# Lehrplan Informatik

## 2.1.1 Übersichtsraster:

### Unterrichtsvorhaben in der Qualikationsphase (Q1)

| Nr. | Beschreibung |
|---|---|
| 1 | **Thema:** Wiederholung und Vertiefung der objektorientierten Modellierung (Memory/Hangman)<br>15 Stunden |
| 2 | **Thema:** Algorithmen zum Suchen und Sortieren auf linearen Datenstrukturen (Arrays)<br>12 Stunden |
| 3 | **Thema:** Modellierung und Implementierung von Anwendungen mit dynamischen und linearen Datenstrukturen (Stack, Queue, Liste, Array)<br>30 Stunden |
| 4 | **Thema**: Modellierung und Implementierung von Anwendungen mit dynamischen nicht-linearen Datenstrukturen (Binärnäume, Suchbäume, rekursive Algorithmen)<br>18 Stunden |

## 2.1.2 Konkretisierte Unterrichtsvorhaben

### Unterrichtsvorhaben Nr. 1

**Thema**
Wiederholung und Vertiefung der objektorientierten Modellierung

**Leitfragen**
Wie wird aus einem anwendungsbezogenen Sachkontext ein informatisches Klassenmodell entwickelt? Wie werden Attribute, Methoden und Beziehungen identifiziert, den Klassen zugeordnet und dargestellt?

**Inhaltliche Schwerpunkte**
- Objekte und Klassen
- Struktogramme

**Inhaltsfelder**
- Daten und ihre Strukturierung
- Algorithmen

**Zeitbedarf**: 15 Stunden

**Vorhabenbezogene Konkretisierung**
Der bereits bekannte objektorientierte Zugang zu informatischer Modellierung wird von einer allgemeinen Betrachtung dieses informatischen Konzepts auf eine konkrete Problematik übertragen. Anhand dieser wird eine anwendungsbezogene Implementation Schritt für Schritt von der Objektidentifikation über das Entwurfs- und Implementationsdiagramm durchlaufen.
Grundlegende Modellierungskonzepte wie Sichtbarkeiten, Assoziationen, Vererbung sowie deren Darstellung in Entwurfs- und Klassendiagrammen und Dokumentationen werden wiederholt. Ebenso wird erneut die grafische Darstellung von Objektkommunikation thematisiert.

| **Unterrichtssequenzen** | **Zu entwickelnde Kompetenzen** | **Beispiele, Medien oder Materialien** |
|---|---|---|
| 1. Wiederholung der grundlegenden Konzepte der objektorientierten Programmierung<br><br>a) Sichtweise der objektorientierten Informatik auf die Welt<br><br>b) OOP als informatikspezifische Modellierung der Realität<br><br>c) Schritte der Softwareentwicklung | Die Schülerinnen und Schüler<br><br>• analysieren und erläutern objektorientierte Modellierungen (A)<br><br>• modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (M)<br><br>• ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M)<br><br>• modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren | *Beispiele:*<br>geeignete Spiele (Memory, Hangman...) |
| 2. Erweiterung der objektorientierten Programmierung<br><br>a) Umsetzung einer Anforderung in Entwurfs- und Klassendiagramm<br><br>b) Klassendokumentation<br><br>c) Umsetzung von Teilen der Modellierung | | |

| | | |
|---|---|---|
| 3. Übung und Vertiefung der OOM / OOP | (M)<br>• nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I)<br>• wenden eine didaktisch orientierte Entwicklungsumgebung zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an (I)<br>• stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar (D)<br>• dokumentieren Klassen (D)<br>• stellen die Kommunikation zwischen Objekten grafisch dar (D)<br>• nutzen die im Unterricht eingesetzten Informatiksysteme selbstständig, sicher, zielführend und verantwortungsbewusst (D). | |

**Unterrichtsvorhaben Nr. 2**

**Thema:**

Algorithmen zum Suchen und Sortieren auf linearen Datenstrukturen

**Leitfrage:**

*Nach welchen Grundprinzipien können Algorithmen strukturiert werden? Welche Qualitätseigenschaften sollten Algorithmen erfüllen? Wie können mithilfe von Such- und Sortieralgorithmen Daten in linearen Strukturen effizient (wieder-)gefunden werden?*

**Inhaltliche Schwerpunkte**

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

**Inhaltsfelder**

- Algorithmen
- Formale Sprachen und Automaten

**Vorhabenbezogene Konkretisierung:**

Zunächst werden anhand eines Anwendungsbeispiels übergreifende Algorithmuseigenschaften (wie Korrektheit, Effizienz und Verständlichkeit) erarbeitet und Schritte der Algorithmusentwicklung wiederholt. Dabei kommen Struktogramme zur Darstellung von Algorithmen zum Einsatz.

