dar (D), • dokumentieren Klassen durch Beschreibung der Funktionalität der Methoden (D).	
2. Verwalten größerer Mengen einfacher grafischer Objekte (Pixel) (a) Verwaltung von Objekten in zweidimensionalen Feldern (Arrays) (b) Operationen auf zweidimensionalen Arrays (z.B. Nachbarn betrachten)		
3. Modellierung und Implementation komplexerer Bildverarbeitungsalgorithmen (a) Modellierung von Algorithmen zur Bildverarbeitung mit Hilfe von Struktogrammen.		

(b) Implementierung eigener Methoden mit und ohne Parameterübergabe (c) Realisierung von Zustandsvariablen (d) Thematisierung des Geheimnisprinzips und des Autonomitätsprinzips von Objekten (e) Bildverarbeitung mit Hilfe des Aufrufs von selbstimplementierten Methoden		

Unterrichtsvorhaben EF-V

Thema: Such- und Sortieralgorithmen anhand kontextbezogener Beispiele

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

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

Zunächst erarbeiten die Schülerinnen und Schüler mögliche Einsatzszenarien für Such- und Sortieralgorithmen, um sich der Bedeutung einer effizienten Lösung dieser Probleme bewusst zu werden. Anschließend werden Strategien zur Sortierung mit Hilfe eines explorativen Spiels von den Schülerinnen und Schülern selbst erarbeitet und hinsichtlich der Anzahl notwendiger Vergleiche auf ihre Effizienz untersucht.

Daran anschließend werden die erarbeiteten Strategien systematisiert und im Pseudocode notiert. Die Schülerinnen und Schüler sollen auf diese Weise das *Sortieren durch Vertauschen*, das *Sortieren durch Auswählen* und mindestens einen weiteren Sortieralgorithmus, kennen lernen.

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

Zeitbedarf: 9 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. Lineare Suche auf unsortierten und Binäre Suche auf sortierten Daten (a) Suchaufgaben im Alltag und im Kontext informatischer Systeme (b) Evtl. Simulationsspiel zum effizienten Suchen mit binärer Suche (c) Effizienzbetrachtungen zur binären	Die Schülerinnen und Schüler • entwerfen Suchverfahren auf unsortierten Daten (A,M), • analysieren den Aufwand für das Suchen auf unsortierten Daten (A,M), • entwerfen Suchverfahren auf sortierenden Daten und analysieren	*Beispiel:* Simulationsspiel zur binären Suche nach Tischtennisbällen Mehrere Tischtennisbälle sind nummeriert, sortiert und unter Bechern verdeckt. Mit Hilfe der binären Suche kann sehr schnell ein bestimmter Tischtennisball gefunden werden. *Materialien:* Computer science unplugged – Searching

Suche	den Aufwand (A,M)	Algorithms, URL: www.csunplugged.org/searching-algorithms, abgerufen: 30. 03. 2014
2. Explorative Erarbeitung eines Sortierverfahrens (a) Sortierprobleme im Kontext informatischer Systeme und im Alltag (z.B. Dateisortierung, Tabellenkalkulation, Telefonbuch, Bundesligatabelle, usw.) (b) Vergleich zweier Elemente als Grundlage eines Sortieralgorithmus (c) Erarbeitung eines Sortieralgorithmus durch die Schülerinnen und Schüler	Die Schülerinnen und Schüler • beurteilen die Effizienz von Algorithmen am Beispiel von Sortierverfahren hinsichtlich Zeit und Speicherplatzbedarf (A), • entwerfen einen weiteren Algorithmus zum Sortieren (M), • analysieren Such- und Sortieralgorithmen und wenden sie auf Beispiele an (D).	*Beispiel:* Sortieren mit Waage Die Schülerinnen und Schüler bekommen die Aufgabe, kleine, optisch identische Kunststoffbehälter aufsteigend nach ihrem Gewicht zu sortieren. Dazu steht ihnen eine Balkenwaage zur Verfügung, mit deren Hilfe sie das Gewicht zweier Behälter vergleichen können. *Materialien:* Computer science unplugged – Sorting Algorithms, URL: www.csunplugged.org/sorting-algorithms abgerufen: 30. 03. 2014
3. Systematisierung von Algorithmen und Effizienzbetrachtungen (a) Formulierung (falls selbst		*Beispiele*: Sortieren durch Auswählen, Sortieren durch Vertauschen, Quicksort

gefunden) oder Erläuterung von mehreren Algorithmen im Pseudocode (auf jeden Fall: Sortieren durch Vertauschen, Sortieren durch Auswählen) (b) Anwendung von Sortieralgorithmen auf verschiedene Beispiele (c) Bewertung von Algorithmen anhand der Anzahl der nötigen Vergleiche (d) Variante des Sortierens durch Auswählen (Nutzung eines einzigen oder zweier Felder bzw. lediglich eines einzigen zusätzlichen Ablageplatzes oder mehrerer neuer Ablageplätze) (e) Effizienzbetrachtungen an einem konkreten Beispiel bezüglich der Rechenzeit und des Speicherplatzbedarfs (f) Analyse des weiteren Sortieralgorithmus (sofern nicht in Sequenz 1 und 2 bereits		Quicksort ist als Beispiel für einen Algorithmus nach dem Prinzip *Teile und Herrsche* gut zu behandeln. Kenntnisse in rekursiver Programmierung sind nicht erforderlich, da eine Implementierung nicht angestrebt wird. *Materialien:* Computer science unplugged – Sorting Algorithms, URL: www.csunplugged.org/sorting-algorithms abgerufen: 30. 03. 2014

geschehen)		

Unterrichtsvorhaben EF-VI

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?*

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 und in Form von Plakatpräsentationen vorgestellt. Schülerinnen und Schüler sollen dabei mit Unterstützung des Lehrenden selbstständige Recherchen zu ihren Themen anstellen und auch eine sinnvolle Eingrenzung ihres Themas vornehmen.

