Freie und Hansestadt Hamburg
Behörde für Schule und Berufsbildung

Schriftliche Abiturprüfung

Informatik

Hinweise und Beispiele zu den
zentralen schriftlichen Prüfungsaufgaben

Impressum

Herausgeber:
Freie und Hansestadt Hamburg
Behörde für Schule und Berufsbildung
Landesinstitut für Lehrerbildung und Schulentwicklung
Felix-Dahn-Straße 3, 20357 Hamburg

Referatsleitung Unterrichtsentwicklung
mathematisch-naturwissenschaftlich-technischer Unterricht: Werner Renz

Fachreferentin Informatik: Monika Seiffert

Veröffentlicht auf: www.li.hamburg.de/publikationen/abiturpruefung

Hamburg 2012

Inhaltsverzeichnis

Vorwort

Sehr geehrte Kolleginnen und Kollegen,

ab dem Schuljahr 2013/2014 wird die Zahl der Fächer mit zentral gestellten Aufgaben in der Abiturprüfung u.a. um die MINT-Fächer Biologie, Chemie, Informatik und Physik erweitert. Die schriftlichen Abituraufgaben für diese Fächer werden zentral von der Schulbehörde erstellt. Sie beziehen sich auf Themen, die etwa 50 % des Unterrichts in der Studienstufe ausmachen und in den Rahmenplänen bereits verbindlich geregelt sind. Damit bleibt in der Profiloberstufe eine vernünftige Balance zwischen schulisch geprägten Themen und zentralen Leistungsanforderungen erhalten. Die fachspezifischen Hinweise im so genannten A-Heft, den „Regelungen für die zentralen schriftlichen Prüfungen“ für das Abitur 2014 (siehe Internet http://www.hamburg.de/abitur-2014/hamburg/3365184/start.html) informieren über die Schwerpunkte und Anforderungen der Prüfungsaufgaben. Sie ermöglichen damit eine langfristige Unterrichtsplanung.

Neu ab dem Abitur 2014 ist zudem die Wahlmöglichkeit für die zu bearbeitenden Prüfungsaufgaben durch die Schülerinnen und Schüler in allen MINT-Fächern. In den naturwissenschaftlichen Fächern und Informatik werden jeweils drei Aufgaben vorgelegt, von denen die Schülerinnen und Schüler zwei zur Bearbeitung auswählen.

Auf den nachfolgenden Seiten finden Sie zu Ihrer Orientierung Beispiele für zentrale Prüfungsaufgaben im Fach Informatik, in denen neben der Aufgabenstellung auch der Erwartungshorizont und die zugeordneten Bewertungseinheiten beschrieben sind.

In der Hoffnung, dass die vorliegende Handreichung hilfreich für Sie und Ihre Unterrichtsarbeit ist, wünsche ich Ihnen und Ihren Schülerinnen und Schülern eine erfolgreiche Vorbereitung auf die schriftliche Abiturprüfung.

Den Mitgliedern der Arbeitsgruppe, die diese Handreichung erstellte, danke ich herzlich für die geleistete Arbeit.

Werner Renz

1 Regelungen für die schriftliche Abiturprüfung

Die Fachlehrerin, der Fachlehrer

- erhält **drei** Aufgaben – Aufgabe I zum Thema „Objektorientierten Modellierung und Programmierung“, Aufgabe II zum Thema „Verteilte Systeme“, Aufgabe III auf grundlegendem Niveau zum Thema „Simulation“, auf erhöhtem Niveau zum Thema „Sprachübersetzung“.

Die Abiturientin, der Abiturient

- erhält **alle drei** Aufgaben und wählt aus den Aufgaben II und III eine aus.
- bearbeitet die Aufgabe I und eine der Aufgaben II und III,
- vermerkt auf der Reinschrift, welche Aufgabe sie / er bearbeitet hat,
- ist verpflichtet, die Vollständigkeit der vorgelegten Aufgaben vor Bearbeitungsbeginn zu überprüfen (Anzahl der Blätter, Anlagen usw.).

Bearbeitungszeit: Grundlegendes Niveau: **240** Minuten
Erhöhtes Niveau: **300** Minuten

Eine Einlesezeit von maximal **30** Minuten kann der Arbeitszeit vorgeschaltet werden. In dieser Zeit darf noch nicht mit der Lösung der Aufgaben begonnen werden.

Hilfsmittel: Taschenrechner (nicht programmierbar, nicht grafikfähig), Formelsammlung, Rechtschreiblexikon, aktuelle Datenschutzgesetze, ggf. IuKDG.

Die in den zentralen schriftlichen Abituraufgaben verwendeten **Operatoren** (Arbeitsaufträge) werden im Anhang genannt und erläutert.

Grundlage der schriftlichen Abiturprüfung ist der aktuell geltende Rahmenplan. Die jeweiligen Schwerpunktthemen entnehmen Sie bitte den *Regelungen für die zentralen schriftlichen Prüfungsaufgaben* des entsprechenden Jahrgangs.

Programmierparadigmen und -sprachen

Auf **grundlegendem Niveau** wird nur die Vertrautheit mit einer Programmiersprache erwartet, die sich sowohl für Implementationen nach dem objektorientierten Paradigma als auch nach dem imperativen Paradigma eignet. Alternativ kann dafür **Python oder Java** gewählt werden.

Auf **erhöhtem Niveau** wird die Vertrautheit mit dem objektorientierten, imperativen und funktionalen Paradigma sowie mit Implementationen in **Java und Scheme** erwartet.

2 Liste der Operatoren

Mehr noch als bei dezentralen Aufgaben, die immer im Kontext gemeinsamer Erfahrungen der Lehrkräfte und Schüler mit vorherigen Klausuren stehen, müssen zentrale Prüfungsaufgaben für die Abiturientinnen und Abiturienten eindeutig hinsichtlich des Arbeitsauftrages und der erwarteten Leistung formuliert sein. Die in den zentralen schriftlichen Abituraufgaben verwendeten Operatoren (Arbeitsaufträge) werden in der folgenden Tabelle definiert und inhaltlich gefüllt. Entsprechende Formulierungen in den Klausuren der Studienstufe sind ein wichtiger Teil der Vorbereitung der Schülerinnen und Schüler auf das Abitur.

Neben Definitionen für die Operatoren enthält die Tabelle auch Zuordnungen zu den Anforderungsbereichen **I**, **II** und **III**, wobei die konkrete Zuordnung auch vom Kontext der Aufgabenstellung abhängen kann und eine scharfe Trennung der Anforderungsbereiche nicht immer möglich ist

Operatoren	AB	Definitionen
analysieren, untersuchen	**II–III**	Unter gezielten Fragestellungen Elemente und Strukturmerkmale herausarbeiten und als Ergebnis darstellen
angeben, nennen	**I**	Elemente, Sachverhalte, Begriffe oder Daten ohne nähere Erläuterungen wiedergeben oder aufzählen
anwenden, übertragen	**II**	Einen bekannten Sachverhalt, eine bekannte Methode auf eine neue Problemstellung beziehen
auswerten	**II**	Daten oder Einzelergebnisse zu einer abschließenden Gesamtaussage zusammenführen
begründen	**II–III**	Einen angegebenen Sachverhalt auf Gesetzmäßigkeiten bzw. kausale Zusammenhänge zurückführen
berechnen	**I–II**	Ergebnisse von einem Ansatz ausgehend durch Rechenoperationen gewinnen
beschreiben	**I–II**	Strukturen, Sachverhalte oder Zusammenhänge unter Verwendung der Fachsprache in eigenen Worten veranschaulichen
bestimmen	**II**	Einen Lösungsweg darstellen und das Ergebnis formulieren
beurteilen	**III**	Zu einem Sachverhalt ein selbstständiges Urteil unter Verwendung von Fachwissen und Fachmethoden formulieren und begründen
bewerten	**III**	Eine eigene Position nach ausgewiesenen Normen oder Werten vertreten
darstellen	**I–II**	Zusammenhänge, Sachverhalte oder Verfahren strukturiert und fachsprachlich einwandfrei wiedergeben oder erörtern
einordnen, zuordnen	**I–II**	Mit erläuternden Hinweisen in einen Zusammenhang einfügen
entwerfen	**II–III**	Ein Konzept in seinen wesentliche Zügen prospektiv / planend erstellen
entwickeln	**II–III**	Eine Skizze, ein Szenario oder ein Modell erstellen, ein Verfahren erfinden und darstellen, eine Hypothese oder eine Theorie aufstellen
erklären	**II–III**	Rückführung eines Phänomens oder Sachverhalts auf Gesetzmäßigkeiten
erläutern	**II**	Ergebnisse, Sachverhalte oder Modelle nachvollziehbar und verständlich veranschaulichen
erörtern	**III**	Ein Beurteilungs- oder Bewertungsproblem erkennen und darstellen, unterschiedliche Positionen und Pro- und Kontra-Argumente abwägen und mit einem eigenen Urteil als Ergebnis abschließen.
herausarbeiten	**II–III**	Die wesentlichen Merkmale darstellen und auf den Punkt bringen

Operatoren	AB	Definitionen
implementieren	**II**	Das Umsetzen eines Algorithmus oder Software-Designs in einer Programmiersprache
skizzieren	**I–II**	Sachverhalte, Strukturen oder Ergebnisse kurz und übersichtlich darstellen mit Hilfe von z.B. Übersichten, Schemata, Diagrammen, Abbildungen, Tabellen
vergleichen, gegenüberstellen	**II–III**	Nach vorgegebenen oder selbst gewählten Gesichtspunkten Gemeinsamkeiten, Ähnlichkeiten und Unterschiede ermitteln und darstellen
zeigen	**II–III**	Aussage, Ergebnis oder Sachverhalt nach gültigen Regeln durch logische Überlegungen und / oder Berechnungen bestätigen

3 Aufgabenbeispiele

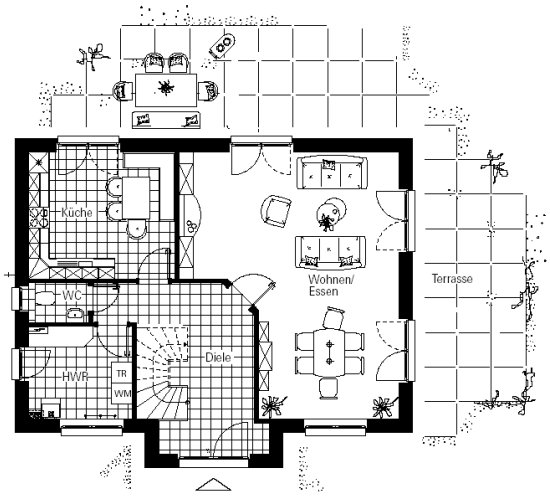
3.1 grundlegendes Anforderungsniveau

Aufgabe I

Objektorientierte Modellierung und Programmierung

Hausbau

Es soll ein einfaches CAD-Programm entwickelt werden, das bei der Planung des Baus von Ein- und Zweifamilienhäusern aus Fertigbauelementen hilft. Dieses Programm soll Käufern zur Verfügung gestellt werden, damit sie sich ihr Traumhaus planen können. Dabei müssen die Bauherren jeweils aus einem bestehenden Katalog vorgegebener Elemente auswählen. In dem Katalog sind die Fertigbauelemente enthalten. Es gibt beispielsweise mehrere Außen- und Innenwände, Türen, Fenster und Treppen. Da die Wandelemente auch transportiert werden müssen, gibt es sie in 5 Längen zwischen 2 m und 4 m. Längere Wände müssen aus Teilwänden zusammengesetzt werden.



Die obige Zeichnung zeigt den Grundriss des Erdgeschosses einer zweigeschossigen Wohnung. Sie können aus dieser Zeichnung ersehen, wie beispielsweise Treppen, Türen, Wände und Fenster dargestellt werden.

a) **Beschreiben** Sie typische Interaktionen des Anwenders mit dem Programm. Entwickeln Sie daraus grundlegende Anforderungen an das Programm. (5P)

Ein Programm dieses Umfangs erfordert eine lange Entwicklungszeit. Aus diesem Grund wird für diese Aufgabe nur ein kleiner Bereich betrachtet.
Der Grundriss soll von der zu entwickelnden Anwendung sehr einfach dargestellt werden. So können im Grundriss beispielsweise die Wände nur als senkrechte und waagerechte schmale Rechtecke dargestellt werden. Auf die Inneneinrichtung ist vollständig zu verzichten.

b) Das nebenstehende Bild zeigt die einfache Darstellung eines Zimmers mit vier Wänden, einer Tür und einem Doppelfenster.

In der Anlage 1 ist das Klassendiagramm abgebildet, auf dessen Grundlage dieses Bild gezeichnet wurde.

