Schulinterner Lehrplan des Städtischen Bertha-von-Suttner-Gymnasiums, Oberhausen, zum Kernlehrplan für die Sekundarstufe II

INFORMATIK – Qualifikationsphase 1 und 2

Stand: Juli 2019

Inhaltsverzeichnis

1. Die Fachschaft Informatik des Bertha-von-Suttner-Gymnasiums

Die Fachschaft Informatik besteht im Schuljahr 2019 / 2020 aus zwei Lehrern mit der Fakultas Informatik Sek. I bzw. II. Eine zukünftige dritte Lehrkraft nimmt an einem Zertifikatskurs Informatik für Oberstufe teil: Es gibt im Bereich der Einführungsphase grundsätzlich mehr Interessenten als freie Kursplätze.

In der Sek. I wird aufgrund personeller Engpässe derzeit zum Teil fachfremder Unterricht erteilt. Hinzu kommt das Wahlverhalten der Schülerinnen und Schüler:

Ein Fachvorsitzender ist für die organisatorischen Belange des Faches in der Schule zuständig, die von der Fachgruppe unterstützt wird. Unter den Fachkollegen findet regelmäßig ein Austausch über Fachinhalte und -methoden statt.

2. Der Informatikunterricht in der Einführungsphase

2.1 Unterrichtsstruktur im Fach Informatik

Für Schülerinnen und Schüler des Bertha-von-Suttner-Gymnasiums ist Informatikunterricht ein reines Wahlfach / Wahlpflichtfach. Es wird in folgenden Jahrgangsstufen angeboten:

- Jgst. 6 (2 Stunden wöchentlich im Rahmen des MINT-Zweiges)
- Jgst. 8 / 9 (2 Stunden wöchentlich im Rahmen des Wahlpflichtangebotes)
- Jgst. 10 (EF) (3 Stunden wöchentlich als Wahlfach. Bei Engpässen in der Unterrichtsverteilung wird der Kurs zwei stündig unterrichtet.)
- Jgst. 11 (Q1) (3 Stunden wöchentlich als Wahlfach – Voraussetzung ist ein erfolgter Unterricht in der Einführungsphase)
- Jgst. 12 (Q2) (3 Stunden wöchentlich als Wahlfach – Voraussetzung ist ein erfolgter Unterricht in der Q1)

2.2 Außerunterrichtliche Angebote

Alle Schülerinnen und Schüler haben die Möglichkeit an außerunterrichtlichen Aktivitäten des Faches Informatik teilzunehmen. Diese umfasst eine (je nach Situation / vorhandenem Interesse der Schülerinnen und Schüler) stattfindende Informatik AG, sowie in unregelmäßigen Abständen angebotene Workshops (z.B. App-Programmierung für Handys, usw.).

2.3 Fachräume und Arbeitsplätze

Für den Informatikunterricht stehen die Fachräume 305 und 306 zur Verfügung.

Raum 306 ist als ein reiner Fachraum zu verstehen, der ausschließlich durch Informatikkurse genutzt wird. Die Ausstattung umfasst 15 Arbeitsplätze (Rechnergeneration 2013) für Schüler,

sowie ein Lehrer-Arbeitsplatz mit angeschlossenem Deckenbeamer. Vorzugsweise findet der Informatikunterricht der Oberstufe in Raum 306 statt.
Raum 305 ist sowohl Unterrichtsraum für das Fach Informatik, als auch von allen anderen Lehrerinnen und Lehrern anderer Unterrichtsfächer bei Bedarf nutzbar (Internetrecherchen, Nutzung von Fachsoftware in den Unterrichtsfächern Mathematik, Musik, usw.). Eine entsprechende Liste der Belegungszeiten hängt zum Eintragen im Lehrerzimmer aus. Die Ausstattung in Raum 305 umfasst 15 Arbeitsplätze (Rechnergeneration 2015) für Schüler, sowie ein Lehrer-Arbeitsplatz mit angeschlossenem Deckenbeamer. Durch die Nutzung unterschiedlichster Lerngruppen ist insbesondere in Raum 305 ein sorgsamer Umgang erforderlich.

2.4 Vernetzung, Anmeldeprofil und Datenspeicherung

Die Arbeitsplätze in den Räumen 305 und 306 sind keine autarken Einzelarbeitsplätze sondern sind in einem pädagogischen Netzwerk zusammengeschlossen. Schülerinnen und Schüler der Informatikkurse bzw. Lehrer der Fachgruppe Informatik verfügen über ein eigenes Nutzerprofil, mit dem Sie sich unabhängig vom physikalischen Arbeitsplatz anmelden können.
Alle anderen Lehrerinnen und Lehrer, bzw. Schülerinnen und Schüler nutzen das Profil mit dem Benutzernamen „sbvs“ (ohne Passwort). Dieses Profil hat kein serverangebundenes Laufwerk, ermöglicht aber den Zugang zum Internet, die Nutzung der installierten Software, usw.. Die Sicherung von Daten hat in diesem Fall auf USB-Stick zu erfolgen. Für sämtliche durch Lehrer oder Schüler lokal oder auf dem Server abgelegten Daten wird keinerlei Haftung übernommen. Zu beachten ist außerdem, dass das Laufwerk C jeder Arbeitsstation Daten nicht dauerhaft speichert. Daten gehen nach dem Herunterfahren verloren. Das Laufwerk D jeder Arbeitsstation wird am Ende jeden Schuljahres ungefragt bereinigt und von dem sich angesammelten Datenmaterial befreit.

2.5 (Server–)Administration

Der Server befindet sich in der hinteren Kammer von Raum 306. Um die Administration und technische Funktionsfähigkeit des Servers (bzw. auch der Arbeitsplätze) kümmert sich ein externer Dienstleister. Das Einrichten und Verwalten von Nutzerprofilen, bzw. auch das Installieren von neuer Software regelt die Fachgruppe Informatik im Rahmen der ihrer zeitlichen und organisatorischen Möglichkeiten selbstständig, bzw. beauftragt dazu den entsprechenden Dienstleister (OGM bzw. VG-Systems).

3. Schulinterne Unterrichtsvorhaben in der Qualifikationsphase

3.1 Voraussetzungen und Rahmenbedingungen in der Qualifikationsphase

Grundsätzliche Informationen zum Unterrichtsfach Informatik und den entsprechenden Rahmenbedingungen am Bertha-von-Suttner-Gymnasium Oberhausen sind dem schulinternen Lehrplan der Einführungsphase zu entnehmen.

Die Voraussetzungen und Rahmenbedingungen, welche sich auf die Qualifikationsphase beziehen sind hier in Kurzform auf Basis des schulinternen Curriculums zur Einführungsphase nochmals zusammengefasst bzw. um Aspekte, die sich ausschließlich auf die Qualifikationsphase beziehen konkretisiert:

- Der Informatikunterricht in der Oberstufe ist aufbauend: Vorraussetzung zur Teilnahme am Informatikunterricht der Qualifikationsphase 1 ist ein erfolgter Informatikunterricht im Rahmen der Einführungsphase, bzw. im Rahmen der Qualifikationsphase 2 ein erfolgter Unterricht der Qualifikationsphase 1.
- Das Kerncurriculum Informatik SII macht keine Angaben über eine zu benutzende Programmiersprache. Da sich die Vorgaben des Zentralabiturs für das Abitur ausschließlich auf die Programmiersprache Java beziehen, wird im Informatikunterricht der Qualifikationsphase 1 und 2 des Bertha-von-Suttner-Gymnasiums weiterhin vorzugsweise die Programmiersprache Java verwendet.
- Im Kontext von Java kommen solche Bibliotheken zum Einsatz, die im Zuge der Vorgaben des Zentralabiturs für die Qualifikationsphase angegeben werden und damit verpflichtend sind. Im Regelfall sind dies im Rahmen eines Grundkurses die Bibliotheken zu dynamischen Datenstrukturen (Stack, Queue, List, BinaryTree, BinarySearchTree, ComparableContent, DatabaseConnector, Queryresult).[1]
- Eventuelle weitere (didaktische) Software, die neben Java im Kontext von bestimmten Unterrichtsvorhaben zusätzlich zum Einsatz kommt, ist optional und kann bei Bedarf entsprechend angepasst bzw. ausgetauscht werden.

[1] Vgl. jeweils aktuelle Abiturvorgaben auf der Homepage des Bildungsportals NRW.

