Schulinterner Lehrplan für das Fach Informatik am Max-Planck-Gymnasium zum Kernlehrplan für die gymnasiale Oberstufe

Informatik

(Stand: 01.02.2015)

Inhalt

1 Die Fachgruppe Informatik des Max-Planck-Gymnasiums Dortmund

Beim Max-Planck-Gymnasium handelt es sich um eine vierzügige Schule in der südlichen Innenstadt von Dortmund mit zurzeit ca. 1100 Schülerinnen und Schülern, 70 Planstellen und 90 Lehrerinnen und Lehrern. Das Einzugsgebiet der Schule umfasst den südlichen Teil der Dortmunder Innenstadt, die Vororte des Stadtbezirks Hombruch sowie Teile umliegender Städte und Stadtbezirke, was zum Teil auf das Angebot der Schule im Fach Informatik zurückzuführen ist.

Das Fach Informatik wird am Max-Planck-Gymnasium ab der Jahrgangsstufe 8 im Wahlpflichtbereich II (WP II) dreistündig unterrichtet und von etwa fünfundzwanzig 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 Robotik eingegangen. Der Unterricht erfolgt dabei in enger Verzahnung mit Inhalten der Mathematik und Physik und wird zum Teil in Form von fächerverbindenden Projekten und in Kooperation mit außerschulischen Partnern gestaltet.

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

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

Der Unterricht der Sekundarstufe II wird mit Hilfe der Programmiersprache Java durchgeführt. In der Einführungsphase kommt dabei zusätzlich eine didaktische Bibliothek zum Einsatz.

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 Max-Planck-Gymnasiums aus zwei Lehrkräften, denen zwei Computerräume mit 15 bzw. 16 Computerarbeitsplätzen zur Verfügung stehen. Alle Arbeitsplätze sind an das schulinterne Rechnernetz angeschlossen, so dass Schülerinnen und Schüler über einen individuell gestaltbaren Zugang zum zentralen Server der Schule alle Arbeitsplätze der zwei Räume zum Zugriff auf ihre eigenen Daten, zur Recherche im Internet oder zur Bearbeitung schulischer Aufgaben verwenden können.

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

2 Entscheidungen zum Unterricht

2.1 Unterrichtsvorhaben

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

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

Im „Übersichtsraster Unterrichtsvorhaben“ (Kapitel 2.1.1) wird die für alle Lehrerinnen und Lehrer gemäß Fachkonferenzbeschluss verbindliche Verteilung der Unterrichtsvorhaben dargestellt. Das Übersichtsraster dient dazu, den Kolleginnen und Kollegen einen schnellen Überblick über die Zuordnung der Unterrichtsvorhaben zu den einzelnen Jahrgangsstufen sowie den im Kernlehrplan genannten Kompetenzen, Inhaltsfeldern und inhaltlichen Schwerpunkten zu verschaffen. Der ausgewiesene Zeitbedarf versteht sich als grobe Orientierungsgröße, die nach Bedarf über- oder unterschritten werden kann. Um Freiraum für Vertiefungen, besondere Schülerinteressen, aktuelle Themen bzw. die Erfordernisse anderer besonderer Ereignisse (z.B. Praktika, Kursfahrten o.ä.) zu erhalten, wurden im Rahmen dieses schulinternen Lehrplans ca. 75 Prozent der Bruttounterrichtszeit verplant.

Während der Fachkonferenzbeschluss zum „Übersichtsraster Unterrichtsvorhaben“ zur Gewährleistung vergleichbarer Standards sowie zur Absicherung von Lerngruppenübertritten und Lehrkraftwechseln für alle Mitglieder der Fachkonferenz Bindekraft entfalten soll, beinhaltet die Ausweisung „konkretisierter Unterrichtsvorhaben“ (Kapitel 2.1.2) Beispiele und Materialien, die empfehlenden Charakter haben. Referendarinnen und Referendaren sowie neuen Kolleginnen und Kollegen dienen diese vor allem zur standardbezogenen Orientierung in der neuen Schule, aber auch zur Verdeutlichung von unterrichtsbezogenen fachgruppeninternen Absprachen zu didaktisch-methodischen Zugängen, fächerübergreifenden Kooperationen, Lernmitteln und -orten sowie vorgesehenen Leistungsüberprüfungen, die im Einzelnen auch den Kapiteln 2.2 bis 2.3 zu entnehmen sind.

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

2.1.1 Übersichtsraster Unterrichtsvorhaben

I) Einführungsphase

Einführungsphase	
Unterrichtsvorhaben E-I **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 • Dateisystem • Internet • Einsatz von Informatiksystemen **Zeitbedarf**: 4 Stunden	*Unterrichtsvorhaben E-II* **Thema**: *Grundlagen der objektorientierten Analyse, Modellierung und Implementierung anhand von statischen Grafikszenen* **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**: 10 Stunden

Einführungsphase	
Unterrichtsvorhaben E-III **Thema**: *Grundlagen der objektorientierten Programmierung und algorithmischer Grundstrukturen in Java anhand von benutzergesteuertes Zeichnen und einfachen Animationen (Dartspiel)* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen und Automaten **Inhaltliche Schwerpunkte:** • Objekte und Klassen • Syntax und Semantik einer Programmiersprache • Analyse, Entwurf und Implementierung einfacher Algorithmen **Zeitbedarf**: 12 Stunden	*Unterrichtsvorhaben E-IV* **Thema**: *Modellierung und Implementierung von Klassen- und Objektbeziehungen anhand von grafischen Spielen und Simulationen (Billard, Grafikprogramm, Zugprojekt)* **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 **Zeitbedarf**: 24 Stunden

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

Thema:
Grundzüge der Ereignisorientierung (Verwenden von Ereignisbearbeiter-Methoden)

Zentrale Kompetenzen:
• Argumentieren
• Modellieren
• 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

