DHBW Stuttgart
Duale Hochschule Baden-Württemberg

PROGRAMMIEREN IN Java

TINF16 - Sommersemester 2017

von Johannes Unterstein - unterstein@me.com

KLAUSURVORBEREITUNG

Grafik von http://it.wikipedia.org/wiki/File:Duke-Guitar.png

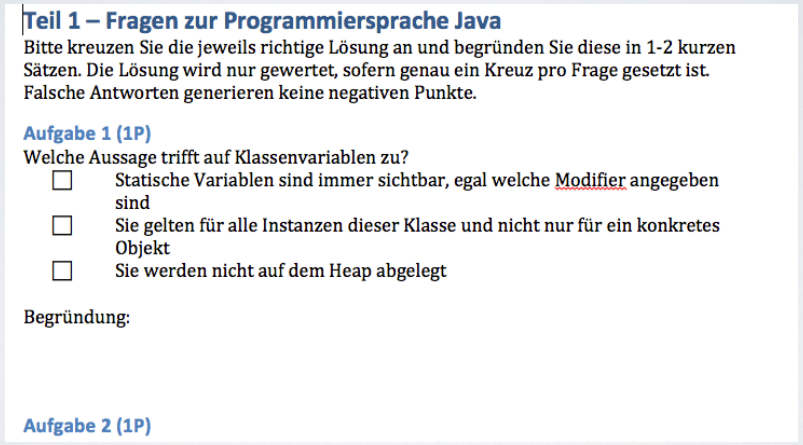
GENERELLES

- 90 Minuten = 90 Punkte
- 4 Teile
- Hilfsmittel: 2 Stifte

AUFGABENTEILE

- Wissensabfrage: Ankreuztest (10 Fragen, 10 Punkte)
  - Genau 1 Antwort richtig, kein negativer Übertrag der Fragen
- Wissensabfrage: Ausformulierung (8-10 Fragen, 48 Punkte)
- Modellierung: Kurze Aufgabe (16 Punkte)
- Codeanalyse: Ausgabe bestimmen (3-4 Aufgaben,16 Punkte)

BEISPIEL AUFGABE 1



Teil 1 – Fragen zur Programmiersprache Java

Bitte kreuzen Sie die jeweils richtige Lösung an und begründen Sie diese in 1-2 kurzen Sätzen. Die Lösung wird nur gewertet, sofern genau ein Kreuz pro Frage gesetzt ist. Falsche Antworten generieren keine negativen Punkte.

Aufgabe 1 (1P)

Welche Aussage trifft auf Klassenvariablen zu?

- ☐ Statische Variablen sind immer sichtbar, egal welche Modifier angegeben sind
- ☐ Sie gelten für alle Instanzen dieser Klasse und nicht nur für ein konkretes Objekt
- ☐ Sie werden nicht auf dem Heap abgelegt

Begründung:

Aufgabe 2 (1P)

Korrekt: b
Klassenvariablen werden auf einer Klasse definiert und sind direkt von der Klasse aus zugänglich. Sie brauchen daher kein konkretes Objekt.

BEISPIEL AUFGABE 1

- Welche Aussage in Bezug auf das Schlüsselwort „final“ ist korrekt? Begründe in einem Satz.
  - [ ] Werte bzw. Attribute eines Objektes, welches als final deklariert wurde, können nicht verändert werden.
  - [ ] Eine Variable, welche als final deklariert wurde, kann nicht neu zugewiesen werden.
  - [ ] Ein Objekt, welches als final deklariert wurde, wird nicht automatisch von der Garbace Collection erfasst.
  - [ ] Ein Objekt, welches als final deklariert wurde, gilt für alle Instanzen der deklarierenden Klasse.

Korrekt: b
Die final-Deklaration bei Variablen besagt, dass diese Variable nicht neu zugewiesen werden kann. Allerdings können die Attribute des Objektes weiterhin verändert werden.

BEISPIEL AUFGABE 1

- Welche Eigenschaft haben statische Variablen? Begründe in einem Satz.
  - ( ) Statische Variablen können nur ein mal initialisiert werden.
  - ( ) Statische Variablen müssen immer final sein.
  - ( ) Statische Variablen können ohne Instanz einer Klasse verwendet werden.

Korrekt: c
Statische Variablen werden auch Klassenvariablen genannt, d.h. sie sind ohne konkretes Objekt, direkt auf der Klasse aufrufbar.