3.2 Inhaltsfelder und Kompetenzbereiche

Die im Plan der Qualifikationsphase benutzte Symbolik zu den Inhaltsfeldern und Kompetenzbereichen ist analog zur Einführungsphase bzw. analog zum offiziellen Kernlehrplan Informatik:

Inhaltsfelder

Daten und ihre Strukturierung

Algorithmen


Formale Sprachen und Automaten


Informatiksysteme


Informatik, Mensch und Gesellschaft

Kompetenzbereiche


Argumentieren


Modellieren


Implementieren


Darstellen und Interpretieren


Kommunizieren und Kooperieren[2]

[2] Der Kompetenzbereich *Kommunizieren und Kooperieren* nimmt insofern eine Sonderstellung ein, da er zugehörig zu allen Unterrichtsvorhaben der Qualifikationsphase ist. Aus Gründen der Lesbarkeit wird dieser Kompetenzbereich innerhalb dieses schulinternen Curriculums nicht in jedem Unterrichtsvorhaben separat aufgeführt. D.h. in jedem Unterrichtsvorhaben gilt mit unterschiedlichem Schwerpunkt, dass Schülerinnen und Schüler

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

3.3 Unterrichtsvorhaben in der Qualifikationsphase 1

3.3.1 Übersichtsraster der Unterrichtsvorhaben in der Qualifikationsphase 1

Nachfolgend sind Übersichtsraster der Unterrichtsvorhaben und die Konkretisierung der Unterrichtsvorhaben für die Qualifikationsphase 1 aufgeführt (die Bezeichnungen der Unterrichtsvorhaben sind keine Themenformulierungen). Die jeweiligen Zeitangaben zu den Unterrichtsvorhaben verstehen sich als Orientierungsgrößen, die nach Bedarf über- oder unterschritten werden können (aufgrund von Vertiefungen von Inhalten, Schülerinteresse, aktuelle Themen, entfallender Unterricht durch organisatorische Veranstaltungen der Jahrgangsstufe, Kursfahrten, usw.). Insgesamt wird im schulinternen Lehrplan der theoretische Umfang von 120 Unterrichtsstunden also bewusst nicht ausgereizt.
Zum Teil haben die Unterrichtsvorhaben inhaltliche Voraussetzungen und bauen aufeinander auf. Das bedeutet nicht, dass eine durchgehend feste Reihenfolge der Unterrichtsvorhaben in der Durchführung eingehalten werden muss, es müssen lediglich die Abhängigkeiten berücksichtigt werden (Ausnahme bildet die Wiederholungsphase des Unterrichtsvorhaben 1).

Unterrichtsvorhaben	Zeitbedarf	Voraussetzungen? / Wann?
1: *Wiederholung objektorientierter Modellierung/Programmierung und algorithmischer Abläufe*	6 Stunden	Zu Beginn der Qualifikationsphase 1
2: *Lineare dynamische Datenstrukturen: Listen* A: *Zur Notwendigkeit dynamischer linearer Datenstrukturen (hier Listen) und deren Grundlagen* B: *Entwicklung einer eigenen Basisklasse zu Listknoten bzw. Liste* C: *Analyse und Basisnutzung der vorgegebene Klasse „list“, bzw. „stack“ und „queue“* D: L*ineare dynamische Datenstrukturen im Kontext konkreter implementierter Anwendungen: Projektphase / Anwendung*	 6 Stunden 14 Stunden 10 Stunden 17 Stunden	Keine Voraussetzung
3: *Nutzung und Modellierung von relationalen Datenbanken in Anwendungskontexten*	30 Stunden	Keine Voraussetzung
4: *Rekursion als alternativer Lösungsansatz im Kontext selbstbezüglicher Problemstellungen*	15 Stunden	Keine Voraussetzung
5: *Modellierung und Implementierung von Anwendungen mit dynamischen, nichtlinearen Datenstrukturen: binäre Bäume*	18 Stunden	Unterrichtsvorhaben 2 + 4

3.3.2 Konkretisierte Unterrichtsvorhaben in der Qualifikationsphase 1

Unterrichtsvorhaben 1: ***Wiederholung objektorientierter Modellierung/Programmierung und algorithmischer Abläufe***					
Inhaltsfelder:	 **Daten und ihre Strukturierung**	 Formale Sprachen und Automaten	 Informatik, Mensch und Gesellschaft	 Informatiksysteme	 **Algorithmen**
Zu entwickelnde Kompetenzen:	Die Schülerinnen und Schüler…				
 **Darstellen und Interpretieren**	…stellen Klassen, Assoziationen- und Vererbungsbeziehungen in UML-Diagrammen grafisch dar.*				
 **Modellieren**	…modellieren Klassen mit ihren Attributen, Methoden und Assoziationsbeziehungen. …ordnen Klassen, Attributen und Methoden ihre Sichtbarkeiten zu. …ordnen Attributen, Parametern und Rückgaben von Methoden einfachen Datentypen zu. …entwerfen einfache Algorithmen und stellen sie umgangssprachlich und grafisch dar.				
 Argumentieren					
 **Implementieren**	…implementieren Klassen in einer Programmiersprache (auch unter Nutzung dokumentierter Klassenbibliotheken). …interpretieren Fehlermeldungen und korrigieren Quellcode. …implementieren Algorithmen unter Verwendung von Variablen und Wertzuweisungen, Kontrollstrukturen sowie Methodenaufrufen. …testen Programme schrittweise anhand von Beispielen. …modifizieren einfache Algorithmen und Programme.				

* Die konkretisierten Kompetenzerwartungen sind dem Kernlehrplan entnommen. Sie stellen den jeweiligen kompetenzbezogenen Kern des Unterrichtsvorhabens dar. Darüber hinaus sind weitere konkretisierte Kompetenzerwartungen im Sinne des Kernlehrplans möglich. Gleiches gilt für die nachfolgenden Unterrichtsvorhaben.

Unterrichtsvorhaben 1: *Wiederholung objektorientierter Modellierung/Programmierung und algorithmischer Abläufe*

Kompetenzerwartung / Beschreibung / Beispiele:

Im Mittelpunkt der Wiederholungsphase steht die Reaktivierung grundlegender Kompetenzbereiche, im Kontext der Inhaltsfelder *Daten und ihre Strukturierung*, sowie *Algorithmen*. Die entsprechenden Grundlagen bilden eine wichtige Voraussetzung für das erfolgreiche Durchlaufen der Qualifikationsphase 1 und sind (prinzipiell) durch die Einführungsphase vorauszusetzen.

- Die SuS wiederholen an unterschiedlichen (Kurz-)Beispielen Grundlagen der objektorientierten Modellierung.

- Vorschlag: Die SuS bilden Expertenpaare für folgende (zu wiederholende) (Java-)Inhalte / Kompetenzen: Grundsätzlicher Aufbau einer Klasse, Sichtbarkeiten, Realisierung von Assoziationen zwischen Klassen, Vererbung, Gebrauch von Methoden, Ansteuern der Konsole, primitive und nicht primitive Datentypen, Variablen, bedingte Anweisungen, Schleife, etc. Die Expertenteams fassen ihr Wissen in Kurzform auf Papier mit einfachen Beispielen zusammen, sodass aus dem zusammengefassten Wissen ein übersichtliches kurzes Grundlagenwerk entsteht.

- Die SuS trainieren ihre Kompetenzen bei Implementatiosaufgaben und greifen dabei – sofern notwendig – auf das Grundlagenwerk zurück.

Anmerkung:

- Die in der Einführungsphase im Kontext von Java benutzte (didaktische) Bibliothek GLOOP spielt im Rahmen der Wiederholungsphase und auch im Rahmen der gesamten Qualifikationsphase 1 keine Rolle.
- Die Bedeutung und der Nutzen von Heftführung im Rahmen der Einführungsphase soll durch das oben angestrebte „Grundlagenwerk“ nicht unterwandert werden. Es geht hier schlicht um eine Art lexikonartige Kurzzusammenfassung, ausschließlich mit dem Akzent zur Hilfestellung bei Implementationsphasen.

Unterrichtsvorhaben 2: ***Lineare dynamische Datenstrukturen: Listen*** ***A: Zur Notwendigkeit dynamischer linearer Datenstrukturen (hier Listen) und deren Grundlagen***					
Inhaltsfelder:	**Daten und ihre Strukturierung**	Formale Sprachen und Automaten	Informatik, Mensch und Gesellschaft	Informatiksysteme	**Algorithmen**
Zu entwickelnde Kompetenzen: **Darstellen und Interpretieren** **Modellieren** **Argumentieren** **Implementieren**	Die Schülerinnen und Schüler… …stellen lineare Strukturen grafisch dar und erläutern ihren Aufbau. …stellen iterative Algorithmen umgangssprachlich und grafisch dar. …ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehung. …stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar. …beurteilen die Effizienz von Algorithmen unter Berücksichtigung des Speicherbedarfs und der Zahl der Operationen …erläutern Operationen dynamischer (hier: *linearer*) Datenstrukturen. …analysieren und erläutern Algorithmen und Programme. …nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und Analyse von Programmen. ...implementieren Klassen in einer Programmiersprache. …interpretieren Fehlermeldungen und korrigieren den Quellcode. …modifizieren Algorithmen und Programme.				

Unterrichtsvorhaben 2: ***Lineare dynamische Datenstrukturen: Listen***
A: Zur Notwendigkeit dynamischer linearer Datenstrukturen (hier Listen) und deren Grundlagen

Kompetenzerwartung / Beschreibung / Beispiele:

- Die SuS analysieren Problemsituationen die im Kontext zu Arrays (Vgl. Arrays im Rahmen der Einführungsphase) auftreten können, bzw. werden mit konkreten Problemsituationen konfrontiert. Ein „Problemkatalog“ zu statischen Datenstrukturen wird erstellt: Ein Array hat eine konstante Größe, es kann nicht wachsen um ggf. weitere Datensätze aufzunehmen bzw. bei nicht vollständiger Auslastung wird Speicherplatz verschenkt. Das Verschieben von mehreren Datensätzen ist nicht ohne Weiteres möglich, etc.