Anschließend wird verstärkt auf den Aspekt des Datenschutzes eingegangen. Dazu wird das Bundesdatenschutzgesetz in Auszügen behandelt und auf schülernahe Beispielsituationen zur Anwendung gebracht. Dabei steht keine formale juristische Bewertung der Beispielsituationen im Vordergrund, die im Rahmen eines Informatikunterrichts auch nicht geleistet werden kann, sondern vielmehr eine persönliche Einschätzung von Fällen im Geiste des Datenschutzgesetzes.

Zeitbedarf: 15 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Selbstständige Erarbeitung von Themen durch die Schülerinnen und Schüler (a) Mögliche Themen zur Erarbeitung in Kleingruppen: • „Eine kleine Geschichte der Digitalisierung: vom Morsen zum modernen Digitalcomputer“ • „Eine kleine Geschichte der Kryptographie: von Caesar zur Enigma“ • „Von Nullen, Einsen und mehr: Stellenwertsysteme und wie man mit ihnen rechnet“ • „Kodieren von Texten und Bildern: ASCII, RGB und mehr“ • „Auswirkungen der Digitalisierung: Veränderungen der Arbeitswelt und Datenschutz“ (b) Vorstellung und Diskussion durch	Die Schülerinnen und Schüler • bewerten anhand von Fallbeispielen die Auswirkungen des Einsatzes von Informatiksystemen (A), • erläutern wesentliche Grundlagen der Geschichte der digitalen Datenverarbeitung (A), • stellen ganze Zahlen und Zeichen in Binärcodes dar (D), • interpretieren Binärcodes als Zahlen und Zeichen (D), • nutzen das Internet zur Recherche, zum Datenaustausch und zur Kommunikation. (K).	*Beispiel:* Ausstellung zu informatischen Themen Die Schülerinnen und Schüler bereiten eine Ausstellung zu informatischen Themen vor. Dazu werden Stellwände und Plakate vorbereitet, die ggf. auch außerhalb des Informatikunterrichts in der Schule ausgestellt werden können. *Materialien:* Schülerinnen und Schüler recherchieren selbstständig im Internet, in der Schulbibliothek, in öffentlichen Bibliotheken, usw.

Schülerinnen und Schüler		
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“		*Beispiel:* Fallbeispiele aus dem aktuellen Tagesgeschehen Die Schülerinnen und Schüler bearbeiten Fallbeispiele aus ihrer eigenen Erfahrungswelt oder der aktuellen Medienberichterstattung. *Materialien:* Materialblatt zum Bundesdatenschutzgesetz (Download EF-VI.1)

II) 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).

Vorgaben, Hinweise und Beispiele des Ministeriums sind unter folgender URL zu finden.

http://www.schulentwicklung.nrw.de/lehrplaene/lehrplannavigator-s-ii/gymnasiale-oberstufe/informatik/hinweise-und-beispiele/hinweise-und-beispiele.html

Werkzeuge/Software:

Die im Unterricht verwendete Software ist freie oder Open Source Software, d.h. in jedem Fall ohne Kosten erhältlich und aktualisierbar.
Die ausgewählten Produkte stehen sämtlich für GNU/Linux Distributionen, Microsoft Windows und Apple Mac OS zur Verfügung.

Der benötigte ***Java*-Compiler** ist in dem *Java Development Kit* (*JDK*) enthalten. Die Dokumentation der *Java*-Klassen muss separat herunter geladen werden.
http://www.oracle.com/technetwork/java/javase/downloads/index.html

Geany ist ein Texteditor und eine kleine und schnelle integrierte Entwicklungsumgebung (IDE) und in ihrem Funktionsumfang sehr gut für den Unterricht geeignet. *Geany* stellt u.a. eine Suche mit regulären Ausdrücken und Blockeditierung zur Verfügung.
http://www.geany.org
https://de.wikipedia.org/wiki/Geany

dc (*desk calculator*) ist ein Rechner für die Kommandozeile, der das Prinzip der umgekehrten polnischen Notation nutzt.
bc (*basic calculator*) ist eine Erweiterung des dc, verwendet jedoch die gewohnte Infix-Notation.
dc und *bc* sind Multipräzisionsrechner.
https://www.gnu.org/software/bc/bc.html
https://en.wikipedia.org/wiki/Dc_(computer_program)

Ncat (*Netcat*) ist ein Kommandozeilenwerkzeug, das Daten in Netzwerken lesen und schreiben kann.
http://nmap.org/ncat
https://de.wikipedia.org/wiki/Netcat

OpenSSL enthält Implementierungen der Netzwerkprotokolle TLS und SSL und bietet auf der Kommandozeile verschiedene Möglichkeiten der Verschlüsselung von Daten und die Erstellung von Zertifikaten an.
https://www.openssl.org
https://de.wikipedia.org/wiki/OpenSSL

GnuPG ist eine freie Implementierung des OpenPGP-Standards und ermöglicht z.B. über das *Thunderbird*-Plugin *Enigmail* als Benutzer-Schnittstelle eine sichere *e*-Mail-Kommunikation.
GnuPG gehört zu den wenigen Kryptographie-Werkzeugen, die (im Juni 2015) als sicher gelten.
https://www.gnupg.org
http://www.gpg4win.de
https://gpgtools.org
http://de.wikipedia.org/wiki/GNU_Privacy_Guard

wxMaxima ist eine IDE für das freie Computer Algebra System *Maxima*, das zur Lehre und im Unterricht an vielen Schulen und Universitäten eingesetzt wird. Insbesondere im Bereich der Kryptographie/Zahlentheorie werden von den Entwicklern didaktisch motivierte Funktionen bereit gestellt.
https://andrejv.github.io/wxmaxima
http://maxima.sourceforge.net

MySQL ist ein relationales Datenbankverwaltungssystem.
https://www.mysql.com
https://de.wikipedia.org/wiki/MySQL

Das freie Office-Paket ***LibreOffice*** enthält u.a. das Datenbankmanagementsystem Base.
http://de.libreoffice.org
https://de.wikipedia.org/wiki/LibreOffice

world-factbook ist eine vom CIA heraus gegebene Datenbank.
https://www.cia.gov/library/publications/the-world-factbook/
https://de.wikipedia.org/wiki/The_World_Factbook