Zeitbedarf: 15 Stunden</td>
<td><u>Unterrichtsvorhaben E-VI</u>

Thema:
Anwendungen der rekursiven Programmierung

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

Inhaltsfelder:
• Algorithmen

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

Zeitbedarf: 9 Stunden</td>
</tr>
<tr><td colspan="2">Summe Einführungsphase: 74</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:
Modellierung und Implementierung von Anwendungen mit statischen Datenstrukturen (Array)

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

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

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

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

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

Zeitbedarf: 15 Stunden
</td>
</tr>
</table>

Qualifikationsphase 1	
Unterrichtsvorhaben Q1-III **Thema:** *Modellierung und Implementierung von Anwendungen mit den Verwaltungsstrukturen Schlange und Stapel* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Daten und ihre Strukturierung • Algorithmen • Formale Sprachen und Automaten **Inhaltliche Schwerpunkte:** • Objekte und Klassen • Analyse, Entwurf und Implementierung von Algorithmen • Algorithmen in ausgewählten informatischen Kontexten • Syntax und Semantik einer Programmiersprache **Zeitbedarf**: 15 Stunden	*Unterrichtsvorhaben Q1-IV* **Thema:** *Suchen und Sortieren auf linearen Datenstrukturen* **Zentrale Kompetenzen:** • Argumentieren • Modellieren • Implementieren • Darstellen und Interpretieren • Kommunizieren und Kooperieren **Inhaltsfelder:** • Algorithmen • Formale Sprachen und Automaten **Inhaltliche Schwerpunkte:** • Analyse, Entwurf und Implementierung von Algorithmen • Algorithmen in ausgewählten informatischen Kontexten • Syntax und Semantik einer Programmiersprache **Zeitbedarf**: 10 Stunden

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

<u>Unterrichtsvorhaben Q1-V</u>

Thema:
Modellierung und Implementierung von Anwendungen mit dynamischen, nichtlinearen Datenstrukturen (hier: Binärbaum)

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

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

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

Zeitbedarf: 15 Stunden

</td>
<td>

<u>Unterrichtsvorhaben Q1-VI</u>

Thema:
Codierung (Morse, Ascii, Huffman)

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

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

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

Zeitbedarf: 15 Stunden

</td>
</tr>
<tr><th colspan="2">Summe Qualifikationsphase 1: 78 Stunden</th></tr>
</table>

Qualifikationsphase 2	

Qualifikationsphase 2	
Summe Qualifikationsphase 2: 56 Stunden	

2.1.2 Konkretisierte Unterrichtsvorhaben

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

Sowohl in der Einführungsphase als auch in der Qualifikationsphase wird die didaktische Bibliothek SuM (Stifte und Mäuse) verwendet. Allen Schülerinnen und Schülern in der EF wird in der ersten Unterrichtsstunde (neben dem Lehrbuch „Informatik mit Java – Eine Einführung mit BlueJ und der Bibliothek Stifte und Mäuse Band I“ von Bernard Schriek, Nili-Verlag Werl) eine stets aktuelle Installations-Anleitung ausgeteilt, um sich diese Lernumgebung (BlueJ, Java, SuM) auf den heimischen Computern installieren zu können. Nähere Informationen zu diesem didaktischen Konzept sind unter der Adresse http://www.mg-werl.de/sum/ zu finden.

Ein Einsatz der didaktische Bibliothek GLOOP ist als Alternative zu SuM allerdings nur in der Einführungsphase ebenfalls in Zukunft denkbar. Informationen zu GLOOP sind unter folgender Adresse zu finden:
http://www.brd.nrw.de/lerntreffs/informatik/structure/material/sek2/einfuehrungen/gloop.php

I) Einführungsphase

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

Die Schülerinnen und Schüler

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

Unterrichtsvorhaben EF-I

Thema: 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 erarbeitet werden und mit dem grundlegenden Prinzip der Datenverarbeitung (Eingabe-Verarbeitung-Ausgabe) in Beziehung gesetzt werden.

Bei der Beschäftigung mit 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: 4 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 informatischen Systemen am Beispiel der Schulrechner (d) Vereinbarung von Richtlinien zur Datenspeicherung auf den Schulrechnern (z.B. Ordnerstruktur, Dateibezeichner usw.)	Die Schülerinnen und Schüler • beschreiben und erläutern den Aufbau und die Arbeitsweise singulärer Rechner (A), • lernen die Art und Weise der Anmeldung eines Rechners in einem Schulnetzwerk kennen und die Speicherung von eigenen Daten auf das Netzwerklaufwerk (in unserem Falle Laufwerk Z) (A), • nutzen die im Unterricht eingesetzten Informatiksysteme selbstständig, sicher, zielführend und verantwortungsbewusst (D), • nutzen das Internet zur Recherche, zum Datenaustausch und zur Kommunikation (K).	*Beispiel*: Schüler melden sich mit den erhaltenen Benutzerdaten am Schulrechner an, erstellen ein Word-Dokument (Schrift- und Bildelemente) und speichern dieses ab. Es muss hier der Unterschied deutlich gemacht werden, ob man unter dem „eigenen Laufwerk Z“ oder unter dem „öffentlichen Laufwerk P“ Dateien abspeichert. Die Ordnerstruktur wird erläutert und auf die Wichtigkeit von Dateibezeichnungen- und endungen Wert gelegt.
2. Informations- und Datenübermittlung in Netzen (a) Informatische Kommunikation in Rechnernetzen am Beispiel des Schulnetzwerks (z.B. Benutzeranmeldung, Netzwerkordner, Zugriffsrechte, Client-Server) (b) Richtlinien zum verantwortungsvollen Umgang mit dem Internet		*Beispiel*: Die Schüler kopieren sich die Installationsanleitung zu „SuM“ (bereitgestellt auf dem Laufwerk P) auf ihren eigenen privaten Bereich (Z). Anschließend kopieren sich die Schüler dieses auf einen USB-Stick. Alternativ kann das Internet genutzt werden, um sich diese Datei per E-Mail zu schicken. Dabei soll auf den verantwortungsvollen Umgang mit dem Internet eingegangen werden.