- Die SuS greifen den Problemkatalog auf und entwickeln auf Basis der Anforderungen ein entsprechendes theoretisches dynamisches Alternativmodell. Dessen zentrales Element ist ein Knoten, der genau einen Knoten als Nachfolger haben kann. Das Knotenelement bildet die Grundlage für die dynamische Datenstruktur Liste.

Unterrichtsvorhaben 2: ***Lineare dynamische Datenstrukturen: Listen*** **B: Entwicklung einer eigenen Basisklasse zu Listknoten bzw. Liste**					
Inhaltsfelder:	 **Daten und ihre Strukturierung**	 Formale Sprachen und Automaten	 Informatik, Mensch und Gesellschaft	 Informatiksysteme	 **Algorithmen**

Zu entwickelnde Kompetenzen:	Die Schülerinnen und Schüler…
 **Darstellen und Interpretieren**	…stellen lineare Strukturen grafisch dar und erläutern ihren Aufbau. …stellen iterative Algorithmen umgangssprachlich und grafisch dar.
 **Modellieren**	…ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehung. …stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar.
 **Argumentieren**	…beurteilen die Effizienz von Algorithmen unter Berücksichtigung des Speicherbedarfs und der Zahl der Operationen …erläutern Operationen dynamischer (hier: *linearer*) Datenstrukturen. …analysieren und erläutern Algorithmen und Programme.
 **Implementieren**	…nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und Analyse von Programmen. ...implementieren Klassen in einer Programmiersprache. …interpretieren Fehlermeldungen und korrigieren den Quellcode. …modifizieren Algorithmen und Programme

Unterrichtsvorhaben 2: *Lineare dynamische Datenstrukturen: Listen*
B: Entwicklung einer eigenen Basisklasse zu Listknoten bzw. Liste

Kompetenzerwartung / Beschreibung / Beispiele:

- Die SuS implementieren das theoretische Modell des Knotens inklusive Basismethoden als eine Klasse *ListNode* mit Basismethoden (*setNext*, *getNext*, *setContent*, *getContent*). Um das Prinzip einer Liste zu fokussieren, wird ausschließlich mit einem konkretisierten Datentyp als Inhalt gearbeitet (z.B. integer), um auf die „Neben–Baustelle“ generische Klasse ausdrücklich zunächst zu verzichten (Vgl. Sequenz C dieser Unterrichtsreihe). Im Kontext von BlueJ kann anschließend das Erstellen, Beschreiben, Auslesen und Verketten von Knoten erprobt werden.

- Die SuS entwickeln und implementieren auf Basis ihrer Klasse *ListNode* eine eigene Klasse *List* um listentypische Abläufe, die zuvor in BlueJ „per Hand“ ausgeführt wurden zu automatisieren. Im Fokus stehen hierbei:
  - die Notwendigkeit der Zeiger *first*, *current* und *last*.
  - die Implementation von Basismethoden wie: neue Liste erstellen (*Konstruktor*), Knoten mit Inhalt anfügen (*append*), in der Liste an den Anfang springen (*toFirst*), in der Liste zum nächsten Element gehen (*next*), Inhalt des aktuellen Knotens auslesen (*getContent*). Je nach Leistungsstand können im Sinne von Binnendifferenzierung auch weitere typische Methoden selbstständig implementiert werden: *remove*, *insert*, usw..

Didaktischer Kommentar:

1. Der Kernlehrplan Informatik gibt eine fertige Klasse *List* vor. Die Sinnhaftigkeit, eine solche Klasse wie hier beschrieben in Ansätzen selber zu implementieren – obwohl es sie schon gibt – besteht darin, dass Schülerinnen und Schüler auf diese Art und Weise auf einem anderen Niveau die Funktionsprinzipien besser durchschauen und vertiefen. Die Sequenz B dieser Unterrichtsreihe zielt damit auf Entwicklung und Verständnis ab, weniger auf eine konkrete Benutzung. (Die Entwicklung einer solchen eigenen Klasse ist im Kontext der dynamischen Datenstrukturen im Rahmen des schulinternen Lehrplans exemplarisch nur hier vorgesehen. Im Zusammenhang von *Queue*, *Stack*, *BinaryTree*, usw. werden die fertigen Klassen benutzt, ggf. analysiert.)
2. Neben der linearen Datenstruktur einer Liste sind durch den Kernlehrplan außerdem noch die linearen Datenstrukturen Schlange (*Queue*) und Stapel (*Stack*) vorgegeben. Obwohl die Datenstruktur einer Liste im Vergleich zu den anderen Datenstrukturen komplexer ist, macht es Sinn mit Listen anzufangen: Das Ausgangsproblem bei der Benutzung von Arrays führt unmittelbar zur Struktur einer Liste, nicht zu einer Schlange oder einem Stapel. Die Datenstrukturen zu Schlange und Stapel sind Teil der Sequenz C dieses Unterrichtvorhabens.

Unterrichtsvorhaben 2: ***Lineare dynamische Datenstrukturen: Listen*** ***C: Analyse und Basisnutzung der vorgegebene Klasse „list“ bzw. „stack“ und „queue“***					
Inhaltsfelder:	**Daten und ihre Strukturierung**	Formale Sprachen und Automaten	Informatik, Mensch und Gesellschaft	Informatiksysteme	**Algorithmen**

Zu entwickelnde Kompetenzen:	Die Schülerinnen und Schüler…
Darstellen und Interpretieren	…stellen lineare Strukturen grafisch dar und erläutern ihren Aufbau. …stellen iterative Algorithmen umgangssprachlich und grafisch dar.
Modellieren	
Argumentieren	…analysieren und erläutern objektorientierte Modellierungen. …analysieren und erläutern Algorithmen und Programme. …erläutern Operationen dynamischer (hier: *linearer*) Datenstrukturen.
Implementieren	…testen Programme systematisch anhand von Beispielen

Unterrichtsvorhaben 2: ***Lineare dynamische Datenstrukturen: Listen***
C: ***Analyse und Basisnutzung der vorgegebene Klasse „list“ bzw. „stack“ und „queue“***

Kompetenzerwartung / Beschreibung / Beispiele:

- Die SuS setzen sich mit dem Quelltext der Klasse List durch die Vorgaben des Zentralabiturs in der jeweils aktuellen Version auseinander. Sie trainieren dabei, sich in komplexen Quelltexten zunächst eine Übersicht zu verschaffen und einen Quelltext „quer“ zu lesen.

- Die SuS vergleichen den vorgegebenen Quelltext mit ihrem eigenen Quelltext aus Sequenz B. Sie verdeutlichen Gemeinsamkeiten, Unterschiede und benennen das durch die Zentralabiturvorgaben erweiterte Methodenrepertoire. Exemplarisch wird das Funktionsprinzip einzelner (bisher unbekannter) Methoden analysiert.

- Die SuS erproben die Nutzung der vorgegebenen Klasse im Kontext eines simplen minimalistischen kontextlosen Beispiels. Im Zusammenhang der Klasse List als generische Klasse steht hierbei die Notwendigkeit von Typecasts im Mittelpunkt.

- Die SuS erweitern ihr Spektrum über lineare dynamische Datenstrukturen um Stapel und Schlange, verdeutlichen sich dabei die Prinzipien LiFo bzw. FiFo und arbeiten mit den entsprechenden vorgegebenen Klassen im Kontext einfacher Beispiele.

Didaktischer Hinweis:
Die vorgegebene Klasse *List* ist eine generische Klasse. Generische Klassen sind durch den Kernlehrplan im expliziten Sinne nicht vorgesehen, das Konzept wird aber indirekt durch die vorgegebene Klasse *List* dennoch benutzt. Die Sequenz C dieser Unterrichtsreihe sollte folglich einen entsprechenden (kurzen) Exkurs zu generischen Klassen enthalten, nicht nur um die Klasse *List* zu verstehen, sondern auch um diese überhaupt nutzen zu können. (Selbiges gilt auch für das Konzept von inneren und äußeren Klassen, welches im Rahmen der vorgegebenen Klasse *List* ebenfalls zum Einsatz kommt.)

Unterrichtsvorhaben 2: ***Lineare dynamische Datenstrukturen: Listen*** ***D: lineare dynamische Datenstrukturen im Kontext konkreter implementierter Anwendungen: Projektphase / Anwendung***					
Inhaltsfelder:	**Daten und ihre Strukturierung**	Formale Sprachen und Automaten	Informatik, Mensch und Gesellschaft	Informatiksysteme	**Algorithmen**

Zu entwickelnde Kompetenzen:	Die Schülerinnen und Schüler… *(die sich entwickelnden Kompetenzen sind in ihrer Gewichtung unter Umständen stark vom jeweiligen Projekt abhängig)*
 **Darstellen und Interpretieren**	…stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar. …dokumentieren Klassen. …stellen iterative Algorithmen umgangssprachlich und grafisch dar. …nutzen bereitgestellte Informatiksysteme und das Internet reflektiert zur Erschließung fachlicher Inhalte.
 **Modellieren**	…stellen Klassen und ihre Beziehungen in Diagrammen grafisch dar. …ermitteln bei der Analyse von Problemstellungen Objekte, ihre Eigenschaften, ihre Operationen und ihre Beziehung. …modellieren Klassen mit ihren Attributen, Methoden und ihren Assoziationsbeziehungen unter Angabe von Multiplizitäten. …ordnen Klassen, Attributen und Methoden ihre Sichtbarkeitsbereiche zu.
 **Argumentieren**	…analysieren und erläutern objektorientierte Modellierungen. …analysieren und erläutern Algorithmen und Programme.
 **Implementieren**	…nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und Analyse von Programmen. ...implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken. …interpretieren Fehlermeldungen und korrigieren den Quellcode. …modifizieren Algorithmen und Programme.