MMIX ist ein Modellprozessor, der von Donald E. Knuth in seinem Standardwerk "*The Art of Computer Programming*" entwickelt wurde.
http://mmix.cs.hm.edu
https://de.wikipedia.org/wiki/MMIX

PARI/GP ist eine freie Mathematiksoftware, spezialisiert auf den Bereich der Zahlentheorie. *PARI/GP* verfügt insbesondere über eine sehr schnelle Implementation des Zahlkörpersiebs zur Faktorisierung großer natürlicher Zahlen.
http://pari.math.u-bordeaux.fr

Netbeans ist eine Entwicklungsumgebung für Software.
https://netbeans.org

Filius ist eine Software zur Netzwerksimulation und wurde entwickelt, um Unterricht zum Internet zu unterstützen.
http://www.lernsoftware-filius.de

Unterrichtsvorhaben Q1-I:

Thema: Suchen und Sortieren in Arrays

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

Vorhabenbezogene Konkretisierung:
„Rules of the game. Our primary concern is algorithms for rearranging arrays of items where each item contains a *key*."
So beschreibt Robert Sedgewick in seiner Standardfachliteratur „*Algorithms (Fourth Edition)"* die Spielregeln, d.h. Rahmenbedingungen für eine Sortierung. Sie wird in der Regel in einem Array durchgeführt. In der *Java Collections API* werden z.B. die Daten einer Liste zur Sortierung in ein Array kopiert, dort neu geordnet und am Ende wieder in die Liste zurück geschrieben. Dieser vermeintliche Umweg unterstreicht den Stellenwert der Arrays in der Sortierung.

Die hier bereits angesprochene Vergleichbarkeit von Objekten mit Hilfe von vergleichbaren Schlüsseln ist für die Sortierung eine wesentliche Voraussetzung und wird später bei den zweidimensionalen Datenstrukturen wieder aufgegriffen und wird auch dort zur grundlegenden Leitidee.

Zunächst werden in einem Anwendungskontext Informationen in einem Array gesucht. Hierzu werden Verfahren entwickelt und implementiert bzw. analysiert und erläutert, wobei die binäre Suche sowohl iterativ auch rekursiv implementiert wird. Die verschiedenen Verfahren werden hinsichtlich der Zahl der Vergleichsoperationen miteinander verglichen.

Anschließend werden Sortierverfahren entwickelt und implementiert. Hierbei soll mit Quicksort auch ein rekursives Sortierverfahren entwickelt werden. Die Implementationen von Quicksort sowie dem Sortieren durch Einfügen werden analysiert und erläutert. Zur Veranschaulichung der Arbeitsweisen der Algorithmen werden die jeweiligen Methodenaufrufe graphisch dargestellt.

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

Zeitbedarf: 12 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Suchen von Daten in Arrays (a) Lineare Suche (b) Binäre Suche, iterativ und rekursiv (c) Untersuchung der beiden Suchverfahren hinsichtlich ihrer Effizienz (Laufzeitverhalten)	Die Schülerinnen und Schüler • analysieren und erläutern Algorithmen und Programme (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • beurteilen die Effizienz von Algorithmen unter Berücksichtigung des Speicherbedarfs und der Zahl der Operationen (A), • entwickeln iterative und rekursive Algorithmen unter Nutzung der Strategien „Modularisierung“ und „Teilen und Herrschen“ (M), • verwenden bei der Modellierung geeigneter Problemstellungen dieMöglichkeiten der Polymorphie (M), • modifizieren Algorithmen und Programme (I), • implementieren und erläutern iterative und rekursive Such- und Sortierverfahren (I), • nutzen die Syntax und Semantik einer	*Beispiel:* Wörterbuch Ein Wörterbuch mit einer Zuordnung von z.B. deutschen und englischen mathematischen Fachbegriffen kann in einem ersten Ansatz auf einfache Weise in einem Array abgelegt werden. Ist das Array sortiert, kann man eine binäre Suche zur effizienten Suche nach einem bestimmten Wort verwenden. Das Wörterbuch ist ein ideales Beispiel für ein erstes Kennenlernen einer Schlüssel-Wert-Zuordnung, die wir später bei Binärbäumen und allgemein bei Datenbanken wieder finden werden.
2. Sortieren in Arrays - Entwicklung und Implementierung von iterativen und rekursiven Sortierverfahren (a) Entwicklung und Implementierung eines einfachen iterativen Sortierverfahrens (b) Entwicklung und Implementierung eines rekursiven Sortierverfahrens		*Beispiel:* Wörterbuch Will man zur Reduzierung des Speicherplatzes obige Zuordnungstabelle deutscher und englischer mathematischen Fachbegriffe nur ein einziges Mal ablegen, muss zur Verwendung der binären Suche das Array bezüglich einer bestimmten Sprache sortiert werden. Ein Wechsel der Ausgangssprache erzwingt

	Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • testen Programme systematisch anhand von Beispielen (I), • stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D).	hier einen Rollenwechsel in der Schlüssel-Wert-Zuordnung, was das Verständnis dieser Zuordnung vertiefen kann.
3. Untersuchung der Effizienz der Sortierverfahren „Sortieren durch Einfügen“ und „Quicksort“ (a) Grafische Veranschaulichung (b) Untersuchung der Anzahl der Vergleichsoperationen (c) Beurteilung der Effizienz		*Beispiel:* Laufzeitmessung Die *Java API* bietet eine einfache Möglichkeit, die Laufzeit eines Programms zu messen. Auf diese Weise können die theoretisch ermittelten Laufzeit-Typen experimentell verifiziert werden

Unterrichtsvorhaben Q1-II:

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

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

Vorhabenbezogene Konkretisierung:
Nach Analyse einer Problemstellung in einem geeigneten Anwendungskontext, in dem Daten nach dem First-In-First-Out-Prinzip verwaltet werden, werden der Aufbau von Warteschlangen am Beispiel dargestellt und die Operationen einer implementierenden Klasse erläutert. Anschließend werden die für die Anwendung notwendigen Klassen modelliert und implementiert. Die Anwendung wird anschließend modifiziert, um den Umgang mit der Datenstruktur zu üben.