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)		*Material*: Demonstrationshardware Durch Demontage eines Demonstrationsrechners 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 von statischen Grafikszenen

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 UML-Klassendiagramme oder -Beziehungsdiagramme eingeführt. Zur späteren computergenerierten Erzeugung von UML-Diagrammen steht das Programm UMLed als Schullizenz zur Verfügung.

Im Anschluss wird mit der Realisierung erster Projekte mit Hilfe der didaktischen Programmierumgebung SuM 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: 10 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Identifikation von Objekten (a) Am Beispiel eines lebensweltnahen Beispiels werden Objekte im Sinne der Objektorientierten Modellierung eingeführt. (b) Die Unterschiedlichkeit der Begriffe Objekt und Klasse wird erläutert. (c) Manche Objekte sind prinzipiell typgleich und werden so zu einer Objektsorte bzw. Objektklasse zusammengefasst. (d) Vertiefung: Modellierung weiterer Beispiele ähnlichen Musters	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), • lernen die Punktnotation bei Dienstaufrufen kennen und verwenden diese (D), • implementieren einfache Algorithmen unter Beachtung der Syntax und Semantik einer Programmiersprache (I), • stellen den Zustand eines Objekts dar (D).	*Beispiel:* Einführung in die Konzeption „Objekt-Klasse“ Schülerinnen und Schüler betrachten (auf dem Schulhof) eine Menge gleichartiger Objekte (Autos), die in einer Klasse mit Attributen und Methoden zusammengefasst werden können.
2. Analyse von Klassen didaktischer Lernumgebungen (a) Objektorientierte Programmierung als modularisiertes Vorgehen (Entwicklung von Problemlösungen auf Grundlage vorhandener Klassen) (b) Teilanalyse der Klassen der didaktischen Lernumgebungen SuM		*Beispiel:* Kennenlernen der Basisklassen von SuM, „Hausprojekt“ Schülerinnen und Schüler lernen die Basisklassen von SuM (Bildschirm, Stift, Maus, Tastatur) kennen. Die Schüler sollen ein Haus (Haus des Nikolaus) auf dem Bildschirm zeichnen. Eventuell kann der Auftrag „zeichneHaus()“ mit Parametern für die Position, Länge der

		Hauskante, Stiftfarbe versehen werden.
3. Implementierung zweidimensionaler, statischer Szenen (a) Grundaufbau einer Java-Klasse (b) Deklaration und Initialisierung von Objekten (c) Methodenaufrufe mit Parameterübergabe zur Manipulation von Objekteigenschaften (z.B. Farbe, Position, Drehung)		*Beispiel: „*Zeichenübungen“ Schülerinnen und Schüler sollen einfache Figuren erstellen (gleichseitiges Dreieck, Quadrat, gleichseitiges Fünfeck, Stern). Es soll der Unterschied zwischen relativen und absoluten Zeichenbefehlen diskutiert werden („bewegeUm()“ und „bewegeBis()“).

Unterrichtsvorhaben EF-III

Thema: Grundlagen der objektorientierten Programmierung und algorithmischer Grundstrukturen in Java anhand von benutzergesteuertes Zeichnen und einfachen Animationen (Dartspiel)

Leitfragen: *Wie lassen sich Animationen und Simulationen optischer Gegenstandsbereiche unter Berücksichtigung von Maus- undTastatureingaben realisieren?*

Vorhabenbezogene Konkretisierung:
Der Schwerpunkt dieses Unterrichtsvorhabens liegt auf der Entwicklung mehrerer Projekte, die durch benutzergesteuerte Animationen gekennzeichnet sind. Zunächst wird ein Freihand-Projekt bearbeitet, bei dem in Anlehnung an das vorangegangene Unterrichtsvorhaben benutzergesteuertes Zeichnen (Freihandzeichnungen mit der Maus) möglich ist. Für die Umsetzung dieses Projekts werden Kontrollstrukturen in Form von Schleifen und Verzweigungen benötigt und eingeführt.

Sind an einem solchen Beispiel im Schwerpunkt Schleifen und Verzweigungen eingeführt worden, sollen diese Konzepte an weiteren Beispielprojekten (z.B. Dartspiel) eingeübt werden. Auch die Erzeugung grafischer Objekte und deren Verwaltung (evtl. in einem Feld) kann ein Anlass zur Verwendung von Kontrollstrukturen sein.

Das Dart-Projekt beinhaltet komplexe grafische Elemente, so dass die Schülerinnen und Schüler mehrere Klassen erstellen müssen, welche für das Dartspiel eine Rolle spielen. Elemente der Szene müssen zu sinnhaften eigenen Klassen zusammengefasst werden, die dann ihre eigenen Attribute und Dienste besitzen. Beim Dart-Projekt sollen Benutzeraktionen den Ablauf des Spiels steuern und am Ende eine Trefferauswertung erfolgen. Diese kann einfach oder auch komplexer angelegt werden.

Komplexere Assoziationsbeziehungen (wie z.B. kennt-Beziehungen) zwischen Klassen müssen in diesem Unterrichtsvorhaben zunächst nicht behandelt werden. Optional kann aber darauf eingegangen werden. Die Assoziationsbeziehungen zwischen Klassen stellen den Schwerpunkt des folgenden Vorhabens (EF-IV) dar.