Unterrichtsvorhaben 2: ***Lineare dynamische Datenstrukturen: Listen***
D: Listen im Kontext konkreter implementierter Anwendungen: Projektphase / Anwendung

Kompetenzerwartung / Beschreibung / Beispiele:

Die Unterrichtssequenz D dieses Unterrichtvorhabens bringt keine neuen Kenntnisse, die mit Blick auf die Vorgaben durch den Kernlehrplan von Bedeutung wären. Die Sequenz dient dazu, vorhandene Kompetenzen bei den Schülerinnen und Schülern zu stärken. Je nach zur Verfügung stehender Zeit ergeben sich unterschiedliche Möglichkeiten:

Projektphase (hoher zeitlicher Umfang):

- Die SuS werden die Basiskompetenzen zum Erstellen einer einfachen grafischen Oberfläche vermittelt (durch den Kernlehrplan nicht explizit vorgegeben).

- Die SuS wenden ihre Kompetenzen im Umgang mit der vorgegebenen Klasse List im Kontext einer grafischen Nutzeroberfläche an und realisieren im Rahmen einer Projektphase in Partnerarbeit eine eigene fiktive Software für einen Endnutzer. Innerhalb dieser Software muss es um ein Problemkontext gehen, bei dem das Konzept Liste als Lösungsmittel im Fokus steht. (Alternativ Stack oder Queue, oder die Kombination von mehreren linearen dynamischen Datenstrukturen)
  (Beispiel: Einkaufsliste: Einzelne Produkte (=Knoten) können hinzugefügt und entfernt werden. Im Sinne einer dynamischen Datenstruktur kann die Einkaufsliste beliebig wachsen und wieder schrumpfen. Im Sinne einer Binnendifferenzierung können die Projekte auch nach belieben komplexer gestaltet werden, etwa durch Listen in Listen: Eine Schule hat in einer bestimmten Jahrgangsstufe mehrere Kurse die im Rahmen einer Liste aufgeführt sind. In jedem Knoten dieser Liste (=jeder Knoten repräsentiert einen Kurs) gibt es wiederum eine Liste, die mit ihren Knoten auf Instanzen von Schülern verweist.

- Die SuS planen den Ablauf des Entwicklungsprozesses. Sie legen realistische Ziele fest und modellieren das Vorhaben als Modell zunächst in UML bevor sie mit der eigentlichen Implementation beginnen.

- Die SuS erstellen einen Katalog mit Bewertungskriterien. Es besteht auch die Möglichkeit einer Ergebnispräsentation, oder einer gegenseitigen Analyse / Testphase der Projekte, etc.

Anwendung auf Basis von Vorgaben (geringer zeitlicher Umfang):

- Alternativ zur Projektphase besteht ebenso die Möglichkeit, eine Anwendungsphase durch eine einheitliche Aufgabenstellung für alle Schüler zeitlich stark zu straffen. Auf eine grafische Oberfläche (durch den Kernlehrplan nicht vorgesehen) kann dabei ebenso verzichtet werden.

Unterrichtsvorhaben 3: ***Nutzung und Modellierung von relationalen Datenbanken in Anwendungskontexten***					
Inhaltsfelder:	**Daten und ihre Strukturierung**	Formale Sprachen und Automaten	Informatik, Mensch und Gesellschaft	Informatiksysteme	**Algorithmen**

Zu entwickelnde Kompetenzen:	Die Schülerinnen und Schüler…
Darstellen und Interpretieren	…überprüfen Datenbankschemata auf vorgegebene Normalisierungseigenschaften. …ermitteln Ergebnisse von Datenbankabfragen über (mehrere verknüpfte) Tabellen.
Modellieren	…ermitteln für anwendungsbezogene Problemstellungen Entitäten, zugehörige Attribute, Relationen und Kardinalitäten. …modifizieren eine Datenbankmodellierung. …modellieren zu einem ER-Diagramm ein relationales Datenbankschema. …bestimmen Primär- und Sekundärschlüssel. …überführen Datenbankschemata in die 1. bis 3. Normalform.
Argumentieren	…analysieren und erläutern die Syntax und Semantik einer Datenbankabfrage. …analysieren und erläutern eine Datenbankmodellierung. …erläutern die Eigenschaften normalisierter Datenbankschemata. …erläutern Eigenschaften, Funktionsweise und Aufbau von Datenbanksystemen unter dem Aspekt sicherer Nutzung.
Implementieren	…verwenden die Syntax und Semantik einer Datenbankabfragesprache, um Informationen aus einem Datenbanksystem zu extrahieren. *…implementieren auf der Grundlage von Modellen oder Modellausschnitten Computerprogramme / modifizieren und erweitern Computerprogramme / testen und korrigieren Computerprogramme systematisch [hier im Kontext der Anbindung von Datenbanken an Java seit den Abiturvorgaben für das Jahr 2020 . Der Kernlehrplan gibt hierzu keine konkretisierten Kompetenzerwartungen an.]*

Unterrichtsvorhaben 3: *Nutzung und Modellierung von relationalen Datenbanken in Anwendungskontexten*

Kompetenzerwartung / Beschreibung / Beispiele:

- Die SuS sammeln ihr Wissen über die Begrifflichkeit „Datenbank“ und konkretisieren die möglicherweise schwammig umrissene Vorstellung auf Basis von Recherchen bzw. Fachtexten. Sie erörtern anschließend, in welchen Zusammenhängen sie täglich mit Datenbanken in Berührung kommen (z.B. Webshops, usw.) und fundieren damit einen Lebensweltbezug zur Unterrichtsreihe.

- Die SuS beschäftigen sich zunächst mit einzelnen nicht zu verknüpfenden vorgegebenen Tabellen und verstehen diese als Relation, bestehend aus Attributen und Datensätzen, welche mit Hilfe unterschiedlicher Basis-Operationen der relationalen Algebra „am Papier“ abgefragt werden kann (Selektion, Projektion, Umbenennung). Analog dazu werden im Anschluss die entsprechenden SQL-Statements hinzugezogen. Die SuS erweitern ihr Kompetenzspektrum um Aggregatfunktionen und Bedeutung eines Primärschlüssels. Software und Beispieldatenbanken (siehe Materialhinweis) dienen dazu, die gängige Informatikpraxis im Kontext von Datenbanken und SQL am Computer zu erproben.

- Die SuS benennen die Notwendigkeit mehrere Tabellen miteinander zu verknüpfen, um bestimmte Abfragen an eine Datenbank zu realisieren. In diesem Kontext arbeiten sie sowohl auf Ebene der relationalen Algebra als auch in SQL mit Kreuzprodukt bzw. Selektion und den sich daraus ergebenden unterschiedlichen Join-Varianten. Vermischte Übungen, auch mit verschachtelten SQL-Abfragen, dienen dazu, die entsprechenden Kompetenzen zu trainieren.

- Die SuS planen eine eigene fiktive Datenbank bzw. werden damit konfrontiert, eine Datenbank für einen bestimmten vorgegebenen Anwendungszweck zu realisieren (z.B. Patientenverwaltung im Krankenhaus mit bestimmten Anforderungen, etc.). Unter Rückgriff auf das Entity-Relationship-Model als Planwerkzeug stehen dabei insbesondere die Umsetzung unterschiedlicher Kardinalitätsformen (1:1, 1:n, n:m) bei der Beziehung zwischen mehreren Relationen im Mittelpunkt. Die sich daraus ergebenen Relationsschemata können mittel *SQLite* (siehe Materialhinweis) umgesetzt und rudimentär mit Beispieldatensätzen gefüllt werden, um die Funktionalität zu erproben.

- Die SuS beschäftigen sich mit den Auswirkungen von Redundanz im Kontext einer Datenbank und dem daraus entstehenden Risiko eines inkonsistenten Datenbestands. Als Lösungskonzept dient die 1. bis 3. Normalform mit entsprechenden Übungen. (Die im Materialhinweis benannten Beispieldatenbanken befinden sich bereits in entsprechenden Normalformen. An dieser Stelle muss ein anderes Szenario hinzugezogen werden, etwa eine fiktive „ungeschickt“ modellierte Datenbank zur Verwaltung einer Musik-Sammlung, die Datenbank einer Buchhandlung, etc. Entscheidend ist dabei, dass die Datenbank dazu verleitet Informationen mehrfach abzulegen, um die Problematik der Redundanz in den Fokus zu rücken.)

- Die SuS nutzen die durch die Kernrichtlinien vorgegebene Klasse *DatabaseConnector* und *QueryResult* und realisieren hiermit Datenbankzugriffe über die Programmiersprache Java.