Um auf einfache Weise Daten löschen zu können, die in einer linearen Struktur zwischen anderen Daten gespeichert sind, werden Listen eingeführt und in einem Anwendungskontext verwendet.

Anhand einer Anwendung, in der Daten nach dem Last-In-First-Out-Prinzip verwaltet werden, werden die Unterschiede zwischen den Datenstrukturen Warteschlange und Stapel erarbeitet.

Die Modellierungen werden dabei in allen Anwendungskontexten in Entwurfs- und Implementationsdiagrammen dargestellt.

Zeitbedarf: 30 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. Die Datenstruktur Warteschlange im Anwendungskontext (a) Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen (b) Erarbeitung der Funktionalität einer Warteschlange (c) Modellierung und Implementierung der Anwendung unter Verwendung einer Warteschlange	Die Schülerinnen und Schüler • erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A), • analysieren und erläutern Algorithmen und Programme (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu (M), • ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M),	*Beispiel: Warteschlangen-Simulation* In einem Supermarkt können sich Kunden im Allgemeinen die Schlange aussuchen, an der sie sich anstellen wollen. In Banken, Kinos oder Bahnhöfen warten Kunden dagegen oft zuerst in einer gemeinsamen Schlange, bevor sie auf die einzelnen Schalter verteilt werden. Diese beiden konkurrierenden Wartesysteme lassen sich in einer Simulation und einer anschließenden statistischen Auswertung mit einander vergleichen. Zur Modellierung des Ablaufs werden die Kunden in einer *ArrayDeque<Kunde>* verwaltet, welche die ADT Warteschlange implementiert. Die Wartezeiten müssen zur Berechnung des Medians sortierbar sein und werden in einer *ArrayList<Integer>* verwaltet, welche dynamisch, linear und sortierbar ist. Die

	• modifizieren Algorithmen und Programme (I), • implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • testen Programme systematisch anhand von Beispielen (I), • stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D).	Sortierung wird an dieser Stelle durch die Methode *sort* der *Java Collections API* bewerkstelligt.
2. Die Datenstruktur lineare Liste im Anwendungskontext (a) Erarbeitung der Vorteile einer Liste im Gegensatz zu den bereits bekannten linearen Strukturen (b) Erarbeitung der Unterschiede verschiedener Implementationsansätze (c) Erarbeitung der Funktionalität eines Iterators für eine Liste (d) Modellierung und Implementierung einer kontextbezogenen Anwendung unter Verwendung einer Liste		*Beispiel:* Zahlenhai - Ein Spiel mit Zahlen Der Zahlenhai ist ein Spiel, mit dem Schüler das Konzept der Teilbarkeit natürlicher Zahlen erlernen und üben können. Anfangs wird eine Liste an Zahlen durch eine obere Schranke festgelegt. Durch geschicktes Auswählen kann der Spieler dann nach einfachen Regeln möglichst viele dieser Zahlen als Punkte gewinnen und damit den Zahlenhai besiegen. Für eine Implementation des Spiels wird die Liste der verwendeten Zahlen in naheliegender Weise durch einen Listen-Typ modelliert. Verschiedene Operationen auf Listen führen zur Untersuchung von unterschiedlicher Listen-Implementationen und zur Behandlung von Iteratoren.
3. Die Datenstruktur Stapel im Anwendungskontext (a) Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen (b) Erarbeitung der Funktionalität eines		*Beispiel*: Auswertung math. Ausdrücke Der *dc* (*desk calculator*) ist ein Rechner für die Kommandozeile, der mathematische Ausdrücke in umgekehrter polnischer Notation (UPN, auch Postfix-Notation)

Stapels (c) Modellierung und Implementierung der Anwendung unter Verwendung eines Stapels		berechnet. Sind die Bestandteile des eingegebenen Terms durch Leerzeichen von einander getrennt, lässt sich der Term recht leicht mit einem *Scanner* zerlegen und in einer Warteschlange speichern. Die lexikalische Analyse allgemeinerer Terme soll erst im Unterrichtsvorhaben Q1-III mit Hilfe regulärer Ausdrücke vorgenommen werden. Zentraler Bestandteil der Verarbeitung und Berechnung der Postfix-Terme ist ein Stapel für die Operanden. *Der bc* (*basic calculator*) ist eine Erweiterung des *dc*, die Terme werden jedoch in der gewohnten Infix-Notation eingegeben. Der Shunting-Yard Algorithmus von Dijkstra ist geeignet, die Infix-Ordnung in eine äquivalente Postfix-Ordnung zu verwandeln. Hierfür wird ein zusätzlicher Stapel für die Operatoren benötigt. Die im Unterricht implementierte Basisversion eignet sich als Grundlage für Facharbeiten in Computermathematik.

Unterrichtsvorhaben Q1-III:

Thema: Endliche Automaten und formale Sprachen

Leitfragen: *Was sind reguläre Ausdrücke? Wie kann man (endliche) Automaten beschreiben und modellieren? Wie können Sprachen durch Grammatiken beschrieben werden? Welche Zusammenhänge gibt es zwischen formalen Sprachen, endlichen Automaten und regulären Grammatiken?*

Vorhabenbezogene Konkretisierung:
Viele moderne Editoren und Textverarbeitungssysteme verfügen über die Möglichkeit, mit Hilfe von regulären Ausdrücken Textbausteine zu suchen und zu ersetzen. In der *Java API* existiert mit der Klasse *Scanner* ein endlicher Automat, der die durch einen regulären Ausdruck beschriebene reguläre Sprache akzeptiert.

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

Anhand weiterer kontextbezogener Beispiele werden verschiedene Darstellungsformen für endliche Automaten ineinander überführt und die akzeptierten Sprachen endlicher Automaten ermittelt. An einem Beispiel werden die Grenzen endlicher Automaten ausgelotet.