- **Erläutern** Sie dieses Diagramm. Gehen Sie dabei insbesondere auf die Beziehungen zwischen den Klassen ein.
- **Beschreiben** Sie, wie ein Raum aus den gegebenen Bauelementen erzeugt wird.
  Geben Sie an, welche **Objekte** benötigt werden, um diesen einfachen Grundriss zu erstellen.
  Geben Sie an, welche Nachrichten an die Objekte geschickt werden müssen, um den Grundriss angezeigt zu bekommen. Der Quellcode der Klassen befindet sich in der Anlage 2.
- Die Anlage 3 enthält die Abbildung der grundlegenden Grafikklassen als UML-Diagramm.
  Nennen Sie die für den Anwender wichtigen Eigenschaften der Klassen. (13P)

c) Bei der angegebenen Lösung wird zwischen der senkrechten und waagerechten Anordnung der Bauelemente im Grundriss unterschieden. **Erläutern** Sie die Implementation in der Klasse „BauElement“. Es wird vorgeschlagen statt der gegeben Implementation die Methoden setzeSenkrecht() und setzeWaagerecht() zu implementieren und zusätzlich die Methode istWaagerecht(), die die Orientierung des Bauelements ermittelt.
Beschreiben Sie die Veränderungen der Klasse BauElement und implementieren Sie die notwendigen Methoden.
Bewerten Sie die alte und die neue Lösung. (7P)

d) **Implementieren** Sie für die Klasse Wand einen Konstruktor, mit dem eine Wand mit der gewünschten Orientierung, Länge und Dicke an der gewünschten Position erstellt werden kann.
Erläutern Sie den Vorteil einer solchen Lösung. (4P)

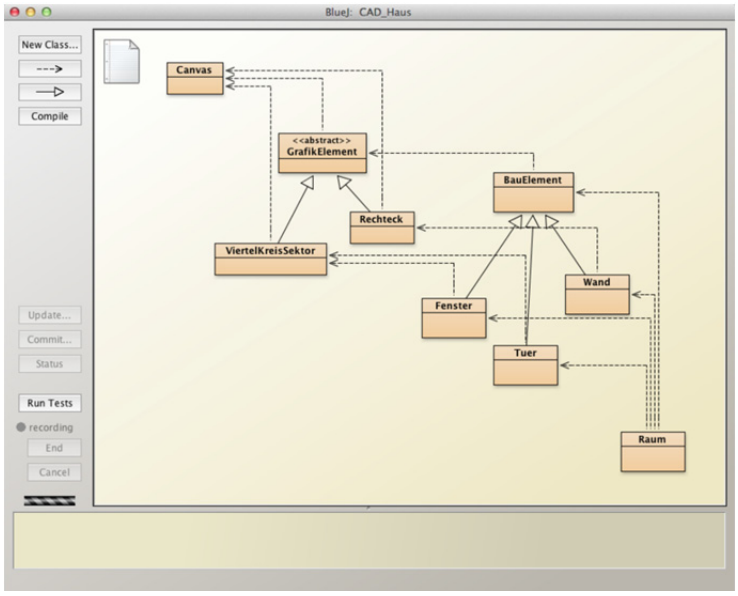
e) Die Tür wird im Programm durch einen Viertelkreis dargestellt. Sie soll etwas schöner dargestellt werden, so dass der Wanddurchbruch und die Tür sichtbar werden wie im nebenstehenden Bild. Verwenden Sie die bestehenden Klassen Rechteck und ViertelKreisSektor.
Passen Sie die Klasse Tür so an, dass das angegebene Bild entsteht. Erläutern Sie die Planung und Durchführung dazu. (10P)

f) Es wird vorgeschlagen, die Klasse BauElement von der Klasse GraphikElement erben zu lassen.
Bewerten Sie diesen Vorschlag. (5P)

g) Alle Wände eines Zimmers müssen gemeinsam verwaltet werden. In Java kann dies mit Hilfe einer ArrayList implementiert werden.
Beschreiben Sie das Konzept der ArrayList und ihre Verwendung in diesem Fall. (6P)

Anlage 1 zur Aufgabe „Hausbau“, Aufgabenteil b)

Klassendiagramm



Anlage 2 zur Aufgabe „Hausbau“, Aufgabenteil c)

Quellcode der Klassen

- BauElement
- Fenster
- Tuer
- Wand
- Raum

```
import java.util.*;

/**
 * Die Klasse BauElement erzeugt ein grundlegendes BauElement
 * Es wird mittig auf der Zeichenflaeche angeordnet.
 */
public class BauElement
{
   public static final boolean WAAGERECHT  = true;
   public static final boolean SENKRECHT  = false;

   private ArrayList<GrafikElement> grafikElemente;
   private boolean orientierung;

   /**
    * Konstruktor der Klasse BauElement
    */
   public BauElement()
   {
      // initialise instance variables
      this.grafikElemente = new ArrayList<GrafikElement>();
   }
```

```
24      /**
         * ergaenzeGrafikElement fuegt ein GraphikElement der Klasse BauElement
26       * der Darstellung des Bauelements hinzu.
         *
28       * @param grafikElement  GrahikElement
         */
30      public void ergaenzeGrafikElement(GrafikElement grafikElement)
        {
32         this.grafikElemente.add(grafikElement);
        }
34
        /**
36       * setzeOrientierung setzt die Orientierung des GrafikElements
         *
38       * @param  boolean   WAAGERECHT = true, SENKRECHT = false
         */
40      public void setzeOrientierung(boolean orientierung)
        {
42         this.orientierung = orientierung;
        }
44
        /**
46       * holeOrientierung bestimmt die Orientierung des GrafikElements
         *
48       * @param  boolean   WAAGERECHT = true, SENKRECHT = false
         */
50      public boolean holeOrientierung()
        {
52         return this.orientierung;
        }
54
        /**
56       * verschiebe verschiebt das BauElement um den angegebenen Wert
         *
58       * @param  xAbstand,yAbstand   Verschiebung in Pixel
         */
60       public void verschiebe(int xAbstand, int yAbstand)
         {
62          GrafikElement grafikElement;
            for  (GrafikElement grafikElement:grafikElemente)
64          {
               grafikElement = it.next();
66             grafikElement.bewegeHorizontal(xAbstand);
               grafikElement.bewegeVertikal(yAbstand);
68          };
         }
70
        /**
72       * zeichne zeichnet
         */
74      public void zeichne()
        {
76         for (Iterator<GrafikElement> it = this.grafikElemente.iterator(); it.hasNext();)
           {
78            it.next().macheSichtbar();
           };
80      }
    }
82
```

```
/**
 * Fenster
* Ein Klasse für Doppelfenster
 *
 */
public class Fenster extends BauElement
{
   private int groesse;
   private String farbe;
   private ViertelKreisSektor formLinks;
   private ViertelKreisSektor formRechts;

   /**
    * Konstruktor der Klasse Fenster
    */
   public Fenster()
   {
      // initialise instance variables
      this.groesse = 100;
      this.farbe = "gray";
      this.formLinks = new ViertelKreisSektor();
      this.formRechts = new ViertelKreisSektor();
      this.formLinks.aendereGroesse(this.groesse / 2);
      this.formRechts.aendereGroesse(this.groesse / 2);
      this.formRechts.aendereQuadrant(2);
      this.formRechts.bewegeHorizontal(this.groesse / 2);
      this.formLinks.bewegeHorizontal(-this.groesse / 2);
      this.formRechts.aendereFarbe(this.farbe);
      this.formLinks.aendereFarbe(this.farbe);
      ergaenzeGrafikElement(formRechts);
      ergaenzeGrafikElement(formLinks);
   }
}

/**
 * Tuer
 * Nur Tueren mit Anschlag rechts
 * Sie können horizontal oder vertikal angeordnet sein.
 * Sie können nach oben und unten bzw. rechts und links
 * geöffnet werden.
 *
 */
public class Tuer extends BauElement
{
   // Konstanten
   public static final boolean LINKS = true;
   public static final boolean RECHTS = false;
   public static final boolean OBEN = false;
   public static final boolean UNTEN = true;

   // instance variables - replace the example below with your own
   private int groesse;
   private boolean oeffnung;
   private String farbe;
   private ViertelKreisSektor form;

```

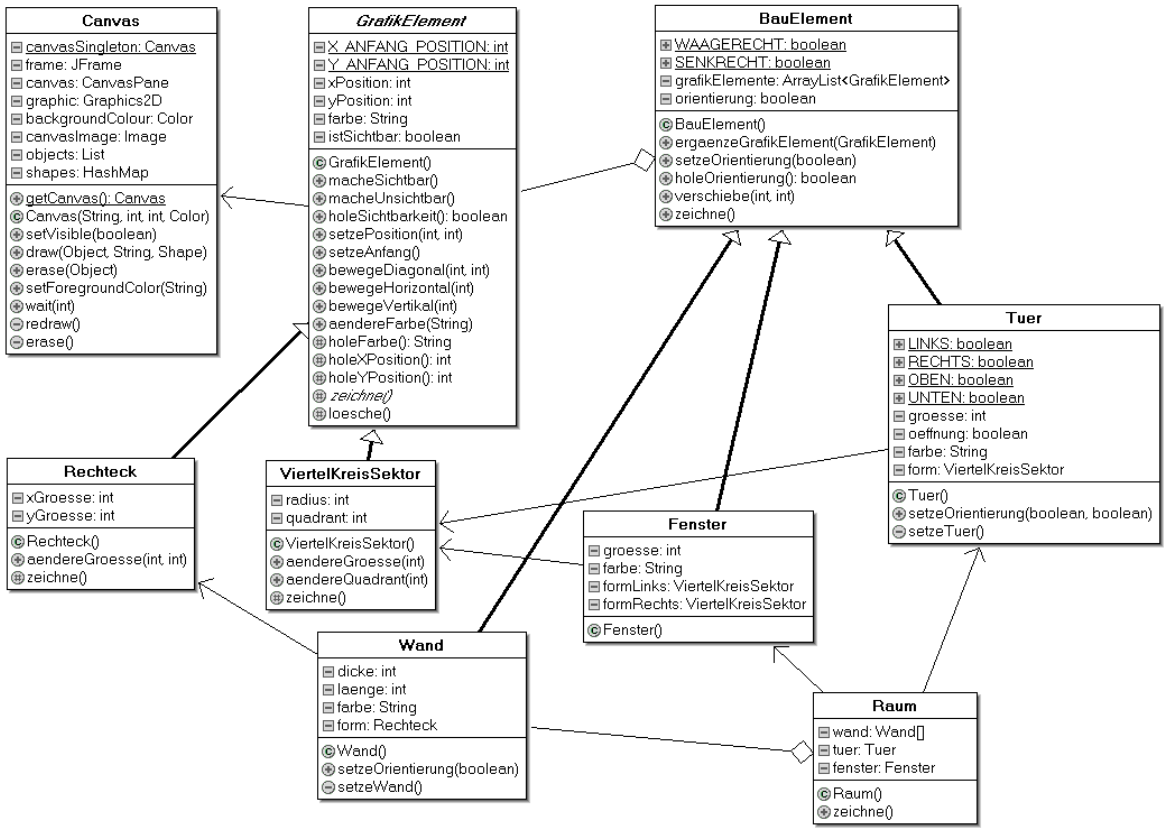
```
140     /**
         * Konstruktor der Klasse Tuer
142      */
        public Tuer()
144     {
           // initialise instance variables
146        this.groesse = 50;
           this.farbe = "gray";
148        this.form = new ViertelKreisSektor();
           this.form.aendereGroesse(this.groesse);
150        setzeOrientierung(BauElement.SENKRECHT, RECHTS);
           ergaenzeGrafikElement(form);
152     }

154     /**
         * setzeOrientierung setzt die Orientierung der Tuer
156      *
         * @param  orientierung boolean   WAAGERECHT = true, SENKRECHT = false
158      * @param  oeffnung     boolean   LINKS | UNTEN = true, RECHTS | OBEN = false
         */
160     public void setzeOrientierung(boolean orientierung, boolean oeffnung)
        {
162        setzeOrientierung(orientierung);
           this.oeffnung = oeffnung;
164        setzeTuer();
        }
166
        /*
168      * Setzt die Groesse der Form aus den Angaben der Wand.
         */
170     private void setzeTuer( )
        {
172        if (holeOrientierung() == SENKRECHT && this.oeffnung == RECHTS)
           {
174           this.form.aendereQuadrant(4);
           }
176        else if (holeOrientierung() == SENKRECHT && this.oeffnung == LINKS)
           {
178           this.form.aendereQuadrant(2);
           }
180        else if(holeOrientierung() == WAAGERECHT && this.oeffnung == OBEN)
           {
182           this.form.aendereQuadrant(1);
           }
184        else if (holeOrientierung() == WAAGERECHT && this.oeffnung == UNTEN)
           {
186           this.form.aendereQuadrant(3);
           }
188     }
    }
190
```

```
/**
 * Wand
 *
 */
public class Wand extends BauElement
{
   private int dicke;
   private int laenge;
   private String farbe;
   private Rechteck form;

   /**
    * Konstruktor der Klasse Wand
    */
   public Wand()
   {
      this.dicke = 10;
      this.laenge = 200;
      this.farbe = "black";               // Waende sind immer schwarz
      this.form = new Rechteck();
      this.form.aendereFarbe(this.farbe);
      this.form.bewegeDiagonal(-this.dicke / 2, -this.dicke / 2);
      setzeOrientierung(BauElement.SENKRECHT);
      setzeWand();
      ergaenzeGrafikElement(form);
   }

   /**
    * setzeOrientierung setzt die Orientierung der Wand
    *
    * @param  boolean   WAAGERECHT = true, SENKRECHT = false
    */
   public void setzeOrientierung(boolean orientierung)
   {
      super.setzeOrientierung(orientierung);
      setzeWand();
   }

   /*
    * Setzt die Groesse der Form aus den Angaben der Wand.
    */
   private void setzeWand( )
   {
      if (holeOrientierung())
      {
         this.form.aendereGroesse(this.laenge + this.dicke, this.dicke);
      }
      else
      {
         this.form.aendereGroesse(this.dicke, this.laenge + this.dicke);
      }
   }
}
```

```
/**
 * Die Klasse Raum
 * Es wird ein Raum zusammengestellt und auf der Zeichenfläche gezeichnet.
 */
public class Raum
{
   private Wand[] wand;
   private Tuer tuer;
   private Fenster fenster;

   /**
    * Konstruktor der Klasse Raum
    */
   public Raum()
   {
      wand = new Wand[4];
      for (int i = 0; i < 4; i++)
      {
         this.wand[i]= new Wand();
      }
      this.wand[1].setzeOrientierung(BauElement.WAAGERECHT); // Standard ist Senkrecht
      this.wand[3].setzeOrientierung(BauElement.WAAGERECHT);
      this.wand[0].verschiebe(-100,-100);
      this.wand[1].verschiebe(-100,-100);
      this.wand[2].verschiebe(100,-100);
      this.wand[3].verschiebe(-100,100);

      this.tuer = new Tuer();
      this.tuer.verschiebe(-20,-100);
      this.tuer.setzeOrientierung(BauElement.WAAGERECHT, Tuer.UNTEN);

      this.fenster = new Fenster();
      this.fenster.verschiebe(0,100);
   }

   /**
    * zeichne()
    *
    * Zeichnet den vollständigen Raum auf der Zeichenflaeche
    */
   public void zeichne()
   {
      for (int i = 0; i < 4; i++)
      {
         this.wand[i].zeichne();
      }
      this.tuer.zeichne();
      this.fenster.zeichne();
   }
}
```