Materialhinweis:

- **Literatur:** T. Kempe / A. Löhr (Hrsg.): Informatik. Band 3., Schöningh 2013.
- **Software:** Für praktische Übungen mit SQL hat sich als besonders tauglich und einfach die Software *SQLite* erwiesen (Freeware). Eine andere Software ist ebenso denkbar. Eine Einbettung von SQL in Java ist ebenfalls möglich, würde aber den Schwerpunkt des Unterrichtsvorhabens verzerren. Als mögliche Beispiel für eine konkrete Datenbank wird an dieser Stelle auf *VideoCenter* (Videothek) verwiesen (=Vorschlag der Qualitäts- und Unterstützungsagentur Landesinstitut für Schule NRW), oder aber auch auf die vom Verlag Terra angebotene *geographische Unterrichtsdatenbank*, etc.)

Didaktischer Hinweis:

Im Rahmen von SQL werden hier keine Sprachelemente im Sinne einer Vollständigkeit benannt. Die genauen verpflichtenden Operatoren etc. sind den jeweiligen Vorgaben und Dokumentationen des Zentralabiturs zu entnehmen.

Unterrichtsvorhaben 4: ***Rekursion als alternativer Lösungsansatz im Kontext selbstbezüglicher Problemstellungen***					
Inhaltsfelder:	 Daten und ihre Strukturierung	 Formale Sprachen und Automaten	 Informatik, Mensch und Gesellschaft	 Informatiksysteme	 **Algorithmen**
Zu entwickelnde Kompetenzen:  **Darstellen und Interpretieren**  **Modellieren**  Argumentieren  **Implementieren**	Die Schülerinnen und Schüler… …stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar. …entwickeln iterative und rekursive Algorithmen unter Nutzung der Strategien „Modularisierung“ und „Teilen und Herrschen“. …implementieren und erläutern iterative und rekursive Such- und Sortierverfahren.				

Unterrichtsvorhaben 4: *Rekursion als alternativer Lösungsansatz im Kontext selbstbezüglicher Problemstellungen*

Kompetenzerwartung / Beschreibung / Beispiele:

- Die SuS finden einen Zugang zu Rekursion durch rekursive Textformen (z. B. „Ein Vater hatte sieben Söhne“) oder im rahmen von Kunst (ein Bild im Bild im Bild….) und benennen grundsätzliche Merkmale und ggf. auch Probleme des „Phänomens“.

- Die SuS machen sich den Unterschied zwischen Iteration und Rekursion anhand von einfachen umgangssprachlichen Algorithmen bewusst. Zentrale Merkmale rekursiver Algorithmen werden dabei herausgestellt (Selbstaufruf, Abbruchbedingungen, Rücklauf).

- Die SuS entwickeln auf Basis bestimmter grafischer Vorgaben einfache lineare rekursive Algorithmen, die genau solche Grafiken erzeugen (siehe Materialhinweis). (Ebenso wäre eine rein textuelle / mathematische Herangehensweise möglich, etwa Algorithmen zur Bestimmung der Fakultät usw. möglich. Mittels Grafiken zu arbeiten bietet jedoch einen geeigneten visuellen Zugang). Sie trainieren damit den Einsatz von Rekursion letztlich als eine Form von Kontrollstruktur im Kontext selbstbezüglicher Problemstellungen, welche unter Rückgriff auf Schleifenstrukturen nicht trivial zu lösen sind.

- Die SuS analysieren Quelltexte zu verzweigenden und kaskadierenden Rekursionen und stellen deren jeweiligen Besonderheiten heraus. Sie versuchen die Ergebnisse dieser Algorithmen auf Basis der Analyse vorherzusehen (Was für eine Grafik wird erzeugt?) bevor der Quelltext ausgeführt wird. Es werden außerdem die Vorteile / Nachteile unterschiedlicher Rekursionstypen herausgestellt.

- Die SuS entwickeln einen Strategiekatalog, der bei der Implementierung rekursiver Algorithmen helfen soll und wenden diesen auf Aufgabenstellungen an (je nach Leistungsstärke des Kurses Fibonacci, Türme von Hanoi, usw.).

- Die SuS beschäftigen sich mit rekursiven Such- und Sortierstrategien, die auf dem Ansatz „Teile und Herrsche“ basieren.

Materialhinweis:

- **Literatur:** T. Kempe / A. Löhr (Hrsg.): Informatik. Band 2., Schöningh 2013.
- **Software:** Turtlegrafiken lassen sich innerhalb von Java z. B. mit der Bibliothek *aplu5* und dem darin enthaltenen Paket *turtle* erstellen. Das entsprechende API ist übersichtlich kurz.

Kommentar:

Rekursion wird durch den Kernlehrplan als Mittel zur Lösung von bestimmt gearteten Problemen nicht im Sinne der Eigenständigkeit direkt in den Vordergrund gerückt. Indirekt wird auf Rekursion jedoch innerhalb des Kernlehrplans immer wieder in unterschiedlichen Zusammenhängen zurückgegriffen: Knotenelemente innerhalb der Datenstruktur Liste / Schlange / Stapel, rekursive Algorithmen im Rahmen der Traversierung von Bäumen, Sortierverfahren usw.. Auch wenn sich Rekursion in die entsprechenden Themengebiete direkt einbetten lässt, erscheint es sinnvoll, Rekursion im Rahmen eines eigenständigen Unterrichtvorhabens gesondert zu betrachten.

Unterrichtsvorhaben 5: ***Modellierung und Implementierung von Anwendungen mit dynamischen, nichtlinearen Datenstrukturen: binäre Bäume***					
Inhaltsfelder:	 **Daten und ihre Strukturierung**	 Formale Sprachen und Automaten	 Informatik, Mensch und Gesellschaft	 Informatiksysteme	 **Algorithmen**
Zu entwickelnde Kompetenzen:	Die Schülerinnen und Schüler…				
 **Darstellen und Interpretieren**	…stellen lineare und nichtlineare Strukturen grafisch dar und erläutern ihren Aufbau.				
 Modellieren					
 **Argumentieren**	…analysieren und erläutern Algorithmen und Programme. …erläutern Operationen dynamischer (nichtlinearer) Datenstrukturen. ..beurteilen syntaktische Korrektheit und die Funktionalität von Programmen.				
 **Implementieren**	…implementieren Klassen in einer Programmiersprache auch unter Nutzung dokumentierter Klassenbibliotheken. …modifizieren Algorithmen und Programme. …testen Programme systematisch anhand von Beispielen. …nutzen Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen. …interpretieren Fehlermeldungen und korrigieren den Quellcode.				

Unterrichtsvorhaben 5: *Modellierung und Implementierung von Anwendungen mit dynamischen, nichtlinearen Datenstrukturen: binäre Bäume*

Kompetenzerwartung / Beschreibung / Beispiele:

- Die SuS werden durch unterschiedliche Situationen zur Datenstruktur eines binären Baumes geführt. Dies kann z. B. durch die Entwicklung einer möglichst günstigen Strategie beim Spiel „Zahlenraten“ (hier im Sinne eines normalen Spiels, d.h. nicht ein Spiel am Computer, oder eine Implementation in Java) erfolgen, bei der so wenig Fragen wie möglich zu stellen sind (durch halbieren des jeweiligen Zahlenraums. Dieser Ansatz folgt unweigerlich zu einer Baumstruktur. Dass es sich im konkreten Fall nicht nur um einen binären Baum, sondern um einen binären Suchbaum handelt spielt keine Rolle. Es geht einzig darum, dass für die Schüler transparent wird, dass sich eine Datenstruktur verzweigt)

- Die SuS setzen die Datenstruktur eines binären Baumes gegenüber der ihnen bekannten Datenstruktur einer Liste und grenzen beide Formen voneinander ab. Sie präzisieren damit ihr Verständnis über binäre Bäume und leiten selbstständig eine entsprechende (rekursive!) Definition ab.

- Die SuS beschäftigen sich (zunächst mit dem Verzicht auf Implementation) mit weiteren Übungen zu binären Bäumen. Als gut geeignet und im Folgenden als begleitendes Beispiel herangezogen hat sich z. B. die Auseinandersetzung mit Huffman-Kodierung erwiesen.

- Die SuS entwickeln Ideen zu einer möglichen Implementation von binären Bäumen in Java. Die im Rahmen der Vorgaben des Zentralabiturs zur Verfügung gestellte Klasse *binaryTree* wird im Anschluss hinsichtlich der durch die SuS entwickelten Ideen analysierend verglichen.

- Die SuS implementieren unter Rückgriff der Klasse *binaryTree* kleinere Programme zu Testzwecken und trainieren damit ihre Basiskompetenz im Umgang mit dieser Klasse.

- Die SuS implementieren einen Huffman-Code-Baum.

- Die SuS entwickeln einen Algorithmus in Form eines Struktogramms, welcher in der Lage ist, eine verschlüsselte Huffman-Nachricht bei gegebenem zugehörigen Huffman-Codebaum automatisiert zu dechiffrieren. Die Struktogramme werden nach gegenseitigem Vergleich ggf. korrigiert und anschließend in Java implementiert.