Zeitbedarf: 10 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien oder Materialien

1. Untersuchung und Entwicklung von Grammatiken regulärer Sprachen (a) Erarbeitung der formalen Darstellung regulärer Grammatiken (b) Untersuchung, Modifikation und Entwicklung von Grammatiken	Die Schülerinnen und Schüler • analysieren und erläutern die Eigenschaften endlicher Automaten einschließlich ihres Verhaltens auf bestimmte Eingaben (A), • analysieren und erläutern Grammatiken regulärer Sprachen (A), • zeigen die Grenzen endlicher Automaten und regulärer Grammatiken im Anwendungszusammenhang auf (A), • ermitteln die formale Sprache, die durch eine Grammatik erzeugt wird (A), • entwickeln und modifizieren zu einer Problemstellung endliche Automaten (M), • entwickeln und modifizieren zu einer Problemstellung endliche Automaten (M), • entwickeln zur akzeptierten Sprache eines Automaten die zugehörige Grammatik (M), • entwickeln zur Grammatik einer regulären Sprache einen zugehörigen endlichen Automaten (M),	*Beispiel:* Suchen und Ersetzen Editoren erlauben es, mit Hilfe von regulären Ausdrücken Textbausteine zu suchen und zu ersetzen. *Beispiel:* Lexikalische Analyse Die *Java*-Klasse *Scanner* erlaubt es, mit Hilfe von regulären Ausdrücken Muster für die Bestandteile math. Terme zu definieren und Übereinstimmungen zu finden.
2. Endliche Automaten (a) Vom Automaten in bekannten Kontexten zur formalen Beschreibung eines endlichen Automaten (b) Untersuchung, Darstellung und Entwicklung endlicher Automaten (c) Entwicklung von endlichen Automaten zum Erkennen regulärer Sprachen, die durch Grammatiken gegeben werden (d) Entwicklung regulärer Grammatiken zu endlichen		*Beispiel: IBAN*-Prüfsumme Die Bearbeitung einer Überweisung benötigt einen Automaten, der eine gültige *IBAN*-Nr. erkennen kann.

Automaten	• modifizieren Grammatiken regulärer Sprachen (M), • entwickeln zu einer regulären Sprache eine Grammatik, die die Sprache erzeugt (M), • stellen endliche Automaten in Tabellen oder Graphen dar und überführen sie in die jeweils andere Darstellungsform (D), • ermitteln die Sprache, die ein endlicher Automat akzeptiert (D), • beschreiben an Beispielen den Zusammenhang zwischen Automaten und Grammatiken (D).	
3. Grenzen endlicher Automaten		*Beispiel:* Klammern Grundsätzlich kann ein Automat prüfen, ob eine geöffnete Klammer auch wieder geschlossen wird. Im Allgemeinen ist jedoch die Klammertiefe nicht begrenzt.

Unterrichtsvorhaben Q1-IV:

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

Leitfragen: *Wie können Daten im Anwendungskontext mit Hilfe von Schlüssel-Wert-Assoziation und binären Baumstrukturen verwaltet werden?*

Vorhabenbezogene Konkretisierung:
Grundlegend für die Verwaltung großer Datenmengen ist die Assoziation von Daten und Schlüsseln. Ein Wert wird über seinen Schlüssel verwaltet (s. z.B. Robert Sedgewick, „*Algorithms (Fourth Edition)“,* Kap. 3). Lassen sich in einem Anwendungskontext die

Schlüssel durch kleine natürliche Zahlen darstellen, bietet sich ein Array mit den Schlüsseln als Array-Indizes als einfache und nahe liegende Struktur zur Verwaltung der Daten an. Andernfalls werden binäre Suchbäume verwendet.

Zunächst werden anhand von einfachen Beispielen grundlegende Begriffe bei Baumstrukturen eingeführt und die Implementation und rekursive Traversierung (Pre-, Post- und Inorder) von Bäumen erarbeitet.

Klassen und ihre Beziehungen werden dabei in Entwurfs- und Implementationsdiagrammen dargestellt. Die Funktionsweise von Methoden wird anhand grafischer Darstellungen erläutert.

Anschließend werden für Problemstellungen im Anwendungskontext, z.B. im Bereich der künstlichen Intelligenz, die für eine Tiefen- oder Breitensuche erforderlichen Klassen modelliert und implementiert.

In konkreten Anwendungen benötigen die Suchalgorithmen die Hilfe einfacher Datenbanken. Hier steht dann die Schlüssel-Wert-Assoziation und ihre Verwaltung im Mittelpunkt. Mit Hilfe der Schlüssel werden Werte/Daten in einer Datenbank verwaltet und gefunden. Ein Suchbaum stellt dann eine geeignete Datenbank zur Verfügung.

Zeitbedarf: 22 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. Analyse von Baumstrukturen in verschiedenen Kontexten (a) Benennung der grundlegenden Begriffe (Wurzel, Knoten, Tiefe, Blatt, Inhalt, Kind, Eltern) (b) Aufbau und Darstellung von Bäumen anhand von Baumstrukturen in verschiedenen	Die Schülerinnen und Schüler • erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A), • analysieren und erläutern Algorithmen und Programme (A), • beurteilen die syntaktische Korrektheit	*Beispiel:* Termbaum Der Aufbau von math. Ausdrücken wird mit Hilfe von Baumstrukturen dargestellt.

Kontexten	und die Funktionalität von Programmen (A), • ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu (M), • modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M), • entwickeln iterative und rekursive Algorithmen unter Nutzung der Konstruktionsstrategien „Modularisierung“ und „Teilen und Herrschen“ (M), • implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • modifizieren Algorithmen und Programme (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der	
2. Die Datenstruktur Binärbaum im Anwendungskontext (a) Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen im Anwendungskontext (b) Modellierung eines Entwurfsdiagramms und Entwicklung eines Implementationsdiagramms (c) Implementierung der Anwendung oder von Teilen der Anwendung (d) Traversierung eines Binärbaums im Pre-, In- und Postorderdurchlauf		*Beispiel*: Sortierbaum Eine einfache binäre Baumstruktur kann zur Sortierung von Daten genutzt werden. Dadurch, dass für jeden zu sortierenden Wert ein zusätzliches Objekt erzeugt und verwaltet werden muss, ist dieses Verfahren für eine effiziente Sortierung eher nicht geeignet. Man kann jedoch an diesem einfachen Beispiel sehr gut den grundsätzlichen Aufbau des Baums und seine Traversierung studieren.
3. Schlüssel-Wert-Assoziation im Anwendungskontext (a) Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen (b) Erarbeitung der Schlüssel-Wert-Zuordnung im Verbund mit der Vergleichbarkeit der Schlüssel als		*Beispiel*: Entscheidungsprobleme Das Bauer-Kohl-Ziege-Wolf-Problem, RushHour, QuoVadis oder das 15-Puzzle sind Aufgaben, die sich mit Hilfe von Strategien aus dem Bereich der künstlichen Intelligenz lösen lassen. Die sich ergebenden Spielstände werden dabei durch eine Klasse beschrieben und in