BEISPIEL AUFGABE 1

- Ein Objekt hat den Typ „VwPolo“ und ist einer Variable vom Typ „VwPolo“ zugewiesen. Wenn dieses Objekt nun einer Variable vom Typ „Object“ zugewiesen wird, was passiert? Begründe in einem Satz.
  - [ ] Das Objekt wird in ein Objekt vom Typ „Object“ umkopiert.
  - [ ] Das initiale Objekt wird umgehend aus dem Speicher entfernt.
  - [ ] Es wird eine neue Referenz auf das bereits existierende Objekt erstellt. Das Objekt wird an sich nicht verändert.
  - [ ] Es passiert nichts.

Korrekt: c
Bei Up-/Down-Casts auf Referenztypen wird lediglich eine neue Referenz angelegt, welche den neuen Typ hat und auf das bestehende Objekt im Speicher zeigt, in diesem Fall `Object` auf unseren Polo.

BEISPIEL AUFGABE 1

- Welche Aussage trifft auf Generics zu? Begründe in einem Satz.
  - [ ] Generics sind nur ein historisches Überbleibsel aus Java 1.0.
  - [] Irgendwas sinnloses.
  - [ ] Generics finden einen großen Einsatzzweck in Verbindung mit Collections.
  - [ ] Generics treten immer mit abstrakten Klassen auf.

9

Korrekt: c (Es war bereits sehr häufig c korrekt?)
Das große mächtige Feature von Generics und Collections ist, dass man komplexe Datenstrukturen implementieren kann, sich zum Zeitpunkt der Implementierung aber noch nicht auf einen konkreten Typ festlegen muss.

BEISPIEL AUFGABE 1

- Was besagt der Project Coin Diamond-Operator? Begründe in einem Satz.
  - [ ] Direkte Schnittstelle um Entwickler für Code zu bezahlen.
  - [ ] Definition von Generics können bei der Initialisierung ausgespart werden.
  - [ ] Besserer Umgang mit Enumerations.

Korrekt: b
Diamond Operator besagt zum Beispiel folgendes: `Map<String, String> meineMap = new HashMap<>();` ist nun möglich, statt bisher `Map<String, String> meineMap = new HashMap<String, String>();`

BEISPIEL AUFGABE 1

- Wie werden Objekte in Java wieder zerstört? Begründe in einem Satz.
  - [ ] Über den Destruktor des Objektes.
  - [ ] Gar nicht.
  - [ ] Von der Garbage Collection.

Korrekt: c
In Java gibt es keine Destruktoren (a fällt raus) und irgendwie müssen nicht benötigte Objekte wieder aus dem Speicher gelöscht werden (b fällt raus), daher muss es einen Mechanismus geben, der dies übernimmt -> Garbage Collection.

BEISPIEL AUFGABE 2

Aufgabe 6 (4P)
Erläutern Sie die catch-or-throw Regel und den Mechanismus zur Weitergabe von Exceptions.

Klausur: Programmieren in Java Seite 7 von 16

Diese Regel besagt, dass eine Exception entweder gefangen oder weitergegeben werden muss. Wenn eine Exception gefangen wird, muss sichergestellt werden, dass in der aktuellen Methode genügend Kontext vorhanden ist, um diese auch sinnvoll zu behandeln. Wenn die Ausnahme nicht sinnvoll aufgelöst werden kann, sollte sie weitergegeben werden.
Es gibt drei Wege der Exception Weitergabe.

1) Eine Exception wird gefangen und wird explizit weiter gegeben
2) Eine Exception wird basierend neu erzeugt und explizit weiter gegeben
3) Eine RuntimeException kommt von weiter unten und geht am catch Block vorbei und schlägt direkt durch

BEISPIEL AUFGABE 2

- Was ist Konstruktorverkettung, in welchen Hierarchieebenen ist dies möglich, was ist der Vorteil davon und was muss man bei der Verwendung beachten? Gebe ein kleines Beispiel.

- Umfang etwa 4-5 Sätze plus Beispiel-Code

Konstruktorverkettung=Aufruf eines Konstruktors aus einem anderen heraus. Dies muss in der ersten Zeile des Konstruktors geschehen. Es können Konstruktoren auf der gleichen Klasse aufgerufen werden oder von der Vaterklasse. Der Vorteil ist, dass man Code nicht mehrfach schreiben muss und man Defaultwerte definieren kann.

Bsp.:

```
class Foo {
	public Foo(int i) {

	}
}

class Bar extends Foo {
	public Bar() {
		super(1);
	}
}
```

BEISPIEL AUFGABE 2

- Was sind Streams (java.io.*) und welche unterschiedlichen Transporteinheiten gibt es?
- Nennen Sie für jede Transporteinheit und jede Transportrichtung eine Beispiel-Klasse.

Streams sind eine Sequenz von Daten, auf die ich entweder drauf schreiben oder von denen ich lesen kann. Es gibt die Transporteinheiten `Byte` und `Character`, weiterhin gibt es die Richtung `Lesen` und `Schreiben`.