Als besondere Struktur von Algorithmen wird die Rekursion an Beispielen veranschaulicht und gegenüber der Iteration abgegrenzt. Rekursive Algorithmen werden von den Schülerinnen und Schülern analysiert und selbst entwickelt.

In der zweiten Unterrichtssequenz geht es um die Frage, wie Daten in linearen Strukturen (lineare Liste und Array) (wieder-)gefunden werden können. Die lineare Suche als iteratives und die binäre Suche als rekursives Verfahren werden veranschaulicht und implementiert. Eine Bewertung der Algorithmen erfolgt, indem jeweils die Anzahl der Vergleichsoperationen und der Speicherbedarf ermittelt wird.

Möchte man Daten effizient in einer linearen Struktur wiederfinden, so rückt zwangsläufig die Frage nach einer Sortierstrategie in den Fokus. Es wird mindestens ein iteratives und ein rekursives Sortierverfahren erarbeitet und implementiert sowie ihre Effizienz bewertet.

**Zeitbedarf**: 21 Stunden

**Sequenzierung des Unterrichtsvorhabens:**

| **Unterrichtssequenzen** | **Zu entwickelnde Kompetenzen** | **Beispiele, Medien oder Materialien** |
|---|---|---|
| 1. Eigenschaften von Algorithmen<br><br>a) Qualitätseigenschaften von Algorithmen<br><br>b) Strukturierung von Algorithmen mit Hilfe der Strategien „Modularisierung“ und „Teile und Herrsche“<br><br>c) Analyse und Entwicklung von rekursiven Algorithmen | Die Schülerinnen und Schüler<br><br>• analysieren und erläutern Algorithmen und Programme (A),<br><br>• modifizieren Algorithmen und Programme (I),<br><br>• stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D),<br><br>• entwickeln iterative und rekursive Algorithmen unter Nutzung der Strategien „Modularisierung“ und „Teilen und Herrschen“ (M),<br><br>• implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I),<br><br>• testen Programme systematisch anhand von Beispielen (I). | *Beispiele:*<br><br>Matheprisma der Universität Wuppertal: Sortierverfahren<br>Rahmenprogramm Suchverfahren |
| 2. Suchen in Arrays<br><br>a) Lineare Suche in einem Array<br><br>b) Binäre Suche in einem Array<br><br>c) Untersuchung der beiden Verfahren bzgl. Laufzeit und Speicherplatzbedarf | | |

| | | |
|---|---|---|
| 3. Sortieren auf Arrays<br><br>a) Entwicklung und Implementierung eines iterativen Sortierverfahrens für eine Liste<br><br>b) Entwicklung und Implementierung eines rekursiven Sortierverfahrens für ein Array<br><br>c) Untersuchung der beiden Verfahren bzgl. Laufzeit und Speicherplatzbedarf | • implementieren und erläutern iterative und rekursive Suchund Sortierverfahren (I),<br><br>• beurteilen die Effizienz von Algorithmen unter Berücksichtigung des Speicherbedarfs und der Zahl der Operationen (A),<br><br>• beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A),<br><br>• nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I),<br><br>• interpretieren Fehlermeldungen und korrigieren den Quellcode (I),<br><br>• testen Programme systematisch anhand von Beispielen (I) | |

**Unterrichtsvorhaben Nr. 3**

**Thema:**

Modellierung und Implementierung von Anwendungen mit dynamischen und linearen Datenstrukturen

**Leitfragen:**
Wie müssen Daten linear strukturiert werden, um in den gestellten Anwendungsszenarien eine beliebige Anzahl von Objekten verwalten zu können?

**Inhaltliche Schwerpunkte**

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

**Inhaltsfelder**

- Daten und ihre Strukturierung
- Algorithmen

**Vorhabenbezogene Konkretisierung:**

Ausgehend von einigen Alltagsbeispielen werden als Erstes die Anforderungen an eine Datenstruktur erschlossen. Anschließend werden die Möglichkeiten des Arrays untersucht, lineare Daten zu verwalten und über deren Grenzen/Probleme die Vorteile einer dynamischen linearen Struktur am Beispiel der Struktur Queue erarbeitet. Die Klasse Queue selbst wird vorgegeben, die Operationen erläutert. Zur Vertiefung der Kenntnisse wird ein weiteres Anwendungsszenario eingeführt, dessen Lösung modelliert und implementiert wird. Darauf folgt die Erarbeitung der Struktur Stack, die mithilfe eines einfachen Anwendungsszenarios eingeführt wird. Auch hier wird die Klasse Stack selbst vorgegeben und die Operationen erläutert. Weitere Aufgaben dienen der Vertiefung und Sicherung.

Um die Unterschiede der beiden Prinzipien FIFO und LIFO zu verstehen, werden zur Lösung der Aufgaben sowohl der Stack als auch die Queue benötigt.