Zeitbedarf: 12 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Bewegungsanimationen (z.B. Freihandzeichnen) (a) Kontinuierliche Verschiebung und Steuerung eines Objekts (Stift) mit Hilfe einer Schleife (while-Schleife) (b) Tastaturabfrage zur Realisierung einer Schleifenbedingung (Farbwechsel des Stifts) (c)	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), • 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),	*Beispiel:* Freihandzeichnen (Zeichnen mit dem Stift anhand von Bewegen der Maus und Tastatureingaben) Die Schüler sollen schrittweise Aufgaben zum Freihandzeichnen lösen. Hier werden die Kontrollstrukturen benötigt (if, while, switch-case).
2. Modellierung und Animation komplexerer grafisch repräsentierbarer Objekte (z.B. Dartspiel) (a) Modellierung eines Simulationsprogramms mit eigenen Klassen (b) Implementierung eigener Methoden mit und ohne Parameterübergabe (c) Mehrstufige Animationen mit mehreren sequenziellen Schleifen (d) Realisierung von Zustandsvariablen (e) Thematisierung des Geheimnisprinzips von Objekten (f) Animation mit Hilfe des Aufrufs von selbstimplementierten Methoden (g) Berechnung von Abständen zwischen Objekten mit Hilfsvariablen (h) Trefferauswertung mit Hilfe von Abstandsberechnungen und Verzweigungen (if-Anweisungen)		*Beispiel*: Dartspiel Die Schülerinnen und Schüler realisieren mit Objekten der SuM-Umgebung ein Dartspiel, bei dem ein Pfeil auf den Bildschirm fällt, bei Mausdruck dreht und beim Loslassen der Maus auf eine runde Zielscheibe zufliegt. Nach dem Steckenbleiben soll eine Trefferauswertung erfolgen.

	• testen Programme schrittweise anhand von Beispielen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I).	

Unterrichtsvorhaben EF-IV

Thema: Modellierung und Implementierung von Klassen- und Objektbeziehungen anhand von grafischen Spielen und Simulationen (Billard, Grafikprogramm, Zugprojekt)

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

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

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

Zeitbedarf: 24 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Entwicklung eines Spiels/Simulation (z.B. Billard) (a) Zunächst soll nur die Kugel von links nach rechts laufen. (b) Dann soll die Kugel am rechten Rand des Tisches abprallen (Einführung der Kennt-Beziehung, als neue Objektbeziehung neben der Hat-Beziehung) (c) Die Kugeln (laufen mittlerweile auch schief auf eine Wand zu) prallen an allen Wänden richtig ab. (d) Unterklassenbildung bzgl. der Klasse Kugel. Herstellung und Eigenschaften einer Ist-Beziehung. (e) Verallgemeinerung und Reflexion des Prinzips der Vererbung am Beispiel der Spezialisierung	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 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),	*Beispiel*: Billardsimulation Die Schülerinnen und Schüler entwickeln eine Simulation, bei der eine oder mehrere Kugeln über einen Tisch laufen und realistisch abbrallen. Bei der Unterklassenbildung: i. Reibekugel (Kugel wird beim Laufen langsamer) - Farbkugel (beim Abprallen an eine Wand ändert sich die Farbe der Kugel) - Nummernkugel (Die Kugel hat eine Nummer) - Pulsierkugel (Kugel wird beim Laufen immer größer, schrumpft aber beim Abprallen wieder schlagartig auf einen kleinen Radius) - Quadratkugel (Ein Quadrat bewegt sich auf dem Tisch) - Sternkugel (Ein Stern bewegt sich auf dem Tisch) Hier kann auch eine eigens geschriebene Klasse „Spezialstift“ verwendet werden, die eine Unterklasse von Buntstift ist, und die eigene Klassen (z.B. zeichneQuadrat()) hat.
2. Entwicklung einer komplexeren Simulation mit grafischen Elementen (Vererbung)		*Beispiel:* Grafikprogramm Die Schülerinnen und Schüler entwickeln objektorientiert ein Grafikprogramm, mit dem

(a) Generalisieren von Unterklassen (Kreise, Quadrate) zu einer Oberklasse, die als abstrakte Oberklasse sinnvoll geplant wird. (b) Reflexion des Prinzips der späten Bindung (die Variable „aktiveForm“ wird erst während der Laufzeit des Programms zu konkreten Objekten zugewiesen) (c) Prinzip einer abstrakten Oberklasse (d) Verkettung (ohne die Listenstruktur weiter auszuführen)	• 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).	auf dem Bildschirm farbige, verschieden große Kreise (und später auch Quadrate und andere Figuren) erzeugt werden. Diese sollen mit der Maus aktiviert und mit der Tastatur bewegt werden können. Die abstrakte Klasse „Form“ wird später unumgänglich. *Beispiel:* Zugprojekt Die Schülerinnen und Schüler entwickeln objektorientiert ein Programm, bei dem auf dem Bildschirm ein Zug gezeichnet werden soll, der (später) auch nach links fährt. Der Zug soll von einer Lokomotive gezogen werden und neben der Lokomotive aus Personen- und Güterwagen bestehen, welche (später) auch noch in beliebiger Reihenfolge (je nach Tastatureingabe: p für Personenwagen, g für Güterwagen) an den Zug angehängt werden sollen. Solange die Maustaste gedrückt gehalten wird, soll der Zug fahren, sobald die Maustaste losgelassen wird, stoppt wieder der Zug. Durch Doppelklick wir das Hauptprogramm beendet. Fährt der Zug von links aus dem Bild heraus, so soll er etwas nach unten versetzt wieder von rechts ins Bild fahren. Beim „Ankuppeln“ benötigt man eine Art der Verkettung von Objekten (später: Hinführung zur Datenstruktur Liste)

Unterrichtsvorhaben EF-V

Thema: Ereignisorientierte Programmierung *Grundzüge der Ereignisorientierung (Verwenden von Ereignisbearbeiter-Methoden)*

Leitfragen: *Wie kann ereignisorientiert programmiert werden?*

Vorhabenbezogene Konkretisierung:
Dieses Unterrichtsvorhaben beschäftigt sich mit der Ereignisorientierung. Angefangen mit Reaktionstabellen, werden Knöpfe selbst programmiert, indem auf die SuM-Klasse „Ereignisanwendung“ zurückgegriffen wird. Neben der Erarbeitung der Phasen eines Softwareprojekts wird mit dem SuM-Programmgenerator einfache Oberflächen gestaltet und das MVC-Modell kennengelernt anhand einiger einfacher Softwareprojekte (z.B. Quadraterzeuger, Taschenrechner, Währungsrechner, Matheübung, Spielautomat, Bruchrechner).