Anlage 3 zur Aufgabe „Hausbau“, Aufgabenteil d)

UML-Diagramm



Erwartungshorizont

	Lösungsskizze	Zuordnung, Bewertung		
		I	II	III
a)	Der Anwender erstellt Geschosse, wählt, positioniert, verschiebt, dreht, löscht Bauelemente, er lässt den Plan zeichnen, er lässt den Preis des Hauses kalkulieren, er speichert und öffnet Varianten. Das Programm muss den Plan speichern und öffnen können. Objekte müssen markiert, Eigenschaften müssen geändert werden können. Elemente müssen aus Listen ausgewählt werden können. Für den Kunden wäre es zusätzlich hilfreich, wenn er mit dem Programm die Kosten kalkulieren könnte.	5		
b)	Es handelt sich um ein Klassendiagramm. In diesem Diagramm ist oben links die Klasse des Canvas angegeben. Diese Klasse wird von den grundlegenden Zeichenklassen ViertelKreisSektor, Rechteck und GrafikElement benutzt. GrafikElement ist <abstract>, d.h. von ihr kann kein Objekt erzeugt werden. Dabei sind Rechteck und ViertelKreisSektor Spezialfälle der Klasse GrafikElement. Diese Klassen werden von BauElement, Fenster, Tür und Wand genutzt. Auch hier wird eine allgemeine Klasse vorgegeben von der die Spezialfälle Fenster, Tuer und Wand abgeleitet werden. Die Klasse Raum nutzt die Klassen Fenster, Tuer und Wand. Es treten die folgenden Beziehung auf: Ist Spezialfall von: beispielsweise Rechteck / Grafikelement Benutzt: beispielsweise Tuer / ViertelKreisSektor Teil-/Ganzes: beispielsweiseRaum / Fenster, Tuer, Wand Aus den Klassen werden die notwendigen Objekte erzeugt. Insgesamt werden vier Objekte für die Wände, ein Objekt für die Tür und eines für das Fenster benötigt. Anschließend werden an die Objekte die Nachrichten mit den gewünschten Größen, Orientierungen und Positionen geschickt. Im letzten Schritt werden die Objekte gezeichnet. Objekte: Wand1 bis Wand4; Tuer Fenster Nachrichten: setzeOrientierung(BauElement.WAAGERECHT); verschiebe(-100,-100); zeichne(); Die Bauelemente enthalten Information über die Farbe, die Form und die Ausrichtung. Die Position wird von der Form verwaltet. Die Form kann aus mehreren Grafikelementen bestehen. Die Formen können verschoben werden, die Orientierung kann geändert werden. Das Bauelement kann gezeichnet werden. Nicht geändert werden kann die Farbe. (Großer Nachteil: Die gegebene Lösung unterstützt unsinnige Konstruktionen.)	9	4	

	Lösungsskizze	Zuordnung, Bewertung		
		I	II	III
c)	`/**` ` * setzeSenkrecht setzt die senkrechte Bauelements` ` */` `public void setzeSenkrecht()` `{` `  this.orientierung = SENKRECHT;` `}` `/**` ` * setzeWaagerecht setzt die waagerechte Bauelements` ` */` `public void setzeWaagerecht()` `{` `  this.orientierung = WAAGERECHT;` `}` `/**` ` * istWaagerecht ermittelt, ob Bauelements waagerecht` ` */` `public boolean istWaagerecht()` `{` `  return this.orientierung;` `}` Es müssen drei Methoden eingefügt werden, die die entsprechenden Werte setzen. Die Lösung ist die Bessere, der Anwender muss weniger Detail über die Implementierung der Klasse wissen. Die Parameter sind überflüssig. Es gilt: Multifunktionale Methoden sind zu vermeiden. Klare, eindeutige Strukturen sind vorzuziehen. Fehler können so vermieden werden.		5	2
d)	`/**` ` * Konstruktor der Klasse Wand` ` *` ` * @param dicke  Dicke der Wand in Pixeln, Werte 6, 10` ` * @param laenge  Laenge der Wand in Pixeln, Werte 100, 200, 300` ` * @param orientierung  WAAGERECHT = true, SENKRECHT = false` ` * @param xPosition  in Pixeln. 0,0 mittig` ` * @param yPosition  in Pixeln. 0,0 mittig` ` */` `public WandMitKonstruktor(int dicke, int laenge, boolean orientierung,` `  int xPosition, int yPosition)` `{` `  this.dicke = dicke;` `  this.laenge = laenge;` `  this.farbe = "black";` `  this.form = new Rechteck();` `  this.form.aendereFarbe(this.farbe);` `  this.form.bewegeDiagonal(-this.dicke / 2, -this.dicke / 2);` `  setzeOrientierung(orientierung);` `  this.form.bewegeDiagonal(xPosition, yPosition);` `  setzeWand();` `  ergaenzeGrafikElement(form);` `}` Der Vorteil zusätzlicher Konstruktoren besteht in der Möglichkeit, gleich bei der Erzeugung ein Objekt im gewünschten Zustand zu erhalten.		3	1
e)	Tuer setzt sich nun aus drei Elementen zusammen, d. h. es werden zusätzlich zwei dickere Linien benötigt. Dabei ist es möglich die Linien übereinander zu legen, dann muss die genaue Reihenfolge eingehalten werden, oder es werden zwei Linien nebeneinander gelegt. `/**` ` * Türen` ` * Nur Türen mit Anschlag rechts` ` * Sie können horizontal oder vertikal angeordnet sein.` ` * Sie können nach oben und unten bzw. rechts und links` ` * geöffnet werden.` ` */`			

	Lösungsskizze	Zuordnung, Bewertung		
		I	II	III

```
public class Tuer2 extends BauElement
{
   // Konstanten
   public static final boolean LINKS = true;
   public static final boolean RECHTS = false;
   public static final boolean OBEN = false;
   public static final boolean UNTEN = true;

   private int groesse, dickeWand, dickeTuer;
   private boolean oeffnung;
   private String farbe1, farbeDurchbruch, farbeTuer;
   private ViertelKreisSektor form;
   private Rechteck formDurchbruch;
   private Rechteck formTuer;

   /**
    * Konstruktor fuer Klasse Tuer2
    */
   public Tuer2()
   {
      this.groesse = 50;
      this.dickeWand = 10;
      this.dickeTuer = 5;
      this.farbe1 = "gray";
      this.farbeDurchbruch = "green";
      this.farbeTuer = "black";

      this.form = new ViertelKreisSektor();
      this.form.aendereGroesse(this.groesse);

      this.formDurchbruch = new Rechteck();
      this.formDurchbruch.aendereFarbe(this.farbeDurchbruch);

      this.formTuer = new Rechteck();
      this.formTuer.aendereFarbe(this.farbeTuer);

      setzeOrientierung(BauElement.SENKRECHT, RECHTS);
      ergaenzeGrafikElement(form);
      ergaenzeGrafikElement(formDurchbruch);
      ergaenzeGrafikElement(formTuer);
   }

   /**
    * setzeOrientierung setzt die Orientierung der Tuer
    * @param orientierung boolean WAAGERECHT = true, SENKRECHT = false
    * @param oeffnung     boolean LINKS | UNTEN = true, RECHTS | OBEN = false
    */
   public void setzeOrientierung(boolean orientierung, boolean oeffnung)
   {
      this.form.setzeAnfang();
      this.formDurchbruch.setzeAnfang();
      this.formTuer.setzeAnfang();
      setzeOrientierung(orientierung);
      this.oeffnung = oeffnung;
      setzeTuer();
   }

   private void setzeTuer()
   {
      if (holeOrientierung() == SENKRECHT)
      {
         this.formDurchbruch.aendereGroesse(this.dickeWand, this.groesse);
         this.formTuer.aendereGroesse(this.dickeTuer, this.groesse);
      }
      else
      {
         this.formDurchbruch.aendereGroesse(this.groesse, this.dickeWand);
         this.formTuer.aendereGroesse(this.groesse, this.dickeTuer);
      }
      if (holeOrientierung() == SENKRECHT && this.oeffnung == RECHTS )
      {
         this.form.aendereQuadrant(4);
         this.form.bewegeHorizontal(this.dickeWand / 2);
         this.formDurchbruch.bewegeHorizontal(-this.dickeWand / 2);
```

	Lösungsskizze	Zuordnung, Bewertung		
		I	II	III
	this.formTuer.bewegeHorizontal(0); } else if (holeOrientierung() == SENKRECHT && this.oeffnung == LINKS) { this.form.aendereQuadrant(2); this.form.bewegeHorizontal(-this.dickeWand / 2); this.formDurchbruch.bewegeHorizontal(-this.dickeWand / 2); this.formTuer.bewegeHorizontal(-this.dickeWand / 2); this.formDurchbruch.bewegeVertikal(-this.groesse); this.formTuer.bewegeVertikal(-this.groesse); } else if(holeOrientierung() == WAAGERECHT && this.oeffnung == OBEN) { this.form.aendereQuadrant(1); this.form.bewegeVertikal(-this.dickeWand / 2); this.formDurchbruch.bewegeVertikal(-this.dickeWand / 2); this.formTuer.bewegeVertikal(-this.dickeWand / 2); } else if (holeOrientierung() == WAAGERECHT && this.oeffnung == UNTEN) { this.form.aendereQuadrant(3); this.formDurchbruch.bewegeHorizontal(-this.groesse); this.formTuer.bewegeHorizontal(-this.groesse); this.form.bewegeVertikal(this.dickeWand / 2); this.formDurchbruch.bewegeVertikal(-this.dickeWand / 2); this.formTuer.bewegeVertikal(0); } } }		6	4
f)	Vorteil: Redundante Elemente werden vermieden, da man Bauelement in der aktuellen Lösung leicht so ändern kann, dass es genau die Methoden von GrafikElement nutzt und sie deshalb nicht neu implementieren muss, dies ist aber ein vordergründiges Kriterium. Nachteil: bei Erweiterung ist die Gefahr sehr groß, zu einer inkonsistenten Hierarchie zu kommen. Vererbung ist nur erlaubt, wenn gilt: Klasse A ist eine Spezialisierung der Klasse B. Dies ist in für die Klassen BauElement und GrafikElement nicht gegeben. Dabei gilt Spezialisierung nur in der Weise, dass es zu einer Erweiterung kommt, es darf keine Einengung vorliegen (Kreis-Ellipsen-Dilemma, Bertrand Meyer). Hier sollen beispielsweise die Bauelemente nur ganz bestimmte Farben haben, diese Farbe ist in der Klasse vorgegeben. Eine spätere Farbänderung ist nicht vorgesehen. Würde man BauElement von GrafikElement erben lassen, führt dies dazu, dass nachträglich die Farbe des Bauelements gesetzt werden kann, da von GrafikElement eine Methode geerbt wird, mit der die Farbe geändert werden kann. GrafikElement ist in der Regel flexibler als dies für die Bauelemente notwendig ist. Bei einer Modellierung mit Vererbung ist es nur trickreich möglich, diese Flexibilität zu unterbinden. Stichworte: Liskov-Prinzip, Kreis-Ellipsen-Dilemma. Aus diesen Gründen ist Delegation vorzuziehen.		2	3
g)	Es könnte das Array oder eine Collection: LinkedList, ArrayList, Vector verwendet werden. Ein Array vom Typ BauElement ist die einfachste Form, aber statisch. Das Programm legt die maximale Anzahl von Elementen fest. Der Zugriff auf die einzelnen Elemente des Feldes erfolgt über einen Index. In Arrays können auch einfache Datentypen gespeichert werden. – Hier haben sie auch ihre Berechtigung. Für Klassen sollte eine Collection eingesetzt werden. Collections sind flexibler und können nur Objekte enthalten. Einfache Datentypen müssten mit einem »Wrapper« eingepackt werden. Die Anzahl der enthaltenen Elemente passt sich während der Programmlaufzeit an die Erfordernisse an. Der Zugriff auf die Elemente der Collection erfolgt über einen Iterator. Beide Elemente sind in der Java-Bibliothek enthalten. Eine LinkedList könnte die Reihenfolge aneinander gefügter Elemente wieder-geben.			