grundsätzliches Verwaltungsprinzip (c) Entwicklung und graphische Darstellung einer einfachen Schlüssel-Wert-Zuordnung in einem Array im Anwendungskontext (d) Implementierung der Anwendung oder von Teilen der Anwendung	Implementierung und zur Analyse von Programmen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • testen Programme systematisch anhand von Beispielen (I), • stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D), • stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D).	die Knoten eines Entscheidungsbaums zur Tiefen- oder Breitensuche eingetragen. Damit sich die Spielstände nicht wiederholen und nur optimale Züge weiter verfolgt werden, müssen die Spielstände gleichzeitig mit Hilfe eines geschickt ausgewählten Schlüssels in einer Zuordnungstabelle eingetragen und gefunden werden. Gelingt es, dass die Schlüssel durch kleine natürliche Zahlen ausgedrückt werden können, kann die Zuordnungstabelle durch ein einfaches Array dargestellt werden, …
4. Die Datenstrukturen binärer Suchbaum im Anwendungskontext (a) Analyse der Problemstellung, Ermittlung von Objekten, ihren Eigenschaften und Operationen (b) Grafische Darstellung der Schlüssel-Wert-Zuordnung und der verwendeten Datenstruktur (c) Erarbeitung der Funktionalität eines Binärbaums (d) Implementierung der Anwendung oder von Teilen der Anwendung		*Beispiel*: Entscheidungsprobleme (Fortsetzung) ... andernfalls muss die Zuordnungstabelle eine ein Suchbaum sein.

Unterrichtsvorhaben Q2-I:

Thema: Sicherheit und Datenschutz in Netzstrukturen

Leitfragen: *Wie werden Daten in Netzwerken übermittelt? Wie lassen sich Daten, Information und Kommunikation schützen? Was ist freie Software?*

Vorhabenbezogene Konkretisierung:
Netzwerkstrukturen, gängige Netzwerk-Protokolle und Client-Server-Kommunikation werden vorgestellt, erläutert und mit Hilfe von Netzwerkwerkzeugen wie z.B. *Netcat* oder den Klassen für Socket-Kommunikation der *Java API* in Anwendungsbeispielen erprobt.

Wo Daten, Information und Kommunikation geschützt werden sollen, kommt Kryptographie ins Spiel. Ein modernes Verfahren wie z.B. *RSA* wird vorgestellt und geschützte *e*-Mail-Kommunikation wird praktisch durchgeführt.

Fallbeispiele zur Datenschutzproblematik und zum Urheberrecht runden dieses Thema ab.

Zeitbedarf: 22 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. Kommunikation von Daten in Netzwerken (a) Beschreibung protokollbasierter Kommunikation von Daten im Netz anhand eines Anwendungskontextes (b) Netztopologien als Grundlage von Client-Server-Strukturen	Die Schülerinnen und Schüler • beschreiben und erläutern Topologien, die Client-Server-Struktur und Protokolle sowie ein Schichtenmodell in Netzwerken (A), • analysieren und erläutern Eigenschaften und Einsatzbereiche symmetrischer und asymmetrischer	*Beispiel*: *FTP* Von den bekannten und häufig genutzten Netzwerk-Protokollen ist *FTP* dasjenige, das sich am Einfachsten mit Hilfe eines Netzwerk-Tools wie z.B. *Netcat* nutzen lässt. Viele reale *FTP*-Server eignen sich als bereitwillige Kommunikationspartner. Die Verarbeitung der Datenströme kann hierbei mit Hilfe einer Implementation in *Java* demonstriert werden.

(c) *TCP*/*IP*-Schichtenmodell als Beispiel für eine Paketübermittlung	Verschlüsselungsverfahren (A), • untersuchen und bewerten anhand von Fallbeispielen die Auswirkungen des Einsatzes von Informatiksystemen, die Sicherheit von Informatiksystemen sowie die Einhaltung der Datenschutz-bestimmungen und des Urheberrechts (A), • untersuchen und bewerten Problem-lagen, die sich aus dem Einsatz von Informatiksystemen ergeben, hinsichtlich rechtlicher Vorgaben, ethischer Aspekte und gesellschaftlicher Werte unter Berücksichtigung unterschiedlicher Interessenlagen (A), • wenden eine didaktisch orientierte Entwicklungsumgebung zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I), • nutzen bereitgestellte Informatiksysteme und das Internet reflektiert zum Erschließen, zur Aufbereitung und Präsentation fachlicher Inhalte (D).	
2. Datenschutz (a) Vertraulichkeit, Integrität, Authentizität in Netzwerken (b) Symmetrische und asymmetrische kryptografische Verfahren (c) Sichere *e*-Mail-Kommunikation		*Beispiel*: *e*-Mail-Verschlüsselung mit *GnuPG* Das freie *GnuPG* macht sichere *e*-Mail-Kommunikation möglich. Prinzipiell können direkt auf der Kommandozeile mit *GnuPG* Schlüssel erzeugt und Dateien verschlüsselt werden, einfacher ist jedoch die Bedienung von *GnuPG* über das Plugin *Enigmail* für den *e*-Mail-Client *Thunderbird*. *GnuPG* verwendet *RSA* zur Verschlüsselung und Signierung der Kommunikation und *AES* zur Verschlüsselung des privaten Schlüssels.
3. Datenschutz und Urheberrechte, Fallbeispiele		*Beispiel*: *GNU General Public Licence* (*GPL*) Freie Software wird unter der *GPL* oder einer ähnlichen Lizenz veröffentlicht. Die *GPL* garantiert einer Gemeinschaft von Nutzern vollständige Kontrolle über Software. *GPL*-lizensierte, d.h. freie Software kommt insbesondere dann zum Einsatz, wenn es um Sicherheit und Datenschutz geht.