Zeitbedarf: 15 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Ereignisorientierung (a) Reaktionstabellen (b) Ereignisanwedung (c) SuM-Programmgenerator (d) MVC-Modell	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	*Beispiel:* Freihand mit EA Die Schülerinnen und Schüler modifizieren die alte Aufgabe zum Freihandzeichnen und lösen die Aufgabe mithilfe der EA (Ereignisanwendung) *Alternatives Beispiel:* Gummilinien *Beispiel:* Quadraterzeuger (es soll ein kleines Projekt erzeugt werden, bei dem auf Knopfdruck das Quadrat einer vorher vom Benutzer einzugebenden Zahl berechnet und ausgegeben wird. *Beispiele:* Taschenrechner, Währungsumrechner, Matheübung, Spielautomat, Bruch-

	Rückgaben von Methoden einfache Datentypen, Objekttypen oder lineare Datensammlungen zu (M), • ordnen Klassen, Attributen und Methoden ihren Sichtbarkeitsbereich zu (M), • modellieren Klassen unter Verwendung von Vererbung (M), • implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I), • testen Programme schrittweise anhand von Beispielen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • modifizieren einfache Algorithmen und Programme (I), • stellen Klassen, Assoziations- und Vererbungsbeziehungen in Diagrammen grafisch dar (D), • dokumentieren Klassen durch Beschreibung der Funktionalität der Methoden (D).	rechner

Unterrichtsvorhaben EF-VI

Thema: *Anwendungen der rekursiven Programmierung*

Leitfrage: *Vergleich iterativer und rekursiver Methoden und Sichtweisen*

Vorhabenbezogene Konkretisierung: Kennenlernen der rekursiven Sichtweise, da diese später bei der Datenstruktur Liste wichtig ist.

Zeitbedarf: 9 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Erarbeitung des Prinzips der Rekursion (a) Je nach Zeit am Ende der EF bieten sich kleine Beispiele „aus der Mathematik“ an, um die Rekursion zu verdeutlichen (Fakultät, Fibonacci). (b) Rekursion bei anderen Situationen (alternative Beispiele)	Die Schülerinnen und Schüler • 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).	*Beispiel:* Fakultät Es soll die Fakultät einer Zahl berechnet werden. Hierbei ist es sinnvoll auf den Zahlenbereich int, long einzugehen. Hier bietet es sich an, die Java-Klasse BigInteger einzuführen. *Beispiel:* Fibonacci Schülerinnen und Schüler recherchieren selbstständig im Internet nach der Fibonacci-Folge. Dann sollen Sie ein Programm schrieben, dass die n-te Fib.-Zahl ausgibt. *Alternatives Beispiel*: Rekursion und Grafik *Alternatives Beispiel*: Palindrom

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).

Falls jemand (z.B. als Seiteneinsteiger von der Realschule) noch die Anleitung zum Installieren von BlueJ / SuM und Java benötigt, wird ihm diese ausgeteilt und kurz darauf eingegangen. In der Regel ist dies den Schülern aber schon aus der EF bekannt.

Unterrichtsvorhaben Q1-I:

Thema: *Modellierung und Implementierung von Anwendungen mit statischen Datenstrukturen (Array)*

Leitfrage: *Wie kann eine vorher festgelegte Anzahl von Daten im Anwendungskontext sinnvoll verwaltet werden?*

Vorhabenbezogene Konkretisierung:
Anhand z.B. des Beispiels „Lottozahlen“ oder“ Augensumme“ wird die Benutzung der Datenstruktur Array eingeführt. Dadurch ist es es nun nicht mehr nötig, viele Variablen eines Datentyps zu deklarieren, da man nun einfach ein Array deklarieren kann und auf die Arrayelemnte direkt zugreifen kann.

Zeitbedarf: 8 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Die Datenstruktur Array im Anwendungskontext (a) Deklaration eines Arrays (b) Erzeugung eines Arrays (c) Zugriff auf Arrayelemente	Die Schülerinnen und Schüler • erläutern Operationen statischer Datenstrukturen (A), • analysieren und erläutern Algorithmen und Programme (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu (M), • ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • modifizieren Algorithmen und Programme (I), • 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 statische Strukturen grafisch dar und erläutern ihren Aufbau (D).	*Beispiel:* Lottozahlen-Projekt Es sollen 6 unterschiedliche ganze Zahlen von 1 bis 49 (also Lottozahlen) vom Benutzer nacheinander einzelnd eingegeben (und in einem Array abgespeichert) werden. Auf Knopfdruck sollen anschließend die vorher eingegebenen Zahlen ausgegeben werden (zusätzlich auch die Summe und das Minimum der Zahlen). Desweiteren kann zusätzlich noch die Suche nach einer in ein Textfeld eingegebenen Zahl realisiert werden. *Beispiel:* Augensumme-Projekt Es sollen die Häufigkeiten der auftretenden Augensummen beim mehrfachen Würfeln eines oder mehrerer Würfel untersucht werden. dabei sollen diese Häufigkeiten gezählt und angezeigt werden. So soll ein Benutzer neben der Anzahl der Würfe auch die Anzahl der Würfel eingeben können, und dann soll ein mehrfaches Würfeln simuliert werden, und die Augensummen-Häufigkeiten ausgegeben 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:
Anhand des Projekts Kursliste, wobei zunächst Schülerojekte der Klasse Schueler in einem Array aufgenommen werden, soll diskutiert werden, wie man eine dynamische Datenstruktur selbst planen und implementieren kann. Durch die Klasse ListenElement wird eine lineare Datenstruktur benutzt, um die Funktionalitäten "Neu anlegen", "Zeige Liste", "Suche", "Ändern" und "Löschen" beim Kurslistenprojekt zu realisieren. Unterschiede zwischen Array und Liste sollen deutlich werden.