	Lösungsskizze	Zuordnung, Bewertung		
		I	II	III
	`` import java.util.*; /** * Klasse Raum * erstellt einen Raum aus gegebenen Elementen und lässt den * Grundriss auf der Leinwand zeichnen. */ public class RaumMitNeuerTuer { private ArrayList<BauElement> bauElemente; /** * Konstruktor der Klasse Raum */ public RaumMitNeuerTuer() { this.bauElemente = new ArrayList<BauElement>(); Wand wand; for (int i = 0; i < 4; i++) { this.bauElemente.add(new Wand()); } this.bauElemente.get(1).setzeOrientierung(BauElement.WAAGERECHT); this.bauElemente.get(3).setzeOrientierung(BauElement.WAAGERECHT); this.bauElemente.get(0).verschiebe(-100,-100); this.bauElemente.get(1).verschiebe(-100,-100); this.bauElemente.get(2).verschiebe(100,-100); this.bauElemente.get(3).verschiebe(-100,100); Tuer2 tuer = new Tuer2(); this.bauElemente.add(tuer); tuer.setzeOrientierung(BauElement.WAAGERECHT, Tuer.UNTEN); tuer.verschiebe(-30,-100); tuer = new Tuer2(); this.bauElemente.add(tuer); tuer.setzeOrientierung(BauElement.SENKRECHT, Tuer.LINKS); tuer.verschiebe(-100,0); tuer = new Tuer2(); this.bauElemente.add(tuer); tuer.setzeOrientierung(BauElement.SENKRECHT, Tuer.RECHTS); tuer.verschiebe(100,0); tuer = new Tuer2(); this.bauElemente.add(tuer); tuer.setzeOrientierung(BauElement.WAAGERECHT, Tuer.OBEN); tuer.verschiebe(0,100); } /** * zeichne() */ public void zeichne() { for (Iterator<BauElement> it = this.bauElemente.iterator(); it.hasNext();) { it.next().zeichne(); } } /** * zeichne1() */ public void zeichne1() { for (BauElement bauElement:bauElemente) { bauElement.zeichne(); } } } ```		3	3
	Insgesamt 50 BWE	14	23	13

3.1 grundlegendes Anforderungsniveau

Aufgabe II

Kommunikation und Kryptographie

Bewerbungen

Wiebold Wichtig, der Leiter der Personalabteilung von *ZukunftPlus*, möchte zukünftig Bewerbungen auch per E-Mail entgegennehmen. Auf der Homepage der Firma werden die jeweils nötigen Informationen (Telefonnummern, E-Mail-Adressen, öffentlicher Schlüssel für das RSA-Verfahren) bekannt gegeben.

Es gibt vier Vorschläge für die praktische Umsetzung:

– Vorschlag A sieht keinerlei weitere Einschränkungen vor.

– Bei Vorschlag B wird ein symmetrisches Verschlüsselungsverfahren (z. B. DES) verwendet. Die einzelnen Bewerber müssen sich telefonisch mit dem Sekretariat der Personalabteilung in Verbindung setzen, damit ein Schlüsselaustausch nach Diffie-Hellman durchgeführt werden kann (*siehe Material 2*).

– Vorschlag C nutzt ein asymmetrisches Verschlüsselungsverfahren (RSA). Der Bewerber erhält den öffentlichen Schlüssel der Firma.

– Bei Vorschlag D verschlüsselt der Bewerber seine Unterlagen nach dem RSA-Verfahren mit seinem eigenen privaten Schlüssel.

a) **Gib an,** welche Aufgaben Protokolle haben, und erläutere dies anhand der Anlage 1. (7P)

b) **Beschreibe,** was man unter *symmetrischen* und *asymmetrischen* Verschlüsselungsverfahren versteht. (6P)

c) **Zeige,** dass mit den Werten $p = 23$, $s = 11$, $a = 3$ und $b = 2$ bei dem *Schlüsselaustausch* nach Diffie-Hellman (siehe Anlage 2) die beiden beteiligten Personen den gleichen Wert für den Schlüssel k erhalten. (9P)

d) **Erkläre,** warum der Begriff „Schlüsselaustausch“ irreführend ist. (4P)

Beim RSA-Verfahren sind zwei Angaben öffentlich bekannt (n und e) und beim Schlüsselaustausch nach Diffie-Hellman sogar vier verschiedene (p, s, α und β).

e) **Stelle dar,** worauf beim Schlüsselaustausch und beim RSA-Verfahren die relativ hohe Sicherheit basiert, obwohl zwei bzw. vier Angaben öffentlich bekannt sind. (6P)

f) **Beschreibe,** was man unter Vertraulichkeit, Integrität und Authentizität im Zusammenhang mit dem Austausch von Nachrichten versteht. (6P)

g) **Bewerte** die vier Vorschläge**. Entwickle** gegebenenfalls einen eigenen Vorschlag. Arbeite dabei auch heraus, welche stillschweigenden Annahmen bei der Formulierung der Vorschläge gemacht wurden. (12P)

Anlage 1 zur Aufgabe „Bewerbungen“, Aufgabenteil a)

Simple Mail Transfer Protocol (Beispiel)

Direkt nach dem Verbindungsaufbau über TCP meldet sich der Mailserver.

```
CLIENT                                  SERVER

                                        220 mail.example.com SMTP Foo Mailserver
HELO mail.example.org
                                        250 Ok
MAIL FROM: hans.muster@example.org
                                        250 Ok
RCPT TO: foo@example.com
                                        250 Ok
DATA
                                        354 End data with .
From: hans.muster@example.org
To: foo@example.com
Subject: Testmail
Date: Fri, 16 Dez 2010 13:10:50 +0200

Testmail
.
                                        250 Ok
QUIT
                                        221 Bye
```

Anlage 2 zur Aufgabe „Bewerbungen“, Aufgabenteil c)

Schlüsselaustausch nach Diffie-Hellmann

Der von Whitfield Diffie und Martin Hellmann entwickelte Algorithmus zur Bestimmung eines gemeinsamen Schlüssels läuft über drei Stufen.

Die beiden beteiligten Personen, nennen wir sie mal Alice und Bob, einigen sich auf eine (große) Primzahl p und eine beliebige Zahl s ($1 < s < p$). Diese beiden Zahlen können über einen unsicheren Kanal ausgetauscht werden; sie sind also als öffentlich bekannt anzusehen.

Alice und Bob wählen außerdem jeweils eine Zahl a bzw. b - ihre privaten geheimen Zahlen. Die Zahlen $\alpha = s^a \bmod p$ und $\beta = s^b \bmod p$ werden wieder über einen unsicheren Kanal ausgetauscht. Für etwaige Angreifer sind also insgesamt vier Zahlen frei verfügbar.

Alice verwendet β und Bob α für die weiteren Berechnungen:
$k_{Alice} = \beta^{\,a} \bmod p$ und $k_{Bob} = \alpha^{\,b} \bmod p$.
Die von Alice und Bob berechneten Zahlen k sind – wie man leicht zeigen kann – gleich und können im Weiteren als Schlüssel für ein beliebiges symmetrisches Verfahren verwendet werden.

Erwartungshorizont

<table>
<tr><th></th><th>Lösungsskizze</th><th colspan="3">Zuordnung, Bewertung</th></tr>
<tr><th></th><th></th><th>I</th><th>II</th><th>III</th></tr>
<tr><td>a)</td><td>Protokolle in der Telekommunikation und Informatik sind Regeln, welche das Format, den Inhalt, die Bedeutung und die Reihenfolge gesendeter Nachrichten zwischen verschiedenen Instanzen (der gleichen Schicht) festlegen. Diese Protokolle regeln den Ablauf, und stellen gleichzeitig dessen Dokumentation sicher.

Mit HELO und QUIT wird die Sitzung gestartet bzw. beendet. Die Kommandos FROM (Adresse des Senders), RCPT (Adressen der Empfänger), DATA und der einzelne Punkt müssen in dieser Reihenfolge erscheinen. Der Bereich zwischen DATA und dem einzelnen Punkt umfasst dann die eigentliche E-Mail. Bei allen Kommandos reagiert der Server mit einer Bestätigung (z.B. 250 Ok) oder einer (hier im Beispiel nicht auftretenden) Fehlermeldung.</td><td>3</td><td>4</td><td></td></tr>
<tr><td>b)</td><td>Bei symmetrischen Verschlüsselungsverfahren wird für das Entschlüsseln entweder derselbe Schlüssel verwendet wie für das Verschlüsseln bzw. ein Entschlüsselungsschlüssel, der aus dem Verschlüsselungsschlüssel auf einfache Art bestimmt werden kann. Bei asymmetrischen Verschlüsselungsverfahren werden unterschiedliche Schlüssel für das Ver- und Entschlüsseln verwendet. Von dem Verschlüsselungsschlüssel kann ohne zusätzliche Informationen nicht / sehr schwer auf den Entschlüsselungsschlüssel geschlossen werden.</td><td>6</td><td></td><td></td></tr>
<tr><td>c)</td><td><table><tr><th>Person 1 (Alice)</th><th>Person 2 (Bob)</th></tr><tr><td>$\alpha = 11^3 \bmod 23 = 20$
$k_{Alice} = 6^3 \bmod 23 = 9$</td><td>$\beta = 11^2 \bmod 23 = 6$
$k_{Bob} = 20^2 \bmod 23 = 9$</td></tr></table>
Beide Personen kommen zu demselben Ergebnis.</td><td></td><td>6</td><td>3</td></tr>
<tr><td>d)</td><td>Bei dem Verfahren wird nicht der eigentliche Schlüssel zwischen den beteiligten Personen ausgetauscht, sondern nur Zahlen, mit denen dann jede Person getrennt den gemeinsamen Schlüssel bestimmen kann.</td><td></td><td></td><td>4</td></tr>
<tr><td>e)</td><td>Die Sicherheit wird über große Primzahlen für p bzw. p und q hergestellt.
Nach dem derzeitigen Kenntnisstand sind diskrete Exponentialfunktionen (Schlüsselaustausch und Ver- / Entschlüsselung beim RSA-Verfahren) und die Faktorisierung (Schlüsselbestimmung beim RSA-Verfahren) Einwegfunktionen.

Bei der Verwendung großer Primzahlen ist der Versuch, die gesuchte Information beispielsweise durch Ausprobieren herauszubekommen, mit einem vertretbaren Aufwand nicht / sehr schwer zu erreichen.</td><td>2</td><td>4</td><td></td></tr>
<tr><td>f)</td><td>Vertraulichkeit: Die Nachricht wird so verändert, dass nur der berechtigte Empfänger in der Lage ist, die Nachricht zu verstehen.
Integrität: Die Nachricht kann nicht unbemerkt von einem Dritten verändert werden.

Authentizität: Die Nachricht stammt eindeutig von einem bestimmten Teilnehmer.</td><td>4</td><td>2</td><td></td></tr>
</table>

	Lösungsskizze	Zuordnung, Bewertung		
		I	II	III
g)	Vorschlag A setzt nur voraus, dass der Bewerber ein eigenes E-Mail-Konto besitzt. Bei Vorschlag B, C und D muss der Bewerber Programme zum Versenden von E-Mails benutzen, bei denen die Verschlüsselung mit Hilfe von DES bzw. RSA möglich ist. Bei Vorschlag D kommt hinzu, dass der Bewerber selbst ein über ein Schlüsselpaar verfügt. Vorschlag A bietet keinen Schutz der übertragenen Daten. Vorschlag B verwendet ein schnelles Verschlüsselungsverfahren. Aufwändig ist der Schlüsselaustausch mit dem Sekretariat. Vertraulichkeit, Integrität und Authentizität sind gewährleistet. Vorschlag C verwendet ein langsames Verschlüsselungsverfahren. Der Bewerber muss keinen weiteren direkten Kontakt mit dem Unternehmen herstellen. Die Authentizität des öffentlichen Schlüssels hängt davon ab, wie gut die Homepage des Unternehmens gegen unberechtigte Veränderungen gesichert ist. Vertraulichkeit ist gewährleistet. Vorschlag D verwendet ein (langsames) Verschlüsselungsverfahren zur Signierung. Die Firma muss den öffentlichen Schlüssel des Bewerbers verwenden. Integrität und Authentizität sind gewährleistet. Die Entscheidung für einen der Vorschläge B, C und D muss gut begründet werden, insbesondere die Abwägung zwischen den jeweiligen Vor- und Nachteilen. Je nach den verwendeten Kriterien für die Bewertung können auch alle Vorschläge als ungeeignet bewertet werden. Dann muss ein entsprechend qualifizierter eigener Vorschlag entwickelt werden.		6	6
	Insgesamt 50 BWE	15	22	13

4.1 grundlegendes Anforderungsniveau

Aufgabe III

Simulation

Beschreibung eines Systems

Das hier zu untersuchende Ökosystem beschreibt ein reales System. Um größere Weideflächen für Schafe zu schaffen, hatte man in New South Wales [Australien] vorhandene Eukalyptuswälder gelichtet. Man achtete darauf, etwa 20 % des Waldbestandes zu erhalten, um eine Versteppung des schon vorhandenen und nun zusätzlich entstehenden Graslandes zu verhindern.
Das System war durch die umliegenden Regionen weitgehend abgeschlossen und die insgesamt von ihm ausgefüllte Fläche allein von Wald oder Grasland bedeckt.
Weiterhin sind zwei für das System wichtige Tierpopulationen vorhanden, die beide sowohl auf das Grasland als auch auf den Wald angewiesen sind. Eine der Tierpopulationen ist eine Insektenpopulation, die das Grasland im Larvenstadium benötigt und sich im Erwachsenenstadium von den Bäumen ernährt. Die andere Tierpopulation ist eine Vogelpopulation, die zum Nisten die Bäume benötigt und sich von den Insekten ernährt.

a) Stellen Sie zu den in den drei folgenden Teilaufgaben jeweils beschriebenen, voneinander unabhängigen Wachstumsvorgängen je ein Wirkungsdiagramm oder ein Flüssediagramm dar und skizzieren Sie den jeweiligen Verlauf des Zeitdiagramms.

   Geben Sie an, welche Wachstumsform jeweils vorliegt, und begründen Sie.