Beispiele:
InputStream
OutputStream
Writer
Reader

BEISPIEL AUFGABE 2

- Was ist eine for-each-Schleife, geben Sie ein Beispiel und erläutern Sie die Vorteile gegenüber einer herkömmlichen for-Schleife.

Bei einer for-each-Schleife wird über jedes Element einer Collection iteriert und typsicher ohne Laufvariable zugänglich gemacht.

Bsp.: for(User user: alleUser) { user.doSomething(); }

Vorteile sind zum Beispiel, dass kein Element vergessen werden kann, dass nicht über die Grenzen der Collection hinaus zugegriffen werden kann oder ein geringfügiger Performancevorteil.

BEISPIEL AUFGABE 2

- Welche Möglichkeiten bieten Java 8 streamed Collections? Zeigen Sie zwei Beispiele und nennen Sie einen Vorteil gegenüber der Verwendung herkömmlichen Collections!

Streamed Collections bieten die Möglichkeit Logik kompakter auszudrücken und somit nicht mehr zu programmieren `wie` etwas geschehen soll, sondern `was` geschehen soll. Als Beispiele wären hier map, forEach, filter oder count zu nennen. Ein Vorteil wurde eben bereits genannt (kompakte Darstellung, weniger Code).

BEISPIEL AUFGABE 2

- Was versteht man unter Default-Implementations? Nennen Sie kurz den Einsatzort und einen Vor- bzw. Nachteil!

Default-Implementations kommen in Zusammenhang mit Interfaces ab Java 8 zum Einsatz. Unter Default-Implementations versteht man, dass in einem Interface Methoden implementiert werden können.
Vorteil: Methoden können implementiert werden - Zum Beispiel wenn ein bestehendes Interface erweitert werden soll.
Nachteil: Methoden können implementiert werden - Bruch mit der OO

BEISPIEL AUFGABE 2

- Erklären Sie kurz das Konzept von `call by reference copy` und erläutern Sie den Unterschied zu `call by reference` und `call by value`!

In Java gibt es `call by reference copy`, was besagt, dass eine Referenz kopiert wird, bevor sie einer Methode übergeben wird. Bei `call by reference` hingegen ist es so, dass die Referenz direkt in die Methode gegeben wird und bei `call by value` wird der Wert als Kopie in die Methode gegeben.

BEISPIEL AUFGABE 2

- In der Vorlesung wurde das Konzept von `casting auf Referenztypen` vorgestellt. Wird bei einem Up-Cast auf einen Referenztyp ein neues Objekt angelegt? Erläutern Sie kurz was im Speicher bei einem solchen Cast geschieht!

Bei einem Up-Cast auf einen Referenztypen wird lediglich eine neue Referenz angelegt, welche auf das existierende Objekt zeigt. Diese Referenz wird abgespeichert, das originale Objekt im Speicher (Heap) allerdings nicht verändert oder kopiert.

BEISPIEL AUFGABE 2

- Erklären Sie in welchem Kontext man von Boxing bzw. Unboxing spricht und was sich hinter diesem Konzept verbirgt? Was für Probleme können dabei auftreten?

Boxing bzw. Unboxing kommen in Zusammenhang mit primitiven Datentypen und deren zugehörigen Wrapper-Klassen zum Einsatz.
Unboxing besagt dabei, dass der primitive Wert aus dem Wrapper automatisch herausgeholt wird.