Zeitbedarf: 15 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. Die "selbstgeschriebene" Datenstruktur ListenElement beim Projekt "Kursliste" **2. Die Datenstruktur Liste aus sum.strukturen (incl. Typcast). Diese Verallgemeinerung kann evtl. auch erst im Unterrichtsvorhaben III diskutiert werden.**	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 Problem-	*Beispiel:* Kursliste Nach der Eingabe der Attribute eines Schülers (Name, Vorname, Kurs, Telefonnummer und E-Mail) kann immer wieder ein neuer Datensatz in die Datenstruktur Liste eingefügt werden. Die anderen Funktionalitäten "Zeige Liste", "Suche", "Ändern" und "Löschen" werden in der SumAnwendung (Oberfläche) direkt implementiert. In der Klasse ListenElement selbst wird aber ein Dienst zum Hinzufügen eines Schülers implementiert.

	stellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • modifizieren Algorithmen und Programme (I), • implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • testen Programme systematisch anhand von Beispielen (I), • stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D).	

Unterrichtsvorhaben Q1-III:

Thema: Modellierung und Implementierung von Anwendungen mit den Verwaltungsstrukturen Schlange und Stapel.

Leitfrage: *Wie können in einer Datenstruktur gespeicherte Daten im Anwendungskontext verwaltet werden?*

Vorhabenbezogene Konkretisierung:
Anhand des Projekts Artztpraxis werden Daten nach dem First-In-First-Out-Prinzip verwaltet. Es wird der Aufbau von Schlangen am Beispiel dargestellt und die Operationen der Klasse `Schlange (Wartezimmer)` erläutert. Anschließend werden für die Anwendung notwendige Klassen modelliert und implementiert. Anschließend kann die selbstgeschriebene Schlange durch die allgemeine Schlange aus sum.strukturen ersetzt werden. Dabei ist ein Typcast nötig. Anschließend wird die Anwendung auf Last-In-First-Out-Prinzip umgeschrieben

und Unterschiede zwischen den Datenstrukturen Schlange und Stapel erarbeitet. Um einfacher an Objekte zu gelangen, wird die Klasse `Liste` aus sum.strukturen eingeführt und in einem Anwendungskontext verwendet. Anschließend kann man das Projekt Terminator durchführen, um die Klasse Stapel einzuüben. Modellierungen werden dabei in Entwurfs- und Implementationsdiagrammen dargestellt.

Zeitbedarf: 15 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	**Zu entwickelnde Kompetenzen**	**Beispiele, Medien, Materialien**
1. Die Datenstruktur Schlange im Anwendungskontext unter Nutzung der Klasse ListenElement (selbstgeschrieben, vgl. Unterrichtsvorhaben II) und später unter der Nutzung der allgemeinen Klasse Liste (bzw. Schlange/Stapel) aus sum.strukturen. **2. Unterschiede des FIFO und LIFO-Prinzips**	Die Schülerinnen und Schüler • erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A), • analysieren und erläutern Algorithmen und Programme (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu (M), • ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • modifizieren Algorithmen und Programme (I), • implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • nutzen die Syntax und Semantik einer	*Beispiel: Arztpraxis / Patientenwarteschlange* (jeder kennt seinen Nachfolger) Sobald ein Patient in einer Arztpraxis eintrifft, werden sein Name und seine Krankenkasse erfasst. Die Verwaltung der Patientenwarteschlange geschieht über eine Klasse, die hier als Wartezimmer bezeichnet wird. Wesentliche Operationen sind das „Hinzufügen“ eines Patienten und das „Entfernen“ eines Patienten, wenn er zur Behandlung gerufen wird. Die Simulationsanwendung stellt eine GUI zur Verfügung, legt ein Wartezimmer an und steuert die Abläufe. Wesentlicher Aspekt des Projektes ist die Modellierung des Wartezimmers mit Hilfe der Klasse Schlange (auch zunächst als "Wartezimmer" bezeichnet). *Beispiel (optional): "Omas Rezepte"* Hier wird die allgemeine Klasse Liste benutzt, um eine Verwaltung von Rezepten zu realisieren. Jedes Rezept hat natürlich einen Namen, eine Personenanzahl und eine Zutatenliste. Dabei macht im Sinne der OOP auch eine Klasse Zutat Sinn, die einen Namen und

	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).	eine Menge als Attribute hat. *Beispiel: Terminator* (Termberechnungen anhand des Stapelprinzips)

Unterrichtsvorhaben Q1-IV:

Thema: Suchen und Sortieren auf linearen Datenstrukturen

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

Vorhabenbezogene Konkretisierung:
In einem Anwendungskontext werden zunächst Informationen in einer linearen Liste bzw. einem Feld gesucht. Hierzu werden Verfahren diskutiert und die "Binäre Suche" analysiert und erläutert.

Anschließend werden Sortierverfahren entwickelt und implementiert (ebenfalls für lineare Listen und Felder). Hierbei werden neben Bubblesort und Selectionsort (Max-/Minsort) auch das Verfahren des Insertsorts besprochen. Die Implementationen eines Indexbasiertes Suchverfajhrens kann ebenfalls durchgeführt werden (Indexsort).

Dabei werden die verschiedenen Sortierverfahren hinsichtlich der Anzahl der benötigten Vergleichsoperationen und des Speicherbedarfs beurteilt.