   - Die Biomasse der Bäume wächst jährlich mit einer konstanten Rate.
   - Die Insekten vermehren sich wöchentlich um ein Zehntel (10 %) ihrer Biomasse.
   - Die Vögel vermehren sich jährlich entsprechend ihrer vorhandenen Biomasse und der noch zur Verfügung stehenden Kapazität, die durch eine Maximalzahl begrenzt ist. (15P)

Der Zuwachs der Biomasse der Bäume wird in der Regel von der vorhandenen Biomasse der Bäume abhängen, gleichzeitig aber durch die zur Verfügung stehende Fläche begrenzt sein. Weiterhin soll der Wald bewirtschaftet werden, indem Teile abgeholzt werden. Die durch Abholzen frei werdende Fläche wird von Grasland eingenommen.

b) Dieses eingeschränkte Szenario kann durch die folgenden Systemgleichungen beschrieben werden:

Zustandsgleichungen[1]
Wald_Biomasse.neu = Wald_Biomasse.alt + dt*(Zuwachs_Wald-Abholzung)
Startwert Wald_Biomasse = 20

Zustandsänderungen
Zuwachs_Wald = ZuwachsRate_Wald*Wald_Biomasse*(Max_Biomasse_Wald-Wald_Biomasse)/Max_Biomasse_Wald
Abholzung = AbholzRate

Parameter
AbholzRate = 2
ZuwachsRate_Wald = 0,1
Max_Biomasse_Wald = 100

Erläutern Sie die angegebenen Systemgleichungen. (5P)

1 Consideo: 20.0 + ∫ - [Abholzung] + [Zuwachs_Wald] dt
In den folgenden Gleichungen jeweils entsprechend

c) Beschreiben Sie die Veränderung der Biomasse des Waldes in den beiden folgenden Diagrammen, die sich allein in der unterschiedlichen Biomasse der jährlichen Abholzung unterscheiden und erklären Sie, weshalb es zu den unterschiedlichen Reaktionen kommt. (5P)

[Consideo-Bilder in der Anlage]

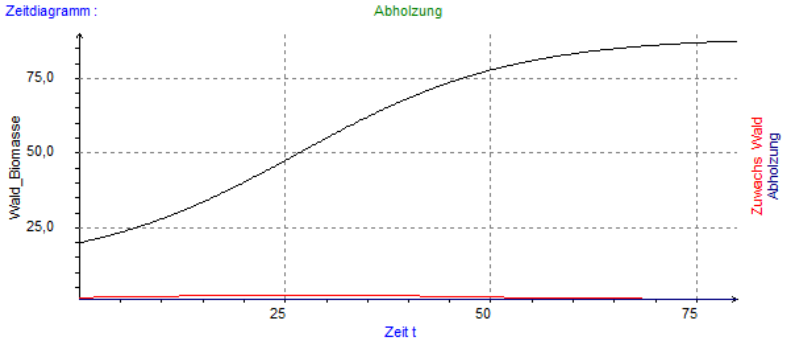


Bild 1

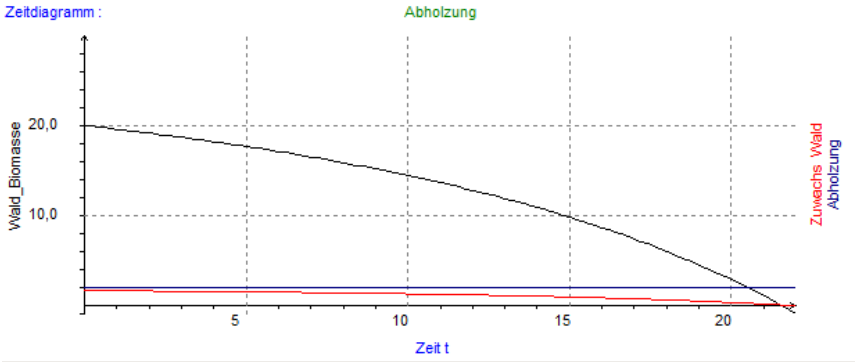
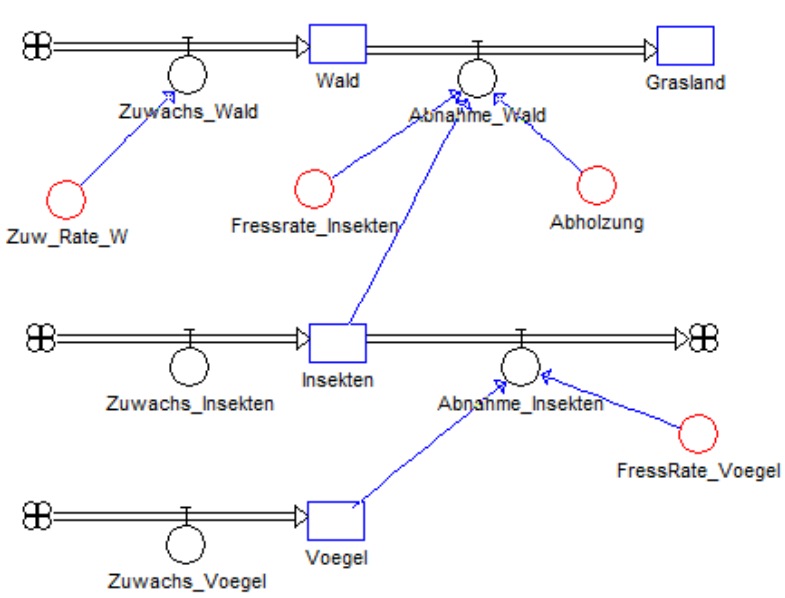


Bild 2

d) Im Gegensatz zu der Annahme der Planer in New South Wales entwickelte sich das Ökosystem nicht wie erwartet, denn der restliche Waldbestand brach nach kurzer Zeit durch den Befall mit Schadinsekten zusammen.

Das nebenstehende Flüssediagramm zeigt einen Vorschlag zu einer Modellierung des Systems, mit der dieses nicht erwartete Verhalten des Systems simuliert werden sollte.



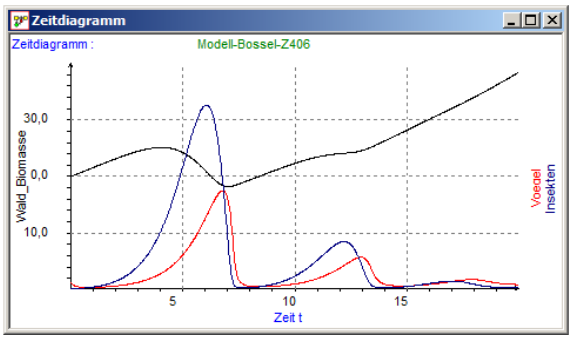
Untersuchen Sie die zu Grunde liegende Modellierung auf Vor- und Nachteile. Gehen Sie dabei auch auf die zu erwartenden Wachstumsformen ein. (8P)

e) *Erläutern* Sie, welche Änderungen Sie für eine vollständige Modellierung des Systems für notwendig halten und *begründen* Sie.

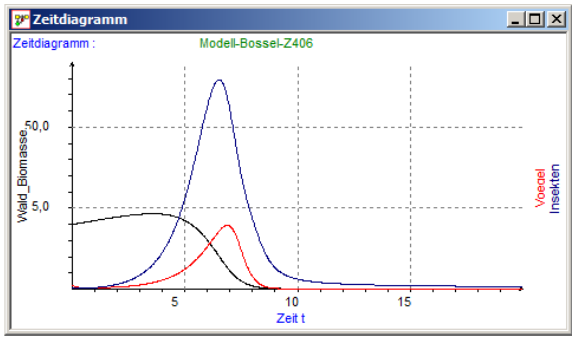
Sie können zur grafischen Darstellung ihrer Veränderungen statt eines Simulationsdiagramms auch ein Wirkungsdiagramm verwenden. (7P)

f) Die folgenden Diagramme zeigen zwei Zeitdiagramme einer anderen Modellierung des Systems bei unterschiedlich hohen Abholzungsraten.

Beschreiben Sie die Unterschiede und die daraus resultierenden Auswirkungen auf das Ökosystem und begründen Sie aus der Beschreibung des Systems heraus, dass die Abholzungsrate zu den beobachteten Veränderungen führen kann. (5P)



niedrige Abholzungsrate



hohe Abholzungsrate

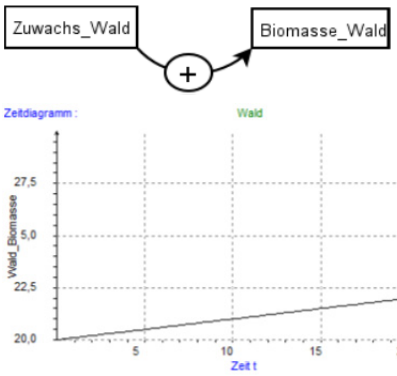
g) *Entwickeln* Sie eine Empfehlung an die Bevölkerung der Region, die ihren Lebensunterhalt teilweise aus der Holzwirtschaft bezieht. (5P)

Erwartungshorizont

	Lösungsskizze	Zuordnung, Bewertung		
		I	II	III
a)				

- Es handelt sich um lineares Wachstum, da der Zuwachs konstant und nicht vom Bestand abhängig ist. Die grafische Darstellung muss eine lineare Funktion darstellen.

  Diagramme (Dynasys):



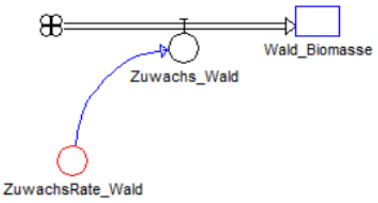
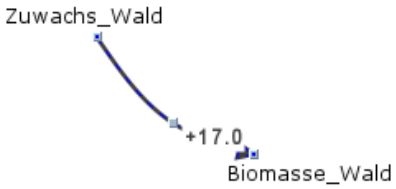
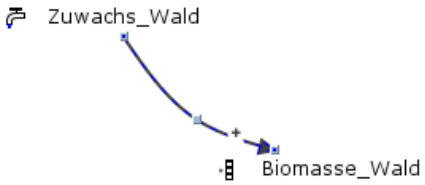


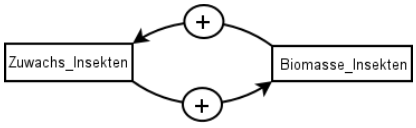
  Diagramme (Consideo):

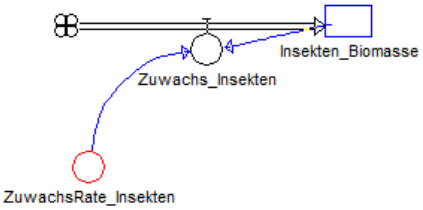


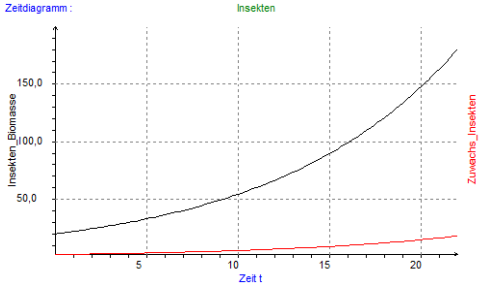


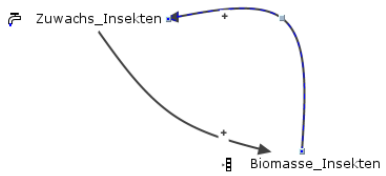
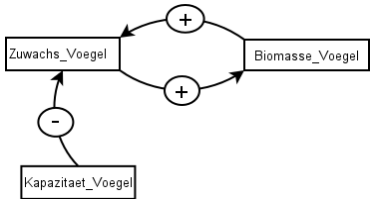
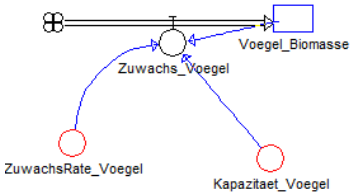
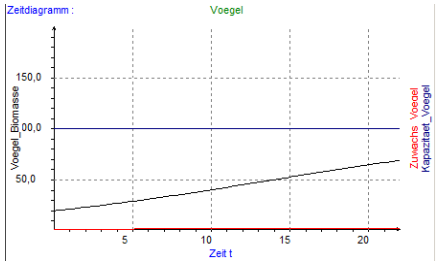
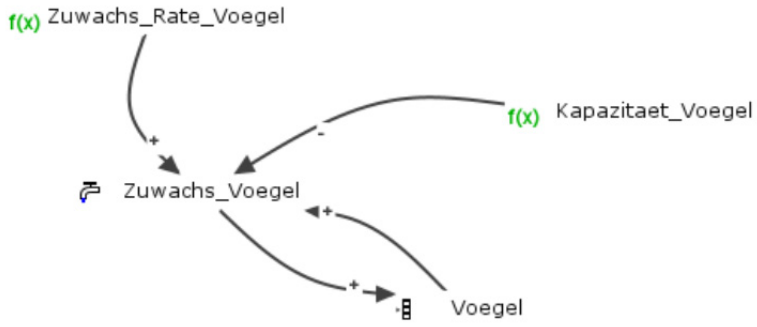
- Es handelt sich um exponentielles Wachstum, da der Zuwachs proportional zum Bestand ist. In der grafischen Darstellung muss der steigende Zuwachs erkennbar sein.