Als letzte lineare dynamische Datenstruktur wird die Liste eingeführt. In dieser Sequenz liegt der Fokus auf der Möglichkeit, auf jedes Element zugreifen zu können. Nachdem die umfangreicheren Standardoperationen dieser Datenstruktur in einem einführenden Beispiel erarbeitet und in einem weiteren Beispiel vertieft wurden, werden abschließend in einem Anwendungskontext verschiedene lineare Datenstrukturen angewendet. Die Modellierung erfolgt beim gesamten Vorhaben in Entwurfs- und Implementationsdiagrammen.

**Zeitbedarf**: 21 Stunden

**Sequenzierung des Unterrichtsvorhabens:**

| **Unterrichtssequenzen** | **Zu entwickelnde Kompetenzend** | **Beispiele, Medien oder Materialien** |
|---|---|---|
| 1. Die Datenstruktur Schlange<br><br>a) Modellierung und Implementierung der Verknüpfung von Objekten<br><br>b) Generische Typen, Trennung von Verwaltung und Inhalt dyn. DS.<br><br>c) Erläuterung von Problemstellungen, die nach dem FIFO-Prinzip bearbeitet werden<br><br>d) Funktionalität der Schlange unter Verwendung der Klasse Queue; Erschließen der Standardoperationen<br><br>e) Modellierung und Implementierung einer Anwendung auf der Basis einer Anforderungsbeschreibung mit Objekten der Klasse Queue | Die Schülerinnen und Schüler<br><br>• erläutern und begründen methodische Vorgehensweisen, Entwurfs- und Implementationsentscheidungen sowie Aussagen über Informatiksysteme (A)<br>• konstruieren zu kontextbezogenen Problemstellungen informatische Modelle (M)<br>• ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M)<br>• ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu (M),<br>• implementieren auf der Grundlage | *Beispiele:*<br><br>Warteschlange, Tastaturpuffer, |

| | | |
|---|---|---|
| 2. Die Datenstruktur Stapel<br><br>a) Erläuterung von Problemstellungen, die nach dem LIFO-Prinzip bearbeitet werden<br><br>b) Funktionalität der Klasse Stapel unter Verwendung der Klasse Stack<br><br>c) Modellierung und Implementierung einer Anwendung auf Basis einer Anforderungsbeschreibung mit Objekten der Klasse Queue<br><br>d) Modellierung und Implementierung einer Anwendung unter Verwendung verschiedener Datenstrukturen (Objekte der Klassen Queue, Stack und Array) | von Modellen oder Modellausschnitten Computerprogramme (I)<br>• testen und korrigieren Computerprogramme (I)<br>• interpretieren Fehlermeldungen und korrigieren den Quellcode (I)<br>• überführen gegebene textuelle und grafische Darstellungen informatischer Zusammenhänge in die jeweils andere Darstellungsform (D)<br>• stellen informatische Modelle und Abläufe in Texten, Tabellen, Diagrammen und Grafiken dar (D)<br>• stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D)<br>• modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten (M)<br>• ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu (M)<br>• dokumentieren Klassen (D)<br>• implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken (I)<br>• grafisch dar(M)<br>• | Beispiel:<br><br>UPN-Rechner |
| 3. Die Datenstruktur Liste<br><br>a) Analyse der Möglichkeiten bisheriger Datenstrukturen zwecks Bestimmung notwendiger Funktionalitäten für komplexere Anwendungen (Abgrenzung zu Stack/Queue, zusätzliche Fähigkeiten der Klasse List)<br><br>b) Erarbeitung der Funktionalität der Liste unter Verwendung der Klasse List<br><br>c) Modellierung und Implementierung einer Anwendung mit Objekten der Klasse List<br><br>d) Modellierung und Implementierung einer Anwendung unter Verwendung verschiedener Datenstrukturen (Stack, | | *Beispiel:*<br><br>Bücherliste, Mitarbeiterliste |

| | | |
|---|---|---|
| Queue, List) | | |
| 4.Übungen und Vertiefungen zur Verwendung linearer und dynamischer Datenstrukturen anhand weiterer Problemstellungen | | selbständige Erarbeitung eines umfangreicheren Projektes |

**Unterrichtsvorhaben Nr. 4**

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

**Leitfragen**
Wie können Daten mithilfe von Baumstrukturen verwaltet werden? Wie können mit binären Suchbäumen Inhalte sortiert verwaltet werden und welche Vor- und Nachteile bietet dies?

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

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

**Vorhabenbezogene Konkretisierung:**

Anhand des Anwendungskontextes Stammbaum werden zunächst der generelle Aufbau von Baumstrukturen und wichtige Grundbegriffe erarbeitet. Die Darstellung von Bäumen mit Knoten und Kanten wird eingeführt.