Materialhinweis:

- **Literatur:** A. Löhr / T. Kempe: Informatik 2. Modellierung, Datenstrukturen und Algorithme, Schönigh 2012, S. 132 – 150.
- **Software:** Das (Hand-)Zeichnen von Struktogrammen kann sich als zeitraubend erweisen, wenn ständig vergessene neue Elemente eingefügt werden müssen, um den Ablauf eines Algorithmus zu korrigieren. Günstig erweist sich in diesem Zusammenhang der Struktogrammeditor von Kevin Krummenauer (Freeware), online unter: http://whiledo.de/index.php?p=struktogrammeditor (zuletzt abgerufen am 13.8.2016)

Kommentar:

Im Vergleich zum Unterrichtsvorhaben 2 geht es innerhalb dieses Unterrichtsvorhabens ausdrücklich nicht darum, dass SuS eine eigene Klasse *binaryTree* implementieren sollen. Im Sinne der Vorgaben durch den Kernlehrplan sollen SuS vielmehr darauf trainiert werden, vorgegebene Klassen nutzen zu können. Die Realisierung (bzw. der Implementationsansatz) einer eigenen Klasse *List* durch SuS im Rahmen von Unterrichtsvorhaben 2 zielt darauf ab, die Funktionsprinzipien einer solchen Klasse durch eigenständiges Tun zu durchleuchten, die Kompetenzen im Bereich der Implementation komplexerer Klassen zu stärken. Dieser Vorgang muss hier im Rahmen von binären Bäumen nicht wiederholt werden und würde auch zu viel Zeit in Anspruch nehmen.

3.4 Unterrichtsvorhaben in der Qualifikationsphase 2

3.4.1 Übersichtsraster der Unterrichtsvorhaben in der Qualifikationsphase 2

Nachfolgend sind Übersichtsraster der Unterrichtsvorhaben und die Konkretisierung der Unterrichtsvorhaben für die Qualifikationsphase 2 analog zur Qualifikationsphase 1 aufgeführt (die Bezeichnungen der Unterrichtsvorhaben sind keine Themenformulierungen). Die jeweiligen Zeitangaben zu den Unterrichtsvorhaben verstehen sich als Orientierungsgrößen. Das theoretisch zur Verfügung stehende Gesamtvolumen von bis zu 85 Schulstunden wird im Rahmen der Qualifikationsphase 2 bewusst nicht ausgereizt..

Zum Teil haben die Unterrichtsvorhaben inhaltliche Voraussetzungen und bauen aufeinander auf. Das bedeutet nicht, dass eine durchgehend feste Reihenfolge der Unterrichtsvorhaben in der Durchführung eingehalten werden muss, es müssen lediglich die Abhängigkeiten berücksichtigt werden.

Unterrichtsvorhaben	Zeitbedarf	Voraussetzungen? / Wann?
1: *Modellierung und Implementierung von Anwendungen mit dynamischen, nichtlinearen Datenstrukturen binäre Suchbäume*	16 Stunden	Keine Voraussetzung
2: *Endliche Automaten und formale Sprachen*	20 Stunden	Keine Voraussetzung
3: *Prinzipielle Arbeitsweise eines Computers und Grenzen der Automatisierbarkeit*	10 Stunden	Unterrichtsvorhaben 2
4: *Sicherheit und Datenschutz in Netzstrukturen / Netzwerktechnologie*	16 Stunden	Keine Voraussetzung
5: *Wiederholungsphase*	15 Stunden	Am Ende der Qualifikationsphase 2

3.4.2 Konkretisierte Unterrichtsvorhaben in der Qualifikationsphase 2

Unterrichtsvorhaben 1: ***Modellierung und Implementierung von Anwendungen mit dynamischen, nichtlinearen Datenstrukturen Binäre Suchbäume***					
Inhaltsfelder:	**Daten und ihre Strukturierung**	Formale Sprachen und Automaten	Informatik, Mensch und Gesellschaft	Informatiksysteme	**Algorithmen**
Zu entwickelnde Kompetenzen:	Die Schülerinnen und Schüler…				
Darstellen und Interpretieren	…stellen iterative und rekursive Algorithmen umgangssprachlich und grafisch dar.*				
Modellieren					
Argumentieren	…analysieren und erläutern Algorithmen und Programme. …erläutern Operationen dynamischer (nichtlinearer) Datenstrukturen. …beurteilen die Effizienz von Algorithmen unter Berücksichtigung des Speicherbedarfs und der Zahl der Operationen.				
Implementieren	…modifizieren Algorithmen und Programme. …implementieren iterative und rekursive Algorithmen auch unter Verwendung von dynamischen Datenstrukturen. …nutzen die Syntax und Semantik einer Programmiersprache bei der Implementierung und zur Analyse von Programmen. …beurteilen die syntaktische Korrektheit und die Funktionalität von Programmen. …interpretieren Fehlermeldungen und korrigieren Quellcode.				

* Die konkretisierten Kompetenzerwartungen sind dem Kernlehrplan entnommen. Sie stellen den jeweiligen kompetenzbezogenen Kern des Unterrichtsvorhabens dar. Darüber hinaus sind weitere konkretisierte Kompetenzerwartungen im Sinne des Kernlehrplans möglich. Gleiches gilt für die nachfolgenden Unterrichtsvorhaben.

Unterrichtsvorhaben 1: ***Modellierung und Implementierung von Anwendungen mit dynamischen, nichtlinearen Datenstrukturen***
Binäre Suchbäume

Kompetenzerwartung / Beschreibung / Beispiele:

Das Unterrichtsvorhaben knüpft unmittelbar an das Unterrichtsvorhaben 5 der Qualifikationsphase 1 an (*Modellierung und Implementierung von Anwendungen mit dynamischen, nichtlinearen Datenstrukturen: binäre Bäume)*.

- Die Schülerinnen und Schüler vergegenwärtigen sich das grundsätzliche Konzept binärer Bäume im Sinne einer Kurzwiederholung. Dabei liegt der Fokus auf der ursprünglichen Intention, Suchanfragen möglichst effizient zu bearbeiten, was im Rahmen des Unterrichtsvorhabens 5 der Qualifikationsphase 1 nicht weiter beachtet wurde. Als Beispiel und Diskussionsgrundlage kann ein fiktionales Nutzerportal hinzugezogen werden, bei dem die Accounts einmal als Liste und einmal als Baum angelegt werden, um schließlich zu einer Definition eines binären Suchbaumes zu gelangen.
- Die Schülerinnen und Schüler entwickeln das nahe liegende notwendige Methodenrepertoire im Kontext binärer Suchbäume um:
  a) …Knoteninhalte in einem bestehenden Baum zu suchen
  b) …neue Knoten an die passende Stelle hinzuzufügen
  c) …bestehende Knoten zu löschen und die Struktur im Sinne der Definition eines binären Suchbaumes zu erhalten.

  Der Schwerpunkt liegt hierbei auf dem grundsätzlichen Verständnis immer im Blick auf dem visuellen Zugang zu binären Suchbäumen, sowie der Entwicklung eines zugehörigen funktionsfähigen Pseudocodes. Es geht nicht um eine Implementation in Java.
- Die Schülerinnen und Schüler machen sich mit den Abiturklassen zu binären Suchbäumen vertraut. Dabei gleichen sie ihre Basismethoden (Pseudocode) mit dem angebotenen Methodenrepertoire ab.
- Im Rahmen der Abiturklassen der binären Suchbäume werden Interfaces benutzt (*ComparableContent*). Interfaces sind den Schülerinnen und Schülern noch unbekannt. Innerhalb eines Kurz-Exkurses wird diese Lücke inklusive der Abgrenzung zu abstrakten Klassen geschlossen (losgelöst von der eigentlichen Thematik binärer Suchbäumen).
- In einer Kurzprojektphase implementieren die Schülerinnen und Schüler einen konkreten Anwendungskontext, bei dem binäre Suchbäume zum Einsatz kommen. (Z. B. ausgehend vom Ausgangsbeispiel des Unterrichtsvorhabens die Verwaltung von Accounts des fiktiven Nutzerportals). Die Schülerinnen und Schüler:
  a) …erstellen eine eigene Inhaltsklasse für Knotenobjekte und implementieren dabei das vorgegebene Interface *ComparableContent*.
  b) …testen ausführlich die Methoden, um Knotenelemente zu Suchen, hinzuzufügen und zu löschen.
  c) …nutzen eine vorgegebene Klasse, um die Vorgänge innerhalb des Baumes zu visualisieren.
- Für unterschiedlichste Anwendungszwecke müssen häufig alle Knoten eines Baumes systematisch durchlaufen werden. Die Schülerinnen und Schüler beschäftigen sich mit dieser Problematik und setzen sich mit den entsprechend unterschiedlichen Traversierungsmodi auseinander. Sie formulieren einen rekursiven Pseudocode und implementieren diesen anschließend innerhalb ihres Projektes.
- Die Schülerinnen und Schüler setzen sich mit Grenzen binärer Suchbäume auseinander. Dabei werden sie mit einem unausgewogenen Baum konfrontiert, der nahezu einer Liste entspricht und suchen entsprechende Ideen, um diese Degeneration zu verhindern. Das Konzept von AVL-Bäumen ist nicht Teil der Vorgaben des Kernlehrplanes, kann hier jedoch angeschnitten werden. In jedem Fall macht es Sinn die Grenzen binärer Suchbäume kurz in den Fokus zu rücken.