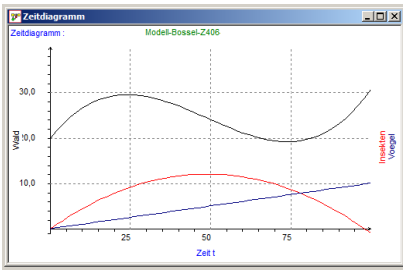
  Diagramme (Dynasys):







	Lösungsskizze	Zuordnung, Bewertung		
		I	II	III
	Diagramme (Consideo):  • Es handelt sich um logistisches Wachstum, da der Zuwachs sowohl vom Bestand als auch von der (freien) Kapazität abhängig ist. In der grafischen Darstellung muss der typische Verlauf eines logistischen Wachstums erkennbar sein. Diagramme (Dynasys):    Diagramme (Consideo):  	8	7	
b)	Die Größe Biomasse_Wald ist ein Bestand [Bestandsfaktor], dessen Wert durch Aufsummieren der Änderungen schrittweise berechnet wird. Zustandsänderungen treten durch die Flüsse Zuwachs_Wald und Abholzung auf. Der Zuwachs_Wald zeigt die typischen Gleichungen für logistisches Wachstum, da er proportional nicht nur zu einem Faktor, sondern auch zum Bestandswert und zur freien Kapazität ist. [Der zusätzliche Teiler dient der Normierung.] Die Abholzung wird als konstant angenommen. Die Parameter definieren die für das System relevanten konstanten Werte.	2	3	

	Lösungsskizze	Zuordnung, Bewertung		
		I	II	III
c)	Die erste Kurve (Bild 1) zeigt den typischen Verlauf für logistisches Wachstum, während die zweite (Bild 2) eine reine exponentielle Abnahme zeigt. Ursache ist, dass bei der zweiten Kurve die Abholzung größer ist als der anfängliche Zuwachs, so dass es gar nicht erst zu Wachstum kommen kann.	3	2	
d)	Die im Flüssediagramm dargestellte Modellierung stellt die auftretenden Bestandsgrößen und Flüsse dar und zeigt auftretende Wirkungen. Es ist sehr einfach und damit leicht zu verstehen. [Hier können die Schülerinnen und Schüler die auftretenden Bestandsgrößen, Flüsse und Wirkungen beschreiben und erreichen damit Leistungen im Bereich I.]  Man erkennt, dass die Zuwächse und Abnahmen jeweils unabhängig von ihrer Bestandsgröße sind, so dass für alle Bestandsgrößen lineares Wachstum modelliert wird [Bereich II]. Für Leistungen im Bereich III müssen sie das als Mangel erkennen und erläutern. Außerdem können sie schon an dieser Stelle auf die fehlenden Wechselwirkungen zwischen den Populationen eingehen. [Im vorliegenden Diagramm taucht jeweils nur eine Richtung auf.] [Das hier dargestellte, den Schülerinnen und Schülern nicht bekannte und von ihnen nicht erwartete Zeitdiagramm zeigt dennoch nicht nur lineare Verläufe, was daran liegt, dass durch die Wirkung von der Bestandsgröße der Vögel auf die Abnahme der Insekten und von der Bestandsgröße der Insekten auf die Abnahme des Waldes eine zweistufige Integratorenkette auftritt, die zu einem quadratischen Verlauf bei den Insekten und zu einem kubischen Verlauf beim Wald führt. Diese Erkenntnis kann von den Schülerinnen und Schülern nicht erwartet werden. Ist das dennoch der Fall, stellt das eine zusätzliche Leistung im Bereich III dar, die entsprechende fehlende Leistungen in der nachfolgend beschriebenen Lösung zu Teilaufgabe g) ersetzen kann. Weiterhin können die Schülerinnen und Schüler an dieser Stelle schon auf die Frage eingehen, ob das Grasland als eigene Größe sinnvoll ist. Dazu s.u.]	2	3	3
e)	Unabhängig davon, ob die Schülerinnen und Schüler in der vorigen Teilaufgabe bei der Bewertung darauf hinweisen, dass es unrealistisch ist, dass der Zuwachs unabhängig vom Bestandswert ist [s.u.], sollen sie in dieser Teilaufgabe zunächst Wirkungen nennen, die zwar in der Beschreibung des Systems auftauchen, im Modell aber nicht. Das ist einmal die Aussage, dass beide Tierpopulationen sowohl auf den Wald als auch auf das Grasland angewiesen sind. Daher muss es von den Bestands-größen Wald und Grasland Wirkungspfeile zu den Flüssen bei den Vögeln und Insekten geben. In den nachfolgenden Sätzen der Systembeschreibung sind diese Wirkungen außerdem vollständiger beschrieben. [Bereich II] Darüber hinausgehend sollen die Schülerinnen und Schüler darauf hinweisen, dass prinzipiell bei biologischen Systemen nicht davon ausgegangen werden kann, dass der Zuwachs unabhängig vom Bestandswert ist. Beim Wald und bei den Tierpopulationen ist davon auszugehen, dass der Zuwachs einerseits proportional zum Bestandswert ist [Bereich II], andererseits sind biologische Systeme in der Regel auch durch eine Kapazität des Systems beschränkt [Bereich III].			

	Lösungsskizze	Zuordnung, Bewertung		
		I	II	III
	Für eine weitergehende Wertung im Bereich III müsste zumindest beim Wald angeführt werden, dass er durch die zur Verfügung stehende Fläche begrenzt ist. Dass es zudem eine [dynamisch veränderliche] Kapazität für den Zuwachs bei Vögeln und Insekten gibt, ergibt die Möglichkeit auf das Fehlen der entsprechenden Kapazitäten und Wirkungen im vorliegenden Modell hinzuweisen und damit die volle Wertung im Bereich III zu erreichen. Fehlende Leistungen im Bereich III können hier teilweise durch die Erkenntnis ersetzt werden, dass das Grasland nicht als eigene Größe modelliert werden muss, sondern aus der Differenz zwischen der zur Verfügung stehenden Fläche und der mit Wald bestandenen Fläche berechnet werden kann.		3	4
f)	Das zweite der beiden Diagramme zeigt das in der Problembeschreibung angegebene tatsächliche Verhalten des Systems: Bei hoher Abholzungsrate bricht der Waldbestand durch den steilen Anstieg der Insektenpopulation zusammen. Das hat die Folge, dass auch die Insektenpopulation und schließlich die Vogelpopulation zusammenbrechen. Bei niedrigerer Abholzungsrate tritt zwar auch ein starker Anstieg der beiden Tierpopulationen auf, der aber nicht zu einem vollständigen Zusammenbruch des Waldbestandes führt. Alle Populationen können sich nach der starken Abnahme wieder erholen und auftretende Schwankungen werden vom System kompensiert.		2	3
g)	Die Empfehlung an die Bevölkerung geht hin zu einer nachhaltigen Forstwirtschaft, bei der die Abholzung so niedrig gehalten wird, dass das System sich selbst dann erholen kann, wenn es zu einer explosionsartigen Vermehrung der Insekten kommt.		2	3
	Insgesamt 50 BWE	15	22	13

Erläuterungen zur Aufgabenstellung

"Der Zustand von Ökosystemen bestimmt sich durch die komplexen Verknüpfungen zwischen ihren Komponenten, die sich im Laufe der evolutionären Entwicklung herausgebildet haben. Meist sind es vielseitige Abhängigkeitsbeziehungen zwischen Organismen über Nährstoffkreisläufe, Nahrungsketten und Nahrungsnetze, Räuber- Beute-Systeme, Symbiosen, Bestäubung, Samen Verteilung und viele andere Prozesse. Die interagierenden dynamischen Prozesse kontrollieren und regeln sich gegenseitig, so dass sich ein für das jeweilige Ökosystem typisches dynamisches Gleichgewicht herausbildet. Eingriffe, die einzelne Komponenten besonders beeinträchtigen oder fördern, können daher zu einem Umkippen des Systems in einen anderen Zustand führen.

Ein solcher Vorgang wurde z.B. in Australien beobachtet und von Trenbath und Smith 1981 als Simulationsmodell dargestellt (Richter 1985). Um größere Weideflächen für Schafe zu schaffen, hatte man in New South Wales vorhandene Eukalyptuswälder gelichtet. Zwar achtete man darauf, etwa 20% des Waldbestandes zu erhalten, um eine Versteppung des Graslandes zu verhindern, aber dennoch brach der restliche Waldbestand nach kurzer Zeit durch den Befall mit Schadinsekten zusammen.

Für die katastrophale Vermehrung der Schadinsekten und den Zusammenbruch des Restwaldes wurden zwei Gründe vermutet: Durch die Vergrößerung der Weidefläche wurden erstens die Lebensbedingungen der Insektenlarven, die sich von Graswurzeln ernähren, verbessert. Zweitens wurde aber auch durch die Verringerung der Nistplätze für Vögel die Population dieser Fressfeinde der Insekten verringert.

Das Modell beschreibt daher die folgenden Zusammenhänge: Eine Region mit einer maximalen Biomassekapazität K besteht zum Teil aus Wald x, zum Teil aus Graslandvegetation (K-x). Vögel brauchen den Wald für Nistplätze und ernähren sich von Insekten. Insekten benötigen den Wald als Futterquelle und das Grasland für das Aufwachsen der Larven. Wird der Wald zunehmend zerstört, so

verschlechtern sich die Bedingungen für die Vögel und verbessern sich für die Insekten. Ab einem gewissen Stadium nehmen die Insekten überhand und zerstören den restlichen Wald."

Hinweise zur Aufgabe und Quelle

Die Aufgabe basiert auf einem Modell von Hartmut Bossel, das in seinem Heft Systemzoo 2 zu finden ist. Es ist das Modell Z406 Vögel, Insekten, Wald und Grasland.

Hartmut Bossel: Systemzoo 2; Klima, Ökosysteme und Ressourcen; ISBN 3-8334-1240-2

Leider macht das Lösungs-Modell bei geringem Einschlag sehr langfristig nicht das, was Bossel beschreibt. Dies hat Bossel möglicherweise übersehen, weil er nicht langfristig genug simuliert hat. Das sollte für die Schülerinnen und Schüler aber kein Problem darstellen.

3.2 erhöhtes Anforderungsniveau

Aufgabe I

Objektorientierte Modellierung und Programmierung

Hausbau

Es soll ein einfaches CAD-Programm entwickelt werden, das bei der Planung des Baus von Ein- und Zweifamilienhäusern aus Fertigbauelementen hilft. Dieses Programm soll Käufern zur Verfügung gestellt werden, damit sie sich ihr Traumhaus planen können. Es soll erst einmal nur der Grundriss gezeichnet werden können. Dabei müssen die Bauherren jeweils aus einem bestehenden Katalog vorgegebener Elemente auswählen. In dem Katalog sind die Fertigbauelemente enthalten. Es gibt beispielsweise mehrere Außen- und Innenwände, Türen, Fenster und Treppen. Da die Wandelemente auch transportiert werden müssen, gibt es sie in 5 Längen zwischen 2 m und 4 m. Längere Wände müssen aus Teilwänden zusammengesetzt werden.

Die Skizzen in der Anlage zeigen den Grundriss einer zweigeschossigen Wohnung. Es soll mit dieser Skizze nur gezeigt werden, wie beispielsweise Treppen, Türen, Wände und Fenster dargestellt werden.

a) **Beschreiben** Sie typische Interaktionen des Anwenders mit dem Programm. Entwickeln Sie daraus grundlegende Anforderungen an das Programm. (8P)

b) **Geben** Sie Kriterien an, wann der Einsatz von Vererbung bei der objektorientierten Modellierung angemessen ist. Beschreiben Sie eine mögliche Alternative zur Vererbung. (9P)

c) **Beschreiben** Sie in kurzen Aussagensätzen die wichtigen Teile einen Hauses und geben sie dabei ihre Beziehung an. Entwickeln Sie für dieses System ein Klassendiagramm. Geben Sie wesentliche Attribute und Methoden an. Begründen Sie ausführlich Ihre Entscheidungen bei der Modellierung. (10P)

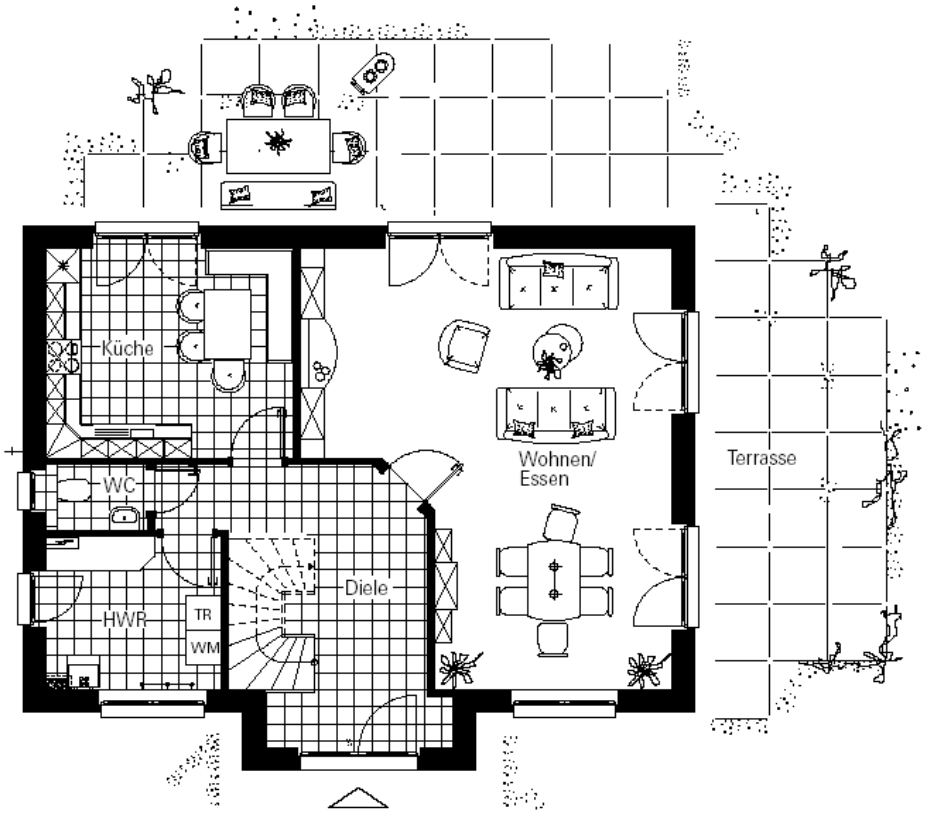
d) Alle Elemente eines Geschosses müssen gemeinsam verwaltet werden. In Java gibt es dafür mehrere Möglichkeiten. **Beschreiben** und **vergleichen** Sie drei von diesen. (10P)