Unterrichtsvorhaben Q2-II:

Thema: Modellierung und Nutzung von relationalen Datenbanken in Anwendungskontexten

Leitfragen*:* *Wie können Fragestellungen mit Hilfe einer Datenbank beantwortet werden? Wie entwickelt man selbst eine Datenbank für einen Anwendungskontext?*

Vorhabenbezogene Konkretisierung:
Prinzipiell ist ein Wörterbuch bzw. eine Zuordnungstabelle bereits eine Datenbank. Insofern wurden in früheren Unterrichtsvorhaben bereits Kompetenzen im Bereich der Datenbanken erworben, wie z.B. das Strukturieren von Daten mit Hilfe einer Schlüssel-Wert-Assoziation. Im Allgemeinen versteht man jedoch unter dem Begriff Datenbank eine Zuordnungsstruktur, bei der Daten auf eine komplexe und vielschichtige Weise in Beziehung stehen, effizient und widerspruchsfrei gespeichert werden und die flexible Abfragen beliebiger Ausschnitte dieser Struktur ermöglicht. Eine eigene Datenbank zu entwickeln, bedeutet erst einmal Ordnung ins Chaos zu bringen.

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

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

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

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

Zeitbedarf: 16 Stunden

Sequenzierung des Unterrichtsvorhabens

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Nutzung von relationalen Datenbanken (a) Aufbau von Datenbanken und Grundbegriffe • Entwicklung von Fragestellungen zur vorhandenen Datenbank • Analyse der Struktur der vorgegebenen Datenbank und Erarbeitung der Begriffe Tabelle, Attribut, Datensatz, Datentyp, Primärschlüssel, Fremdschlüssel, Datenbankschema (b) SQL-Abfragen • Analyse vorgegebener SQL-Abfragen und Erarbeitung der Sprachelemente von SQL (`SELECT (DISTINCT) …FROM, WHERE, AND, OR, NOT`) auf einer Tabelle • Analyse und Erarbeitung von SQL-Abfragen auf einer und mehrerer Tabelle zur Beantwortung der Fragestellungen (`JOIN, UNION, AS, GROUP BY,ORDER BY, ASC, DESC, COUNT, MAX, MIN, SUM`, Arithmetische Operatoren: `+, -, *, /, (…)`, Vergleichsoperatoren: `=, <>, >, <, >=, <=, LIKE,`	Die Schülerinnen und Schüler • erläutern die Eigenschaften und den Aufbau von Datenbanksystemen unter dem Aspekt der sicheren Nutzung (A), • analysieren und erläutern die Syntax und Semantik einer Datenbankabfrage (A), • analysieren und erläutern eine Datenbankmodellierung (A), • erläutern die Eigenschaften normalisierter Datenbankschemata (A), • bestimmen Primär- und Sekundärschlüssel (M), • ermitteln für anwendungsbezogene Problemstellungen Entitäten, zugehörige Attribute, Relationen und Kardinalitäten (M), • modifizieren eine Datenbankmodellierung (M), • modellieren zu einem Entity-Relationship-Diagramm ein relationales Datenbankschema (M),	*Beispiel: world-factbook* Das *CIA* gibt mit dem *World Factbook* eine wöchentlich aktualisierte Datenbank heraus. Es enthält statistische Daten über 267 Länder der Welt, zumeist aus den Bereichen Demographie, Wirtschaft, Infrastruktur, Kommunikation, Politik und Militär.

BETWEEN, IN, IS NULL) (c) Vertiefung an einem weiteren Datenbankbeispiel	• bestimmen Primär- und Sekundärschlüssel (M), • überführen Datenbankschemata in vorgegebene Normalformen (M), • verwenden die Syntax und Semantik einer Datenbankabfragesprache, um Informationen aus einen Datenbanksystem zu extrahieren (I), • ermitteln Ergebnisse von Datenbankabfragen über mehrere verknüpfte Tabellen (D), • stellen Entitäten mit ihren Attributen und die Beziehungen zwischen Entitäten in einem Entity-Relationship-Diagramm grafisch dar (D), • überprüfen Datenbankschemata auf vorgegebene Normalisierungseigenschaften (D).	
2. Modellierung von relationalen Datenbanken (a) Entity-Relationship-Diagramm • Ermittlung von Entitäten, zugehörigen Attributen, Relationen und Kardinalitäten in Anwendungssituationen und Modellierung eines Datenbankentwurfs in Form eines Entity-Relationship-Diagramms • Erläuterung und Modifizierung einer Datenbankmodellierung (b) Entwicklung einer Datenbank aus einem Datenbankentwurf • Modellierung eines relationalen Datenbankschematas zu einem Entity-Relationship-Diagramm inklusive der Bestimmung von Primär- und Sekundärschlüsseln (c) Redundanz, Konsistenz und Normalformen • Untersuchung einer Datenbank hinsichtlich Konsistenz und Redundanz in einer Anwendungssituation • Überprüfung von Datenbankschemata hinsichtlich		*Beispiel:* Videothek Die Datenverwaltung einer Videothek soll in einem Datenbanksystem umgesetzt werden. Ausgehend von der Modellierung soll mit Hilfe eines ER-Modells und eines Datenbankschemas dieser erste Entwurf normalisiert und in einem Datenbanksystem umgesetzt werden. Es schließen sich diverse SQL-Abfragen an, wobei auf die Relationenalgebra eingegangen wird. *Beispiel:* Buchungssystem In dem Online-Buchungssystem einer Schule können die Lehrer Medienräume, Beamer, Laptops, Kameras, usw. für einen bestimmten Zeitpunkt buchen, der durch Datum und die Schulstunde festgelegt ist. Dazu ist die Datenbank zu modellieren, ggf. zu normalisieren und im Datenbanksystem umzusetzen. Weiter sollen sinnvolle Abfragen entwickelt werden.

der 1. bis 3. Normalform und Normalisierung (um Redundanzen zu vermeiden und Konsistenz zu gewährleisten)		

Unterrichtsvorhaben Q2-III:

Thema: Prinzipielle Arbeitsweise eines Computers und Grenzen der Automatisierbarkeit

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

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

Ausgehend von den prinzipiellen Grenzen endlicher Automaten liegt die Frage nach den Grenzen von Computern bzw. nach Grenzen der Automatisierbarkeit nahe. Es soll plausibel werden, dass es unmöglich ist, ein Informatiksystem zu entwickeln, das für jedes beliebige Computerprogramm und jede beliebige Eingabe entscheidet, ob das Programm mit der Eingabe terminiert oder nicht (Halteproblem). Compiler versuchen z.B. erst gar nicht, die Terminierung einer Schleife zu untersuchen.