Zeitbedarf: 10 Stunden

Sequenzierung des Unterrichtsvorhabens:

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Suchen von Daten in Listen und Arrays (a) Lineare Suche in Listen und in Arrays (b) Binäre Suche in Arrays als Beispiel für rekursives Problemlösen (c) Untersuchung der beiden Suchverfahren hinsichtlich ihrer Effizienz (Laufzeitverhalten, Speicherbedarf) **2. Sortieren in Listen und Arrays - Entwicklung und Implementierung von Sortierverfahren** (a) Bubblesort (b) Maxsort (c) Minsort **3. Untersuchung der Effizienz der Sortierverfahren „Sortieren durch direktes Einfügen“ (Insertsort), evtl. Indexsort** (a) Untersuchung der Anzahl der Vergleichsoperationen und des Speicherbedarf bei beiden Sortierverfahren (b) Beurteilung der Effizienz der beiden Sortierverfahren	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), • modifizieren Algorithmen und Programme (I), • implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • implementieren und erläutern iterative und rekursive Such- und Sortierverfahren (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • interpretieren Fehlermeldungen und korrigieren den Quellcode (I), • testen Programme systematisch anhand von Beispielen (I), • stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D).	*Beispiele:* Bubblesort, Selectionsort (Max-/Minsort), Insertsort, Indexsort Dabei: Sortieren von int-Zahlen (Zufallszahlen), von Strings, von Objekten wie "Fußballvereine"

Unterrichtsvorhaben Q1-V:

Thema: Modellierung und Implementierung von Anwendungen mit dynamischen, nichtlinearen Datenstrukturen.
Hier: Die Datenstruktur Binärbaum

Leitfragen*:*
- *Wie können Daten im Anwendungskontext mit Hilfe binärer Baumstrukturen verwaltet werden?*
- *Wie kann dabei der rekursive Aufbau der Baumstruktur genutzt werden?*
- *Welche Vor- und Nachteile haben Suchbäume für die geordnete Verwaltung von Daten?*
- *Wie kann eine Datenstruktur direkt die binäre Suche unterstützen?*

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 Baum aus sum.strukturen

Die Funktionsweise von Methoden wird anhand grafischer Darstellungen von Binärbäumen erläutert.

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

Zeitbedarf: 15 Stunden

Sequenzierung des Unterrichtsvorhabens

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Die Datenstruktur Binärbaum (a) Planung und Implementierung der Klasse Knoten (b) Speichern von Zahl- bzw. Personenobjekten in einer Baumstruktur (Einfügen, Ausgeben, Suchen, Anzeige von Min/Max)	Die Schülerinnen und Schüler • erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A), • analysieren und erläutern Algorithmen und Programme (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu (M), • ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • modifizieren Algorithmen und Programme (I), • implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I), • nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I), • 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).	*Beispiel:* Baum Es werden nicht nur Zahl-, sondern auch Personen-Objekte in eine Baumstruktur aufgenommen und verwaltet (Anzeigen via in-order, pre-order, post-order). Dabei wird deutlich, dass auch die binäre Suche allein durch die Verwendung des Binärbaums direkt beim Suchvorgang unterstützrt wird.

Unterrichtsvorhaben Q1-VI:

Thema: *Codierung (Morse, Ascii, Huffman)*

Leitfragen*:* *Wozu dient Codierung?*
Wie ist der Morsecode, Asciicode, Huffman-Code aufgebaut?
Wie kann man ein Übersetzungsprogramm für die Übersetzung in Morsecode, Asciicode, Huffman-Code planen und implementieren?

Zeitbedarf: 15 Stunden

Sequenzierung des Unterrichtsvorhabens

Unterrichtssequenzen	Zu entwickelnde Kompetenzen	Beispiele, Medien, Materialien
1. Codierverfahren (a) Morsecode (b) Asciicode (c) Huffman-Code (optimaler Code: möglichst kurze Bitlänge), Fano-Bedingung, Implementierung der Schritte zur Erstellen eines Huffmancodes	Die Schülerinnen und Schüler • erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A), • analysieren und erläutern Algorithmen und Programme (A), • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A), • ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M), • 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).	*Beispiel: Morsecode* Es wird anhand einer grafischen Benutzeroberfläche ein Text (deutscher Text) in Mosecode übersetzt und in einem Zeichenbereich ausgegeben. Dabei kann die Datenstruktur, die das Morsealphabet benutzt, von den Schülern frei gewählt werden. Es können für die Speicherung der Buchstaben und Codes Arrays oder auch Listen benutzt werden. Es kann auch eine eigens für diesen Zweck angelegte Klasse CuB (CodeUndBuchstabe) geplant und implementiert werden. Alternative: Die Morsecodierungen können in einem Binärbaum dargestellt werden, so dass ein Übergang zum linken Teilbaum einem Punkt und ein Übergang zum rechten Teilbaum einem Strich entspricht. Wenn man im Gesamtbaum startet und durch Übergänge zu linken oder rechten Teilbäumen einen Pfad zum gewünschten Buchstaben sucht, erhält man die Morsecodierung des Buchstabens. *Beispiel: Ascii-Code* Ähnlich zu dem Beispiel Ascii-Code. Hier kann auch die hexadezimale Darstellung eines Zeiches erläutert werden. *Beispiel: Huffman-Code* Wie findet man einen optimalen Binärcode,

		dessen Gesamtcodewortlänge möglichst klein ist? Die Strategie des Erstellens eines Huffmancodes (unter Verwendung der Datenstruktur Binärbaum) wird auch implementiert. So wird am Ende in einer grafischen Oberfläche ein Text auf Knopfdruck in einen Huffmancode samt Codearray überführt. Dabei wird auch der Grad der "Komprimierung" im Vergleich zum reinen 8bit Ascii-Code berechnet.

Die konkretisierten Unterrichtsvorhaben für die Qualifikationsphase Q2 werden noch ergänzt.

Dabei erstes Thema: Sicherheit und Datenschutz in Netzstrukturen

2.2 Grundsätze der fachmethodischen und fachdidaktischen Arbeit

In Absprache mit der Lehrerkonferenz sowie unter Berücksichtigung des Schulprogramms hat die Fachkonferenz Informatik des Konrad-Zuse-Gymnasiums die folgenden fachmethodischen und fachdidaktischen Grundsätze beschlossen. In diesem Zusammenhang beziehen sich die Grundsätze 1 bis 14 auf fächerübergreifende Aspekte, die auch Gegenstand der Qualitätsanalyse sind, die Grundsätze 15 bis 21 sind fachspezifisch angelegt.