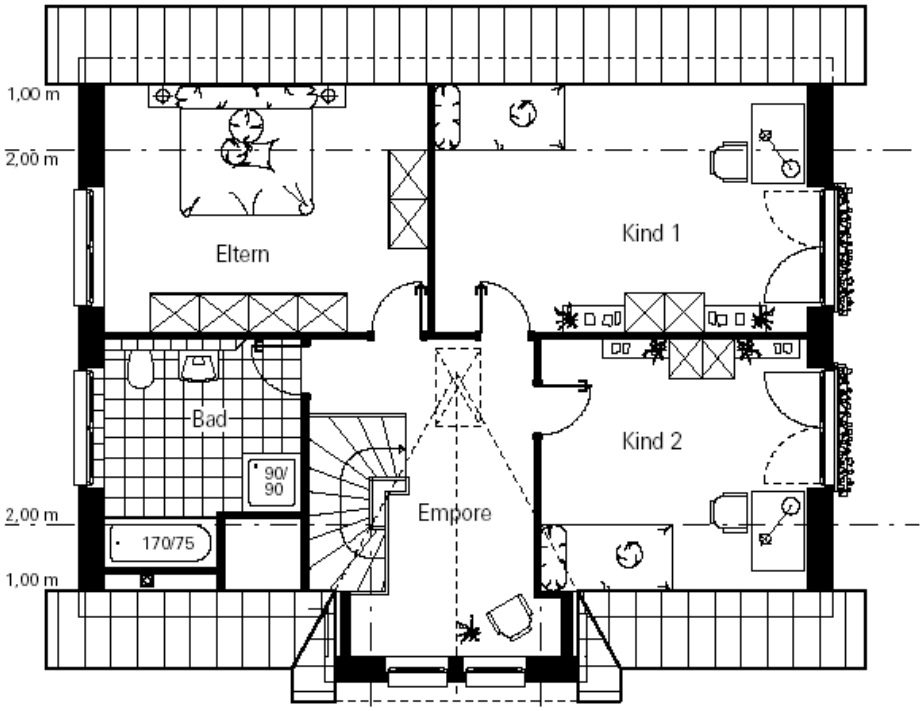
e) Gehen Sie jetzt davon aus, dass alle Elemente eines Geschosses mit Hilfe einer ArrayList verwaltet werden und jedes Element über eine Methode gibPreis() verfügt.
Implementieren Sie eine Methode gibGesamtpreis und erläutern Sie Ihre Implementation. (7P)

f) Eine Treppe gehört jeweils zu zwei Geschossen. Nun wird die Treppe im Erdgeschoss verschoben. Dieses muss im Obergeschoss berücksichtigt werden.
Entwickeln Sie einen Lösungsansatz für dieses Problem unter Verwendung von Nachrichten zwischen den beteiligten Instanzen. (6P)

Hilfsmittel: Dokumentationen von Klassenbibliotheken

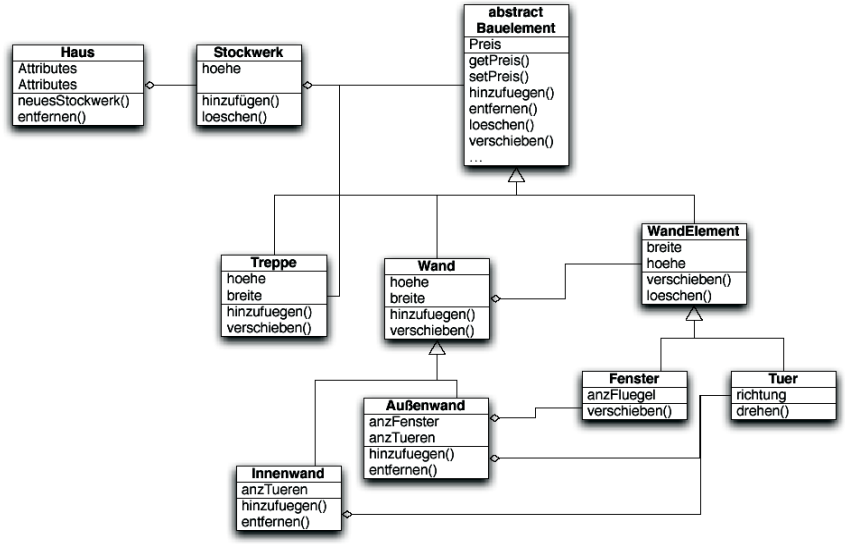
Anlage zur Aufgabe „Hausbau“





Erwartungshorizont

	Lösungsskizze	Zuordnung, Bewertung		
		I	II	III
a)	Der Anwender erstellt Geschosse, wählt, positioniert, verschiebt, dreht, löscht Bauelemente, er lässt den Plan zeichnen, er lässt dem Preis des Hauses kalkulieren, er speichert und öffnet Varianten. Das Programm muss den Plan speichern und öffnen können. Objekte müssen markiert, Eigenschaften müssen geändert werden können. Es muss eine Kalkulation erstellt werden können. Elemente müssen aus Listen ausgewählt werden können.	4	4	
b)	Ausgehend von den konkreten Klassen muss es möglich sein, einen allgemeinen Fall abzuleiten. Es müssen Klassen vorliegen, die auf gleiche Nachrichten reagieren, deren zugehörige Methoden aber verschieden sind. Zwischen den Klassen muss ein Merkmal angegeben werden können, das die Klassen unterscheidbar macht. Da jetzt gleiche Methoden in die Super-Klasse ausgelagert werden können, wird Redundanz vermieden. Neben der Vererbung muss Delegation als Alternative in Betracht gezogen werden. Bei diesem Konstrukt benutzt eine Klasse eine andere Klasse. Delegation führt in der Client-Klasse zu sehr kurzen Methoden, da die Arbeit an ein anderes Objekt einer i. d. R anderen Klasse übergeben wird. Vererbung darf nur eingesetzt werden, wenn die Sub-Klassen Spezialfälle der Super-Klasse sind. Aber bei Erweiterung ist die Gefahr sehr groß, zu einer inkonsistenten Hierarchie zu kommen. Vererbung ist nur erlaubt, wenn gilt: Klasse A ist Spezialisierung von Klasse B. Dies ist aber nicht gegeben. Dabei gilt Spezialisierung nur in der Weise, dass es zu einer Erweiterung kommt, es darf keine Einengung vorliegen. (Kreis-Ellipsen-Dilemma, Bertrand Meyer) Vererbung setzt Abstraktion voraus. Damit wird es möglich, nicht auf den konkreten Details zu programmieren, um von den Änderungen an kapselbaren Entwurfsentscheidungen wenig betroffen zu sein. – Lose Kopplung)	2	4	3
c)	Da es sich um eine offene Aufgabenstellung handelt kann hier nur eine mögliche Lösung angegeben werden, andere sind wahrscheinlich. So ist es durchaus möglich, dass die Räume beispielsweise von ihrer Nutzung her betrachtet werden, dann könnte es eine übergeordnete Klasse Raum und davon abgeleitet Funktionsräume geben. Es ist denkbar beispielsweise das Factory-Entwurfsmuster zu verwenden. Wichtig ist, dass die Beschreibung des Hauses und das sich daraus ergebende Modell konsistent ist. Das Modell muss mehrere Klassen enthalten. Beziehungen müssen sinnvoll sein. Vererbung muss korrekt genutzt werden (siehe Antwort b.). Eine mögliche Lösung: Das Haus hat zwei Stockwerke, in jedem Stockwerk gibt es mehrere Wände, eine Wand kann eine Außenwand und Innenwand sein. Eine Innenwand kann nur Türen enthalten, Außenwände enthalten Fenster und Türen. Innen- bzw. Außenwand sind ein Spezialfall einer allgemeinen (abstrakten) Wand. Fenster und Türen sind in Wänden enthalten. Wände, Fenster und Türen sind Bauelemente. Hier ist ein übergeordnetes abstraktes Element gegeben. So können später beispielsweise leichter andere Elemente wie eine Durchreiche ergänzt werden.			

	Lösungsskizze	Zuordnung, Bewertung		
		I	II	III
			5	5
d)	Es könnte das Array oder eine Collection: LinkedList, ArrayList, Vector verwendet werden. Ein Array vom Typ BauElement ist eine einfache Form, aber statisch. Das Programm legt die maximale Anzahl von Elementen fest. Der Zugriff auf die einzelnen Elemente des Feldes erfolgt über einen Index. In Arrays können auch einfache Datentypen gespeichert werden. – Hier haben sie auch ihre Berechtigung. Für Klassen sollte eine Collection eingesetzt werden. Collections sind flexibler. Collections können nur Objekte enthalten. Einfache Datentypen müssten mit einem »Wrapper« eingepackt werden. Die Anzahl der enthaltenen Elemente passt sich während der Programmlaufzeit an die Erfordernisse an. Der Zugriff auf die Elemente der Collection erfolgt über einen Iterator. Beide Elemente sind in der Java-Bibliothek enthalten. Eine LinkedList könnte die Reihenfolge aneinander gefügter Elemente wiedergeben.	10		
e)	Der GesamtPreis kann wie folgt kalkuliert werden: `import java.util.*;` `public class Geschoss` `{` `  private ArrayList bauElemente;` `  public double gibGesamtpreis()` `  {` `  double gesamtPreis = 0.0;` `  BauElement tmpBauElement;` `  ListIterator<BauElement> it = bauElemente.listIterator();` `  while(it.hasPrevious())` `  {` `  tmpBauElement = it.previous();` `  gesamtPreis += tmpBauElement.getPreis();` `  }` `  return gesamtPreis;` `  }` `}` Die Prüflinge können anstelle des Iterators auch eine for-Schleife nutzen.		5	2

	Lösungsskizze	Zuordnung, Bewertung		
		I	II	III
f)	Die Stockwerke müssen eine Referenz auf das über bzw. unter ihnen liegende Stockwerk besitzen. Nur so können sie sich gegenseitig benachrichtigen. Die Verschiebung der Treppe hat dann in dem einen Stock zur Aufgabe, die Nachricht an das andere Stockwerk zu senden. Damit würde man ein Beobachtermuster nutzen. Es hängt aber davon ab, wie die Koordinatensysteme und die Bezüge im Haus realisiert sind. Es ist zwar auch möglich, einen übergeordneten Verwalter zu verwenden. Dies ist aber keine gute Lösung. Es werden zusätzliche Abhängigkeiten geschaffen.		3	3
	Insgesamt 50 BWE	16	21	13

Hinweise zur Aufgabe:

Die Aufgabe setzt voraus, dass sich die Prüflinge die im Rahmenplan genannten Konzepte der objektorientierten Modellierung und Programmierung erarbeitet haben.

Dazu gehört u. a. die Erstellung von Use-Case- und Klassendiagrammen (UML), die Eigenschaften verschiedenartiger Beziehungen zwischen Klassen sowie die Konzepte der Kapselung, der Vererbung und der Polymorphie. Ein Schwerpunkt stellte die Beurteilung wichtiger Modellierungsalternativen dar, dazu gehört beispielsweise die Alternative Vererbung / Delagation. Modelle sind mit Hilfe Java implementiert und Klassenbibliotheken genutzt worden. In Java wurden die einfachen Datentypen, der strukturierte Datentyp Array und einfache Konzepte der Collections behandelt. Darüber hinaus wurden einige Entwurfsmuster, u.a. das Kompositum erarbeitet.

Im Zusammenhang mit der Implementation von Methoden für Grafikobjekte wurde das Drehen behandelt. Bezüglich der Aufgabenstellung d) wurden mindestens drei Varianten im Unterricht diskutiert.

Kommentar

Es ist sinnvoll, die Bewertung des Anwendungskontextes als weiteren Aufgabenteil zu ergänzen. Dazu müsste ein aktueller „authentischer“ Text vorgeben werden (z. B. von der Architektenkammer über Selbstplanung von Häusern), zu dem die Prüflinge Stellung nehmen müssten.