Unterrichtsvorhaben 2: ***Endliche Automaten und formale Sprachen***					
Inhaltsfelder:	Daten und ihre Strukturierung	**Formale Sprachen und Automaten**	Informatik, Mensch und Gesellschaft	**Informatiksysteme**	Algorithmen

Zu entwickelnde Kompetenzen:	Die Schülerinnen und Schüler…
 **Darstellen und Interpretieren**	…ermitteln die Sprache, die ein endlicher Automat akzeptiert. …stellen endliche Automaten in Tabelle oder Graph dar und überführen sie in die jeweils andere Darstellungsform. …beschreiben an Beispielen den Zusammenhang zwischen Automaten und Grammatiken.
 **Modellieren**	…entwickeln und modifizieren zu einer Problemstellung endliche Automaten. …entwickeln zur Grammatik einer regulären Sprache einen zugehörigen endlichen Automaten. …modifizieren Grammatiken regulärer Sprachen. …entwickeln zu einer regulären Sprache eine Grammatik, die die Sprache erzeugt. …entwickeln zur akzeptierten Sprache eines Automaten eine zugehörige Grammatik.
 **Argumentieren**	…analysieren und erläutern die Eigenschaften endlicher Automaten einschließlich ihres Verhaltens bei bestimmten Eingaben. …analysieren und erläutern Grammatiken regulärer Sprachen. …ermitteln die formale Sprache, die durch eine Grammatik erzeugt wird. …zeigen die Grenzen endlicher Automaten und regulärer Grammatiken im Anwendungszusammenhang.
 **Implementieren**	…wenden eine didaktische orientierte Entwicklungsumgebung zur Demonstration, zum Entwurf, zur Implementierung und zum Test von Informatiksystemen an.

Unterrichtsvorhaben 2: ***Endliche Automaten und formale Sprachen***

Kompetenzerwartung / Beschreibung / Beispiele:

- Als Hinführung in die Thematik eignet sich das Bauer-Hund-Ziege-Kohlkopf-Problem (BHZK-Problem): Hierbei müssen alle Akteure von der linken zur rechten Seite an einem Flussufer wechseln. Im Boot hat immer nur der Bauer plus ein weiterer Akteur platz. Der Bauer darf nicht die Ziege mit dem Kohlkopf alleine lassen. Ebenso darf der Hund nicht mit der Ziege alleine gelassen werden. Auch die Konstellation Hund, Ziege und Kohlkopf ohne Bauer ist unzulässig.
  Anhand dieser Problemstellungen lassen sich zentrale Elemente des Unterrichtvorhabens erarbeiten:
  a) Die immer wieder im Zentrum des Unterrichtvorhabens stehenden Fragen können aufgegriffen werden: Gibt es zu einem bestimmten Problem eine Lösung? Gibt es mehrere Lösungen? Gibt es eine beste Lösung?
  b) Unterschiedliche Darstellungsformen von Automaten mit ihren jeweiligen Vor- und Nachteilen können anhand des BHZK-Problems erarbeitet werden: Transitionsgraph, Formale Darstellung, Transitionstabelle
  c) Das BHZK-Problem macht deutlich, worum es bei Automaten überhaupt geht: um die vereinfachte Darstellung des Verhaltens einer abstrakten Maschine.

- Die SuS trainieren ihre Kompetenz im Umgang mit formaler Darstellung, Transitiosgraph und Transitionstabellen von Automaten und deren Überführung ineinander anhand weiterer Übungsbeispiele (Snack-Automat, Parser, usw.). Sinnvollerweise werden dabei softwareorientierte Lernumgebungen hinzugezogen (z. B. CeNFA).

- Vermeintlicher Entscheidungsfreiheit: Determinismus und Nichtdeterminismus im Kontext endlicher Automaten.

- Das bisherige Verständnis von Automaten wird erweitert: Ein Automat wird als einen Akzeptor verstanden, der entscheiden kann, ob ein bestimmtes Wort zu einer Sprache gehört oder nicht.

- Die SuS entwickeln reguläre Grammatiken um in Abgrenzung zu Akzeptoren Wörter nicht zu überprüfen, sondern Wörter zu generieren. (Auch wenn reguläre Ausdrücke nicht durch den Kernlehrplan vorgesehen sind, macht es durchaus Sinn diese als Brücke zwischen Automaten und Grammatiken hinzuzuziehen.)

- Die SuS setzen sich mit den Grenzen von Automaten bzw. den Grenzen der regulären Sprachen auseinander (typischerweise durch $L_1=(ab)^n$ im Vergleich zu $L_2=a^nb^n$): Ein Automat hat abgesehen von seinen Zuständen im engeren Sinne keinen Speicher, um sich etwas „zu merken“.

- Je nach Kurssituation und zur Verfügung stehender Zeit besteht die Möglichkeit, Inhalte des Leistungskurses anzuschneiden. Die Inhalte sind nicht schwieriger als die des Grundkurses und knüpfen unmittelbar an das reguläre Ende des Unterrichtvorhabens 2 an: Automaten mit Speicher: Kellerautomat / kontextfreie Sprachen / Chomskyhierarchie / Turing-Maschine / Busy-Beaver).

Unterrichtsvorhaben 3: ***Prinzipielle Arbeitsweise eines Computers und Grenzen der Automatisierbarkeit***					
Inhaltsfelder:	Daten und ihre Strukturierung	Formale Sprachen und Automaten	**Informatik, Mensch und Gesellschaft**	**Informatiksysteme**	Algorithmen
Zu entwickelnde Kompetenzen: Darstellen und Interpretieren Modellieren **Argumentieren** Implementieren	Die Schülerinnen und Schüler… …erläutern die Ausführung eines einfachen maschinennahen Programms sowie die Datenspeicherung auf einer „Von-Neumann-Architektur“. …untersuchen und beurteilen Grenzen des Problemlösens mit Informatiksystemen.				

Unterrichtsvorhaben 3: ***Prinzipielle Arbeitsweise eines Computers und Grenzen der Automatisierbarkeit***

Kompetenzerwartung / Beschreibung / Beispiele:

Der schulinterne Lehrplan zitiert im Kontext dieses Unterrichtvorhabens den durch das Ministerium für Schule und Weiterbildung NRW veröffentlichten Beispiellehrplan:

- *Von-Neumann-Architektur und die Ausführung maschinennaher Programme:*
  a) prinzipieller Aufbau einer von Neumann-Architektur mit CPU, Rechenwerk, Steuerwerk, Register und Hauptspeicher
  b) einige maschinennahe Befehlen und ihre Repräsentation in einem BinärCode, der in einem Register gespeichert werden kann
  c) Analyse und Erläuterung der Funktionsweise eines einfachen maschinennahen Programms

- *Grenzen der Automatisierbarkeit:*
  a) Vorstellung des Halteproblems
  b) Unlösbarkeit des Halteproblems
  c) Beurteilung des Einsatzes von Informatiksystemen hinsichtlich prinzipieller Möglichkeiten und prinzipieller Grenzen

(Vgl.: Ministerium für Schule und Weiterbildung: Beispiel für einen schulinternen Lehrplan zum Kernlehrplan für die gymnasiale Oberstufe. Informatik, Stand: 30.03.2014, online unter: http://www.schulentwicklung.nrw.de/lehrplaene/lehrplannavigator-s-ii/gymnasiale-oberstufe/informatik/hinweise-und-beispiele/hinweise-und-beispiele.html, S. 54f)

Unterrichtsvorhaben 4: ***Sicherheit und Datenschutz in Netzstrukturen / Netzwerktechnologie***					
Inhaltsfelder:	 Daten und ihre Strukturierung	 Formale Sprachen und Automaten	 **Informatik, Mensch und Gesellschaft**	 **Informatiksysteme**	 Algorithmen
Zu entwickelnde Kompetenzen:  **Darstellen und Interpretieren**  Modellieren  **Argumentieren**  Implementieren	Die Schülerinnen und Schüler… …nutzen bereitgestellte Informatiksysteme und das Internet reflektiert zum Erschließen, zur Aufbereitung und Präsentation fachlicher Inhalte. …beschreiben und erläutern Netzwerk-Topologien, die Client-Server-Struktur und Protokolle sowie ein Schichtenmodell in Netzwerken. …untersuchen und bewerten anhand von Fallbeispielen die Auswirkungen des Einsatzes von Informatiksystemen, die Sicherheit von Informatiksystemen sowie die Einhaltung der Datenschutzbestimmungen und des Urheberrechts. …untersuchen und bewerten Problemlagen, die sich aus dem Einsatz von Informatiksystemen ergeben, hinsichtlich rechtlicher Vorgaben, ethischer Aspekte und gesellschaftlicher Werte unter Berücksichtigung unterschiedlicher Interessenlagen.			

Unterrichtsvorhaben 4: ***Sicherheit und Datenschutz in Netzstrukturen / Netzwerktechnologie***

Kompetenzerwartung / Beschreibung / Beispiele:

Der schulinterne Lehrplan orientiert sich im Kontext dieses Unterrichtvorhabens am Beispiellehrplan des Ministeriums für Schule und Weiterbildung NRW:

- Die SuS beschäftigen sich mit *„Netztopologien als Grundlage von Client-Server-Strukturen und TCP/IP Schichtenmodell als Beispiel für eine Paketübermittlung in einem Netz“*[*]. Zentrale Begrifflichkeiten können dabei auch sein: IPv4, IP-Adresse, Subnet-Maske, Adressraum, Host, Server, Client, Router, Switch, usw.
  Das zentrale didaktische Hilfsmittel bildet die Software „Filius“ (Freeware), zur Simulation von Rechnernetzen.