Überfachliche Grundsätze:

1) Geeignete Problemstellungen zeichnen die Ziele des Unterrichts vor und bestimmen die Struktur der Lernprozesse.
2) Inhalt und Anforderungsniveau des Unterrichts entsprechen dem Leistungsvermögen der Schüler/innen.
3) Die Unterrichtsgestaltung ist auf die Ziele und Inhalte abgestimmt.
4) Medien und Arbeitsmittel sind schülernah gewählt.
5) Die Schüler/innen erreichen einen Lernzuwachs.
6) Der Unterricht fördert eine aktive Teilnahme der Schüler/innen.
7) Der Unterricht fördert die Zusammenarbeit zwischen den Schülern/innen und bietet ihnen Möglichkeiten zu eigenen Lösungen.
8) Der Unterricht berücksichtigt die individuellen Lernwege der einzelnen Schüler/innen.
9) Die Schüler/innen erhalten Gelegenheit zu selbstständiger Arbeit und werden dabei unterstützt.
10) Der Unterricht fördert strukturierte und funktionale Partner- bzw. Gruppenarbeit.
11) Der Unterricht fördert strukturierte und funktionale Arbeit im Plenum.
12) Die Lernumgebung ist vorbereitet; der Ordnungsrahmen wird eingehalten.
13) Die Lehr- und Lernzeit wird intensiv für Unterrichtszwecke genutzt.
14) Es herrscht ein positives pädagogisches Klima im Unterricht.

Fachliche Grundsätze:

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

2.3 Grundsätze der Leistungsbewertung und Leistungsrückmeldung

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

2.3.1 Beurteilungsbereich Klausuren

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

Instrumente:

- Einführungsphase: 1 Klausur je Halbjahr
  Dauer der Klausur: 2 Unterrichtsstunden
- Grundkurse Q 1: 2 Klausuren je Halbjahr
  Dauer der Klausuren: 2 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 der Q1 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 Hilfspunktsumme zu den Notenstufen an dem Zuordnungsschema des Zentralabiturs.
Von diesem kann aber im Einzelfall begründet abgewichen werden, wenn sich z.B. besonders originelle Teillösungen nicht durch Hilfspunkte gemäß den Kriterien des Erwartungshorizontes abbilden lassen oder eine Abwertung wegen besonders schwacher Darstellung (APO-GOSt §13 (2)) angemessen erscheint.

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

2.3.2 Beurteilungsbereich Sonstige Mitarbeit

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

Leistungsaspekte

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

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

Sonstige schriftliche Leistungen
- Lernerfolgsüberprüfung durch kurze schriftliche Übungen
  In Kursen, in denen höchstens 50% der Kursmitglieder eine Klausur schreiben, finden schriftliche Übungen mindestens einmal pro Kurshalbjahr statt, in anderen Kursen entscheidet über die Durchführung die Lehrkraft.
  Schriftliche Übung dauern ca. 20 Minuten und umfassen den Stoff der letzten ca. 4–6 Stunden.
- Bearbeitung von schriftlichen Aufgaben im Unterricht

Kriterien

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

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

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

Bei Gruppenarbeiten auch auf
- das Einbringen in die Arbeit der Gruppe,

- die Durchführung fachlicher Arbeitsanteile und
- die Qualität des entwickelten Produktes.

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 Fach in der Qualifikationsphase.

3 Entscheidungen zu fach- und unterrichtsübergreifenden Fragen

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

Zusammenarbeit mit anderen Fächern
Im Informatikunterricht werden Kompetenzen anhand informatischer Inhalte in verschiedenen Anwendungskontexten erworben, in denen Schülerinnen und Schülern aus anderen Fächern Kenntnisse mitbringen können. Diese können insbesondere bei der Auswahl und Bearbeitung von Softwareprojekten berücksichtigt werden und in einem hinsichtlich der informatischen Problemstellung angemessenem Maß in den Unterricht Eingang finden. Da im Inhaltsfeld Informatik, Mensch und Gesellschaft auch gesellschaftliche und ethische Fragen im Unterricht angesprochen werden, kann eine mögliche Zusammenarbeit mit den Fächern Sozialwissenschaften und Philosophie in einer gemeinsamen Fachkonferenz ausgelotet werden.

Projekttage
Alle zwei Jahre werden am Max-Planck-Gymnasium Projekttage angeboten. Die Fachkonferenz Informatik bietet in diesem Zusammenhang mindestens ein Projekt mit informatischem Themenbezug für Schülerinnen und Schüler an.

Vorbereitung auf die Erstellung der Facharbeit
Möglichst schon im zweiten Halbjahr der Einführungsphase, spätestens jedoch im ersten Halbjahr des ersten Jahres der Qualifikationsphase werden im Unterricht an geeigneten Stellen Hinweise zur Erstellung von Facharbeiten gegeben. Das betrifft u. a. Themenvorschläge, Hinweise zu den Anforderungen und zur Bewertung. Es wird vereinbart, dass nur Facharbeiten vergeben werden, die mit der eigenständigen Entwicklung eines Softwareproduktes verbunden sind.

4 Qualitätssicherung und Evaluation

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

Das schulinterne Curriculum (siehe 2.1) ist zunächst bis 2017 für den ersten Durchgang durch die gymnasiale Oberstufe nach Erlass des Kernlehrplanes verbindlich. Erstmalig nach Ende der Einführungsphase im Sommer 2015, werden in einer Sitzung der Fachkonferenz Erfahrungen ausgetauscht und ggf. Änderungen für den nächsten Durchgang der Einführungsphase beschlossen, um erkannten ungünstigen Entscheidungen schnellstmöglich entgegenwirken zu können.

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