3.2 erhöhtes Anforderungsniveau

Aufgabe II

Sprachverarbeitung

Einsatz automatisierter Übersetzer bei Bewerbungsschreiben

Maxi Mustermann hat endlich ihren Bachelor in Informatik gemacht und möchte nun ihre erste Bewerbung schreiben. Die Firma hat ihren Hauptsitz in England und Maxi muss das Anschreiben in Englisch verfassen. Leider kann Maxi in Englisch keine längeren Texte fehlerfrei schreiben. Daher probiert sie automatische Übersetzer aus, die ihr diese Arbeit abnehmen sollen.
Folgende Bewerbung hat Maxi in Deutsch geschrieben (nur der Anfang ist hier abgedruckt):

> *Bewerbung auf Ihre Anzeige "Junge Systementwickler gesucht"*
> *Sehr geehrter Herr Maier,*
> *in den Salzburger Nachrichten las ich, dass Sie zum 15. Mai 2002 eine junge Systementwicklerin mit der Aufgabe einstellen wollen, Systeme zur laufenden Anpassung des internen Großrechners an die Bedürfnisse der Marketing-Spezialisten zu entwickeln. Ich bewerbe mich bei Ihnen, weil ich glaube, die dafür notwendigen Voraussetzungen mitzubringen.*
> *Nach dem Abitur studierte ich an der Universität Salzburg Informatik. Ich lernte in den ersten vier Semestern die Grundlagen des Programmierens. Anschließend verbrachte ich zwei äußerst interessante Auslandssemester an der Eidgenössischen Technischen Hochschule in Zürich, wo ich eine Vorliebe für kreative Systementwicklung entwickelte. Nach Salzburg zurückgekehrt, schloss ich mein Informatikstudium mit dem Bachelorthema "Die Probleme der Bedarfsabklärung bei Systemanpassungen" ab…..*

Der Google-Übersetzer hat folgende Übersetzung geliefert:

> *Application on your ad 'boy system developers wanted "*
> *Dear Mr. Maier,*
> *in the Salzburger Nachrichten, I read that you are 15 Want to set May 2002 a young system developer with the task of developing systems for the continuous adjustment of internal mainframe to the needs of marketers. I apply to you because I think to bring the necessary prerequisites.*
> *After high school I studied computer science at the University of Salzburg. I learned in the first four semesters of the basics of programming. Then I spent two very interesting semester abroad at the Swiss Federal Institute of Technology in Zurich, where I developed a penchant for creative development system. Back in Salzburg, I closed my computer science degree with a bachelor on "The problems of the needs assessment for system adjustments" from…*

Trotz ihrer schlechten Englischkenntnisse kann Maxi sofort erkennen, dass sie sich damit nicht bewerben kann. Sie probiert noch diverse andere Übersetzer aus, bekommt aber immer ähnlich schlechte Ergebnisse.

a) **Erläutern Sie,** welche Schwierigkeiten der Übersetzer offensichtlich bei der Übersetzung der Bewerbung hatte. (6P)

b) Als Informatikerin sagt sich Maxi, dass sie das doch bestimmt besser kann. Sie fängt an, sich in das Thema automatische Sprachverarbeitung einzuarbeiten.
Dabei entwickelt sie in Scheme zunächst einen einfachen Satzübersetzer und probiert ihn auf einem kleinen Lexikon aus:

```
(define *lexikon1*
  '((Erfolg  success)
    (feiern celebrate)
    (mit  with)
    (bei    at)
    (der  the)
    (die  the)
    (Aufgabe  exercise)
    (Test  test)))

(define
  (wortuebersetzereinfach  wort  lex)
  (cond
    ((null?  lex) '(Wort nicht im Lexikon))
    ((equal?  wort  (first  (first  lex)))
     (first  (rest  (first  lex))))
    (else
     (wortuebersetzereinfach  wort  (rest  lex)))))

(define
  (satzuebersetzungeinfach satz lex)
  (cond
    ((null?  satz) '())
    ((null?  lex)
     (cons
      '(Wort nicht im Lexikon)
      (satzuebersetzungeinfach (rest  satz)  *lexikon1*)))
    ((equal?  (first satz) (first (first lex)))
     (cons
      (wortuebersetzereinfach (first  satz)  *lexikon1*)
      (satzuebersetzungeinfach (rest  satz) *lexikon1*)))
    (else
     (satzuebersetzungeinfach  satz  (rest  lex)))))
```

Geben Sie an, welche Ausgabe folgender Aufruf erzeugt. **Skizzieren Sie** dazu auch den Ablauf der Rekursion bei diesem Aufruf:

```
(satzuebersetzungeinfach `(Erfolg mit dieser Aufgabe) *lexikon1*)
```
(9P)

c) Die oben angegebene Lösung der Satzübersetzung ist nicht optimal, da das Lexikon zweimal durchlaufen wird.
Entwickeln Sie eine bessere Lösung und **implementieren Sie** diese.
Erläutern Sie ihren Ansatz. (7P)

d) Maxis Übersetzer beachtet bisher keine Grammatik. Diese soll nun schrittweise berücksichtigt werden.
Zunächst möchte sie sich eine passende formale Grammatik mit einfachen Regeln aufbauen, die obiges Lexikon abbildet.
Folgendes Grundgerüst einer formalen Grammatik sei gegeben

S -> NP VP	VP -> V	N -> Aufgabe	
NP -> N	VP -> V NP		
NP -> ART N			

Geben Sie die Bestandteile einer formalen Grammatik an. (5P)

e) **Erweitern Sie** das Grundgerüst so, dass der Satz „Erfolg feiern mit der Aufgabe“ zu der von der Grammatik erzeugten Sprache gehört. (8P)

f) Bevor ein Satz korrekt übersetzt werden kann, muss er auf seine grammatische Korrektheit überprüft werden. Dies macht folgender Parser:

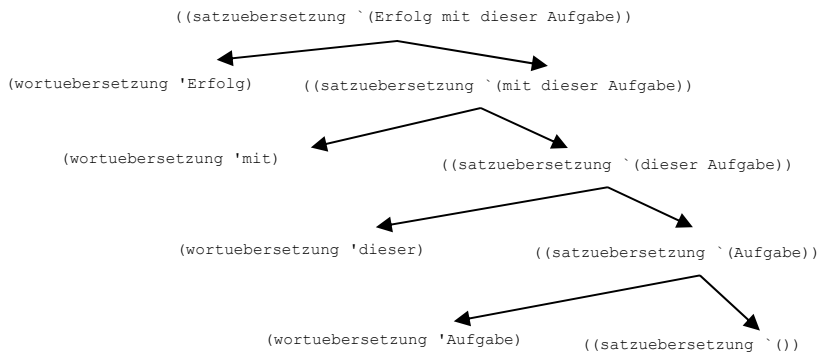
```
(define (start eingabe)
    (parse eingabe (list 'np 'vp)))

(define (parse eingabe keller)
  (cond
      ((and (null? eingabe) (null? keller)) #t)
      ((null? keller) #f)
      ((null? eingabe) #f)
      ((equal? (first keller) 'vp) (vp-fkt eingabe (rest keller)))
      ((equal? (first keller) 'np) (np-fkt eingabe (rest keller)))
      ((equal? (first keller) 'pp) (pp-fkt eingabe (rest keller)))
      ((equal? (first keller) 'n) (n-fkt eingabe (rest keller)))
      ((equal? (first keller) 'v) (v-fkt eingabe (rest keller)))
      ((equal? (first keller) 'p) (p-fkt eingabe (rest keller)))
      ((equal? (first keller) 'art) (art-fkt eingabe (rest keller)))
      ((equal? (first eingabe)(first keller)) (parse (rest eingabe)(rest keller)))
      (else #f)))

(define (vp-fkt eingabe keller)
    (or (parse eingabe (cons 'v keller))
        (parse eingabe (cons 'pp keller))
        (parse eingabe (append (list 'v 'np) keller))))

(define (np-fkt eingabe keller)
  (or (parse eingabe (cons 'n keller))
      (parse eingabe (cons 'pp keller))
      (parse eingabe (append (list 'art 'n) keller))))

(define (n-fkt eingabe keller)
  (or (parse eingabe (cons 'Raum keller))
      (parse eingabe (cons 'Aufgabe keller))
      (parse eingabe (cons 'Erfolg keller))))

(define (art-fkt eingabe keller)
  (parse eingabe (cons 'der keller)))

(define (v-fkt eingabe keller)
  (or (parse eingabe (cons 'öffnen keller))
      (parse eingabe (cons 'feiern keller))))
```

```
(define (pp-fkt eingabe keller)
  (parse eingabe (append (list 'p 'np) keller)))

(define (p-fkt eingabe keller)
  (or (parse eingabe (cons 'mit keller))
      (parse eingabe (cons 'hinein keller))))
```

Beschreiben Sie an dem Beispielaufruf
`(start `(Erfolg feiern mit der Aufgabe))`
wie der Parser arbeitet. (8P)

g) **Erläutern Sie**, welche weiteren Schritte man unternehmen könnte, um die Übersetzung zu verbessern und **bewerten Sie** diese. (7P)

Erwartungshorizont

<table>
<tr><th></th><th>Lösungsskizze</th><th colspan="3">Zuordnung, Bewertung</th></tr>
<tr><th></th><th></th><th>I</th><th>II</th><th>III</th></tr>
<tr><td>a)</td><td>- falsche, fehlende Grammatik
- Mehrdeutigkeit bei Wörtern
- Redewendungen</td><td>4</td><td>2</td><td></td></tr>
<tr><td>b)</td><td> 

Der Aufruf:
<code>(satzuebersetzungeinfach `(Erfolg mit dieser Aufgabe) *lexikon1*)</code>
erzeugt folgende Ausgabe:
<code>(success with (Wort nicht im Lexikon) exercise)</code></td><td>6</td><td>3</td><td></td></tr>
<tr><td>c)</td><td>Eine Optimierung sähe z. B. folgendermaßen aus:
<pre>
define (satzu satz lex)
  (cond
    ((null? satz) '())
    (else
      (cons
         (wortuebersetzereinfach (first satz) lex)
         (satzu (rest satz) lex)))))
</pre>
Nun läuft lediglich die Wortübersetzung durch das Lexikon. Die Satzübersetzung arbeitet Wort für Wort ab, ohne durch das Lexikon zu gehen.</td><td></td><td></td><td>7</td></tr>
<tr><td>d)</td><td>Eine formale Grammatik besteht im Wesentlichen aus:
- Regeln
- Startsymbol
- Nichtterminale
Terminale</td><td>5</td><td></td><td></td></tr>
<tr><td>e)</td><td>Im Wesentlichen müssen die terminalen Symbole eingefügt werden, aber auch das Erkennen von Präpositionen. Folgende Erweiterungen wären nötig, um `(Erfolg feiern mit der Aufgabe) zu erkennen:
<table>
<tr><td>S -> NP VP</td><td>VP -> V PP</td><td>N -> Aufgabe</td></tr>
<tr><td>NP -> N</td><td>PP -> P NP</td><td>N -> Erfolg</td></tr>
<tr><td>NP -> ART N</td><td></td><td>ART -> der</td></tr>
<tr><td>VP -> V</td><td></td><td>P -> mit</td></tr>
<tr><td>VP -> V NP</td><td></td><td>V -> feiern</td></tr>
</table></td><td></td><td>8</td><td></td></tr>
</table>

	Lösungsskizze	Zuordnung, Bewertung		
		I	II	III
	Der Aufruf `(start `(Erfolg feiern mit der Aufgabe))` würde #t ausgeben. Der Parser arbeitet Schritt für Schritt folgende Regeln ab: S → **NP** VP → **N** VP → Terminale einsetzen und bei „Erfolg“ als korrekt erkennen und rausstreichen. ~~N~~ VP → **V** → Terminale einsetzen und bemerken, dass noch korrekt, da im Satz noch Wörter vorhanden, keller-Liste aber leer. VP → **V NP** → für V Terminale ersetzen, dann NP ersetzen und feststellen, dass so kein gültiger Weg. VP → **V PP** → für V Terminale ersetzen, dann PP ersetzen. ~~V~~ PP → **P NP** → für P Terminale ersetzen und dann NP Ersetzen. ~~P~~ NP → **N** → für N Terminale ersetzen und feststellen, dass so kein gültiger Weg. NP → **ART N** → Terminale ersetzen. Satz und keller-Liste gleichzeitig abgearbeitet, daher #t ausgeben		8	
g)	Folgende Bereiche könnten verbessert werden: - Lexikon erweitern - Grammatik-Regeln erweitern. - nach dem Parsen übersetzen (mit Berücksichtigung einer Zielgrammatik). - Redewendungen mit einfließen lassen. Worte miteinander verknüpfen, um bei möglichen Mehrfachbedeutungen besser zuordnen zu können.			7
	Insgesamt 50 BWE	15	21	14

Scheme-Kurzreferenz

Listenfunktionen

Funktion	Funktionsaufruf allgemein	Beispiel	Erläuterung
list	(**list** *element-1 ... element-n*)	(**list** 'der 'hund 'frisst) → (der hund frisst)	liefert die Liste mit den Elementen
cons	(**cons** *element liste*)	(**cons** 'der '(hund frisst)) → (der hund frisst)	fügt das Element vorn in die Liste ein
append	(**append** *liste-1 ... liste-n*)	(**append** '(der hund) '(frisst) '(leckeren Pansen)) → (der hund frisst leckeren Pansen)	liefert eine Liste, die der Reihe nach die Elemente der n Listen enthält
null?	(**null?** *liste*)	(**null?** '(der)) → #f (**null?** '()) → #t	prüft, ob die Liste leer ist

list?	(**list?** *objekt*)	(**list?** 'a) → #f (**list?** '(der hund)) → #t	prüft, ob das Objekt eine Liste ist
equal?	(**equal?** *objekt-1 objekt-2*)	(**equal?** 'der 'the) → #f	prüft, ob die Objekte gleich sind
not	(**not** arg1)	(**not** test)	Gibt „wahr“ zurück, wenn das Argument falsch ist
number?	(**number?** arg1)	(**number?** 234)	Gibt „wahr“ zurück, wenn das Argument eine Zahl ist.

Makros

cond	(**cond** (*bedingung1 ausdruck2*) ... (*bedingung-n ausdruck-n*))	(**cond** ((>? x 2) (+ x 5)) ((=? x 2) (* x 5)) (else (* x 7)))	Mehrfachfallunterscheidung: liefert den Ausdruck zurück, dessen Bedingung zu true evaluiert
define	(**define** *variable ausdruck*)	(**define** *lex* '((cat katze) (dog hund)))	Definition von globalen Variablen
define	(**define** (*funktionsname parameter-1 ... parameter-n) ausdruck*)	(**define** (sqr x) (* x x))	Definition von Funktionen
let	(**let** ((var-1 ausdr-1) (var-2 ausdr-2) ... (var-n ausdr-n)) Ausdruck)	(**let** ((x 2) (y 3)) (* x y))	lokale Variablenbindungen