- Die SuS setzen sich mit *„Vertraulichkeit, Integrität, Authentizität in Netzwerken sowie symmetrische und asymmetrische kryptografische Verfahren (Cäsar-, Vigenère-, RSAVerfahren) als Methoden Daten im Netz verschlüsselt zu übertragen“*[*] auseinander.
  Je nach zur Verfügung stehender Zeit kann der Themenbereich zu Datensicherheit unmittelbar erweitert werden durch Bedeutung und Nutzung von Firewalls, Sniffertools wie Wireshark, usw. was im Kontext der Software Filius wiederum praktisch erprobt werden kann.

[*](Vgl.: Ministerium für Schule und Weiterbildung: Beispiel für einen schulinternen Lehrplan zum Kernlehrplan für die gymnasiale Oberstufe. Informatik, Stand: 30.03.2014, online unter: http://www.schulentwicklung.nrw.de/lehrplaene/lehrplannavigator-s-ii/gymnasiale-oberstufe/informatik/hinweise-und-beispiele/hinweise-und-beispiele.html, S. 45f)

Materialhinweis:

- **Literatur:** A. Löhr / T. Kempe: Informatik 3. Netzwerkanwendungen, Informatik und Gesellschaft, Datenbanken und theoretische Informatik, Schönigh 2012, S. 6 – 51.
- **Software:** C. Eibl / S. Freischlad: Filius (didaktische Lernumgebung für Netzstrukturen und Netzwerktechnologie), Freeware, online unter**:** www.lernsoftware-filius.de

Unterrichtsvorhaben 5: ***Wiederholungsphase***					
Inhaltsfelder:	**Daten und ihre Strukturierung**	**Formale Sprachen und Automaten**	**Informatik, Mensch und Gesellschaft**	**Informatiksysteme**	**Algorithmen**
Zu entwickelnde Kompetenzen: **Darstellen und Interpretieren** **Modellieren** **Argumentieren** **Implementieren**	Die Schülerinnen und Schüler… siehe jeweiligen Unterrichtsvorhaben der Qualifikationsphase 1 + 2				

Unterrichtsvorhaben 5: *Wiederholungsphase*

Kompetenzerwartung / Beschreibung / Beispiele:

- Die Wiederholungsphase hat das Ziel, – insbesondere mit Blick auf Schülerinnen und Schüler, die das Fach Informatik als drittes oder viertes Abiturfach gewählt haben – die Inhalte der Qualifikationsphase 1 und 2 nochmals in ihrer Gesamtheit zu betrachten. Dabei geht darum:
  …das sich Schülerinnen und Schüler zunächst selbst in kompakter Form eine Übersicht über alle inhaltliche Themengebiete verschaffen.
  …eventuelle Lücken zu schließen ohne dabei jedoch Inhalte grundsätzlich zu erweitern.
  …einzelne Teilbereiche nochmals zu trainieren und fachliche bzw. auch methodische Kompetenzen zu stärken.

  …dass solche Schülerinnen und Schüler, die das Fach Informatik als drittes oder viertes Abiturfach gewählt haben:
  - …sich mit den jeweils gültigen Abiturklassen auseinandersetzen.
  - …sich nochmals der Bedeutung der gängigen Operatoren bei Aufgabenstellungen vergegenwärtigen.
  - …sich je nach Situation ggf. dazu bereit erklären, an der Simulation einer mündlichen Abiturprüfung teilzunehmen.

- Sofern sich im Kurs keine Schülerinnen und Schüler befinden, die das Fach Informatik als drittes oder viertes Abiturfach gewählt haben und alle verpflichtenden Unterrichtsinhalte abgedeckt sind, ergeben sich Freiräume, die unterschiedlich genutzt werden können: Fortführung und Erweiterungen einzelner Unterrichtsvorhaben, App-Entwicklung im Kontext von Handys, Physical-Computing, usw.

3.5 Exkursionen in der Qualifikationsphase

Im Rahmen der Möglichkeiten (Abhängigkeiten allgemein zu Klausurterminen, etc.) kann in der Qualifikationsphase 1 bzw. Qualifikationsphase 2 eine Exkursion zum Heinz Nixdorf Forum durchgeführt werden. Zwischen dem Angebot der Dauerausstellung und den Unterrichtsvorhaben der Qualifikationsphase 1 und 2 bzw. auch den Unterrichtsvorhaben der Einführungsphase bieten sich unterschiedliche Schnittpunkte an, sodass eine Exkursion keinem konkreten Unterrichtsvorhaben zugeordnet ist.

4. Leistungsbewertungskonzept in der Qualifikationsphase

4.1 Grundsätze der Leistungsbewertung in der Qualifikationsphase

Auf der Grundlage von §48 SchulG, §1 bis §19 APO-GOSt, sowie Kapitel 3 des Kernelhrplans Informatik hat die Fachkonferenz im Einklang die nachfolgenden Grundsätze zur Leistungsbewertung 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.

4.2 Leistungsbewertung und Leistungsrückmeldung im Bereich der sonstigen Mitarbeit

- Zum Beurteilungsbereich „Sonstige Mitarbeit“ gehören alle im Zusammenhang mit dem Unterricht erbrachten schriftlichen, mündlichen und praktischen Leistungen (vgl. APO-GOSt, 2.11.2012, §15 Abs. 1)
- Die Leistungsbewertung und Leistungsmessung orientiert sich an den im Kernlehrplan aufgeführten Überprüfungsformen (vgl. Kapitel 3, Abschnitt „Überprüfungsformen“)
- Zusätzlich zu den Vorgaben in Kapitel 3 des Kernlehrplans Informatik verständigt sich die Fachkonferenz Informatik auf folgende Grundsätze und Absprachen zur Leistungsbewertung:
    - *Prozessbewertung*, z. B. Beobachtung des Lern- und Arbeitsverhaltens bei programmierpraktischen Aufgabenstellungen, usw.
    - *Präsentationsbewertung*, z. B. Bewertung von Präsentationen am Ende einer Projektphase, usw. (vgl. Kapitel 3, Abschnitt „Überprüfungsformen“. Hier wird nur von Erläuterung gesprochen, ohne das es expliziet um eine Erläuterung im Sinne einer Präsentation geht.)
    - *Produktbewertung*, z. B. Bewertung eines Programmier-/ Implementationsprojektes, die Dokumentation einer Programmieraufgabe, Lernplakat, usw.
- Sonstige Festlegungen zur Leistungsmessung und Leistungsrückmeldung sind ggf. in den konkretisierten Unterrichtsvorhaben aufgeführt.

- Die Führung eines durchgehenden Materialordners für die gymnasiale Oberstufe ist obligatorisch und kann nach vorhergehender Festlegung der Kriterien zur Bewertung herangezogen werden.

4.3 Leistungsbewertung und Leistungsrückmeldung im Bereich Klausuren

- Die Klausuren innerhalb der Qualifikationsphase 1 haben den Umfang von zwei Schulstunden, Klausuren in der Qualifikationsphase 2 einen Umfang von drei Schulstunden.
- Je nach aktuellem Unterrichtsinhalt können Klausuren vollständig, oder in Teilen aus (programmier)praktischer Arbeit am Computer bestehen. Mit Blick auf die im Zentralabitur nicht vorgesehene Arbeit am Computer wird auf praktische Anteile innerhalb von Klausuren im Verlauf der Q2 zunehmend verzichtet, um die Klausuren bis spätestens zur Vorabiklausur den jeweils geltenden Abiturbedingungen anzupassen.
- Je nach Situation besteht im Rahmen der Qualifikationsphase 1 die Möglichkeit eine Klausur einheitlich für alle Schülerinnen und Schüler die das Fach schriftlich gewählt haben durch eine Projektarbeit zu ersetzen.
- Im Unterrichtsfach Informatik besteht grundsätzlich die Möglichkeit eine Facharbeit anzufertigen. Hierbei bietet sich die Entwicklung eines Softwareproduktes an.
- Die Bewertung und Leistungsrückmeldung von Klausuren erfolgt auf der Grundlage eines Kriterienkataloges. Eine Leistungsrückmeldung gibt perspektivische Hinweise für die individuelle Leistungsentwicklung.
- Die Bewertung der Klausuren bezieht sich auf die inhaltliche Leistung und auf die Darstellungsleistung. Die formale Korrektheit (z.B. bei der Generierung von Quellcode, Gebrauch einer korrekten Fachsprache, etc.) spielt ebenso eine Rolle.

5. Qualitätssicherung und Evaluation

Das schulinterne Curriculum Informatik ist nicht als starre Größe zu verstehen. Gerade mit Blick auf die kleine Fachschaft und das überschaubare Kursangebot (siehe *1. Die Fachschaft Informatik des Bertha–von–Suttner–Gymnasiums* bzw. *2.1. Unterrichtsstruktur im Fach Informatik*) ist es (stets unter der Maßgabe der Kernrichtlinien) problemlos möglich, Modifikationen bei den Unterrichtsvorhaben umzusetzen und zu erproben.
Eine quantitative Evaluation zu Unterrichtsmethoden und Unterrichtsinhalten findet derzeit nicht statt, durchaus aber Gruppengespräche mit Schülerinnen und Schülern, deren Position und Meinung dann auch in den Fachkonferenzen berücksichtigt wird. Auch Ideen und Anregung Seitens der Eltern werden bei den Fachkonferenzen einbezogen.