Anschließend rückt der Fokus auf die binären Bäume, deren rekursiver Aufbau für die Traversierung der Datenstruktur genutzt wird. Die Preorder-Traversierung wird verwendet, um einen gespeicherten Inhalt in einem Binärbaum zu finden (Tiefensuche).
Der Anwendungskontext Ahnenbaum wird mithilfe der Klasse BinaryTree (der Materialien für das Zentralabitur in NRW) modelliert und (ggf. in Teilen) implementiert. Dabei wird u. a. die Erzeugung eines Binärbaums mithilfe der beiden Konstruktoren der Klasse BinaryTree thematisiert.

Möchte man Daten geordnet speichern, so bietet sich die Struktur des binären Suchbaums an. An Beispielen wird zunächst das Prinzip des binären Suchbaums erarbeitet. Die Operationen des Suchens, Einfügens, Löschens und der sortierten Ausgabe werden thematisiert.

Um Daten in einem Anwendungskontext mithilfe eines binären Suchbaums verwalten zu können, müssen sie in eine Ordnung gebracht werden können, d. h. sie müssen vergleichbar sein. Diese Vorgabe wird mithilfe des Interfaces Item realisiert, das alle Klassen, dessen Objekte in einem Suchbaum verwaltet werden sollen, implementieren müssen. Auf diese Weise wird ein Anwendungskontext mithilfe der Klassen BinarySearchTree und Item modelliert und (ggf. in Teilen) implementiert.

**Zeitbedarf**: 18 Stunden

**Sequenzierung des Unterrichtsvorhabens:**

| **Unterrichtssequenzen** | **Zu entwickelnde Kompetenzen** | **Beispiele, Medien oder Materialien** |
|---|---|---|
| 1. Aufbau von Baumstrukturen und Grundbegriffe<br><br>a) Erarbeitung der Begriffe Wurzel, Knoten, Blatt, Kante, Grad eines Knotens und eines Baumes, Pfad, Tiefe, Ebene, Teilbaum<br><br>b) Aufbau und Darstellung von Baumstrukturen in verschiedenen Anwendungskontexten | Die Schülerinnen und Schüler<br><br>• stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau (D)<br><br>• erläutern Operationen dynamischer (linearer oder nicht-linearer) Datenstrukturen (A)<br><br>• analysieren und erläutern Algorithmen und Programme (A)<br><br>• stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar (D) | *Beispiele:*<br><br>Stammbaum |

| | | |
|---|---|---|
| 2. binäre Bäume<br><br>a) rekursiver Aufbau eines binären Baums<br><br>b) Traversierungen (pre-, in-, postorder)<br><br>c) Modellierung eines Binärbaums in einem Anwendungskontext mit Hilfe der Klasse BinaryTree (als Entwurfs- und Implementationsdiagramm)<br><br>d) Implementation einer Anwendung der Datenstruktur binärer Baum (ggf. in Teilen) | • beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen (A)<br><br>• ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehungen (M)<br><br>• ordnen Attributen, Parametern und Rückgaben von Methoden einfache Datentypen, Objekttypen sowie lineare und nichtlineare Datensammlungen zu (M)<br><br>• modellieren abstrakte und nicht abstrakte Klassen unter Verwendung von Vererbung durch Spezialisieren und Generalisieren (M)<br><br>• verwenden bei der Modellierung geeigneter | *Beispiel:*<br><br>Termbaum |

| | | |
|---|---|---|
| 3. Binäre Suchbäume<br><br>a) Prinzip des binären Suchbaums, Ordnungsrelation<br><br>b) Operationen auf dem binären Suchbaum (Suchen, Einfügen, Löschen, sortierte Ausgabe)<br><br>c) Modellierung eines binären Suchbaums in einem Anwendungskontext mit Hilfe der Klasse BinarySearchTree (als Entwurfs- und Implementationsdiagramm) und dem Interface Item<br><br>d) Implementation einer Anwendung der Datenstruktur binärer Suchbaum (ggf. in Teilen) | Problemstellungen die Möglichkeiten der Polymorphie (M)<br><br>• entwickeln iterative und rekursive Algorithmen unter Nutzung der Konstruktionsstrategien „Modularisierung“ und „Teilen und Herrschen“ (M)<br><br>• implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen (I)<br><br>• modifizieren Algorithmen und Programme (I)<br><br>• nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen (I)<br><br>• interpretieren Fehlermeldungen und korrigieren den Quellcode (I)<br><br>• testen Programme systematisch anhand von Beispielen (I) | *Beispiel:*<br><br>Schlagwortverzeichnis |
| 4. Übung und Vertiefungen der Verwendung von Binärbäumen oder binären Suchbäumen anhand weiterer Problemstellungen | | *Beispiel:*<br><br>Stichwortverzeichnis; Städteraten |