Anschließend werden Vor- und Nachteile der Grenzen der Automatisierbarkeit angesprochen und der Einsatz von Informatiksystemen hinsichtlich prinzipieller Möglichkeiten und prinzipieller Grenzen beurteilt. Insbesondere beruht die moderne IT-Sicherheit u.A. auf den grundsätzlich beschränkten Möglichkeiten zur Lösung des RSA-Problems. Es kann z.B. leicht demonstriert werden, dass sich Primfaktorzerlegungen nur in sehr engen Grenzen durchführen lassen.

Zeitbedarf: 8 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien oder Materialien
1. Von-Neumann-Architektur und die Ausführung maschinennaher Programme a) Prinzipieller Aufbau einer von Neumann-Architektur mit CPU, Rechenwerk, Steuerwerk, Register und Hauptspeicher b) Maschinennahe Befehle und ihre Repräsentation in einem Binär-Code, der in einem Register gespeichert werden kann c) Analyse und Erläuterung der Funktionsweise eines einfachen maschinennahen Programms	Die Schülerinnen und Schüler • erläutern die Ausführung eines einfachen maschinennahen Programms sowie die Datenspeicherung auf einer „Von-Neumann-Architektur“ (A), • untersuchen und beurteilen Grenzen des Problemlösens mit Informatiksystemen (A).	*Beispiel:* MMIX Mit Hilfe des Modellprozessors MMIX können einfache, maschinennahe Rechenoperationen und die dahinter liegende Architektur veranschaulicht werden.
2. Grenzen der Automatisierbarkeit a) Vorstellung des Halteproblems b) Unlösbarkeit des Halteproblems c) Beurteilung des Einsatzes von Informatiksystemen hinsichtlich prinzipieller Möglichkeiten und prinzipieller Grenzen		*Beispiel:* Halteproblem *Beispiel:* IT-Sicherheit Laufzeitabschätzungen für die Faktorisierung großer Zahlen und die Berechnung diskreter Logarithmen in PARI/GP.

Unterrichtsvorhaben Q2-IV:

Thema: Wiederholung und Vertiefung ausgewählter Kompetenzen im Bereich der objektorientierten Modellierung und Programmierung und Vorbereitung auf die Zentralprüfung.

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

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

Die Schülerinnen und Schülern erläutern und modifizieren den ersten Entwurf und modellieren sowie implementieren weitere Klassen und Methoden für eine entsprechende Anwendung. Klassen und ihre Beziehungen werden in einem Implementationsdiagramm dargestellt. Dabei werden Sichtbarkeitsbereiche zugeordnet. Exemplarisch wird eine Klasse dokumentiert. In diesem Zusammenhang wird das Nachrichtenkonzept der objektorientierten Programmierung wiederholt.

Zeitbedarf: 8 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien

1. Wiederholung und Erweiterung der objektorientierten Modellierung und Programmierung durch Analyse und Erweiterung eines kontextbezogenen Beispiels (a) Analyse der Problemstellung (b) Analyse der Modellierung (Implementationsdiagramm) (c) Erweiterung der Modellierung im Implementationsdiagramm (Vererbung, abstrakte Klasse) (d) Kommunikation zwischen mindestens zwei Objekten (grafische Darstellung) (e) Dokumentation von Klassen (f) Implementierung der Anwendung oder von Teilen der Anwendung	Die Schülerinnen und Schüler • analysieren und erläutern objektorientierte Modellierungen (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (M), • ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M), • modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I),	*Beispiel:* Künstliche Intelligenz Ausgehend von einer vorliegenden Implementation des Brettspiels *Othello* (*Reversi*) für zwei menschliche Spieler, können dem Programm künstliche Intelligenzen als Gegenspieler hinzu gefügt werden. Die neuen Spieler können über die bereits vorhandene Schnittstelle *Spieler* definiert werden und z.B. die *MinMax*- oder *Alpha-Beta*-Strategie implementieren. Das Werkzeug *javadoc* des *Java Development Kit* (*JDK*) kann hierbei zur Erstellung einer *html*-Klassendokumentation genutzt werden.

	• stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D), • dokumentieren Klassen (D), • stellen die Kommunikation zwischen Objekten grafisch dar (D).	

3 Qualitätssicherung und Evaluation

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

Die beiden (im Juni 2015) in der Oberstufe im Fach Informatik unterrichtenden Kollegen stehen in einem intensiven informellen Austausch über die Inhalte und Methoden des Unterrichts, wodurch ein hohes Maß an fachlicher Qualitätssicherung erreicht wird.

Die zu den Unterrichtsvorhaben

EF-VI (Geschichte der EDV, Datenschutz),

Q1-III (Automaten, formale Sprachen),

Q2-I (Netzstrukturen, Datenschutz),

Q2-II (Datenbanken),

Q2-III (Computer, Grenzen der Automatisierbarkeit)

gehörenden Inhaltsfelder und Kompetenzen wurden als obligatorisch in den Kernlehrplan für den Grundkurs aufgenommen und am Gymnasium Würselen bisher nicht oder nur teilweise unterrichtet bzw. vermittelt.

Deshalb sollen insbesondere die in diesen Unterrichtsvorhaben gemachte Erfahrungen jeweils zeitnah ausgetauscht und reflektiert werden, um gegebenfalls auch dieses Dokument entsprechend anpassen zu können.