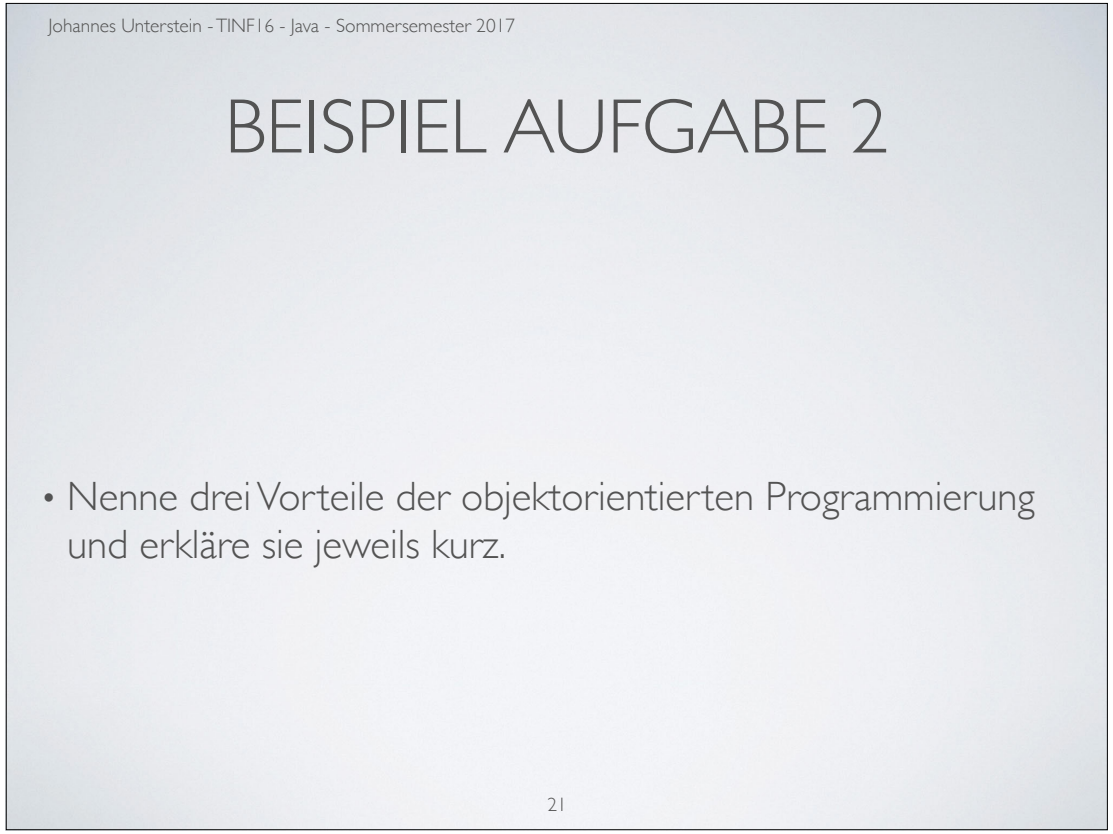
```
Integer i1 = new Integer(12);
int i2 = i1; // Unboxing!
```

Boxing besagt dabei, dass der Wrapper automatisch erstellt werden kann.

```
Integer i3 = i2; // Boxing!
```

Probleme können dabei auftreten, wenn `null`-Werte bei einem Wrapper auftreten können, welche dann automatisch in einen primitiven Datentypen verwandelt werden sollen, da es keine `null`-Repräsentation bei primitiven Datentypen gibt.

```
Integer i1 = null;
int i2 = i1; // Booom!
```



1.) Wiederverwendbarer Code
Da Source Code in Klassen gekapselt sind, können diese in anderen Programmen leichter erneut verwendet werden

2.) Komplexität wird verringert
Da komplexe Probleme in kleinere Probleme aufgeteilt werden können, wird die Komplexität der einzelnen Klassen reduziert und somit leichter verständlich.

3.) Parallele Entwicklung
Wenn große Probleme in kleinere geteilt werden, können diese auch auf verschiedene Personen oder sogar Teams aufgeteilt werden.

BEISPIEL AUFGABE 2

- Was sind Bulk-Operations in Java 8 und was ermöglicht dieses Feature im Vergleich zur klassischen Iteration?

Bulk-Operations ist eine neue Möglichkeit in Java 8 um auf einer Collection (Stream) Funktionen auf jedem einzelnen Event dieses Streams anzuwenden. Dabei gibt es verschiedene Operatoren, wie zum Beispiel map, forEach, reduce, filter, anyMatch, allMatch. Wichtig ist hier zu beachten, dass nicht mehr ausgedrückt wird, wie etwas zu iterieren ist, sondern was man für ein Ergebnis haben möchte.

BEISPIEL AUFGABE 2

- Erläutern Sie das Konzept von Exceptions? Was sind Exceptions, wie werden sie benutzt und muss man beachten?

Das Konzept von Exceptions besagt, dass der Code um die Fehlersituation zu behandeln örtlich von dem Code getrennt ist, der die eigentliche fachliche Logik beinhaltet. Dabei können Exceptions ausgelöst werden und an den Aufrufer weitergegeben werden. Dieser kann dann entscheiden, ob er mit dieser Ausnahmesituation etwas anfangen möchte oder nicht.
Eine Exception als solches ist ein Objekt einer Klasse, welche von der Klasse Exception erbt. Dieses Objekt kann regulär Attribute und Methoden haben.
Beim Fangen von Exceptions ist zu beachten, dass die catch Blöcke in der Reihenfolge von speziell zu genereller gehen müssen, da sonst ggf. eine Exception nicht gefangen werden kann, da der catch Block nicht ausgewertet wird.

BEISPIEL AUFGABE 3

- Zeichne das UML-Klassendiagramm für alle Klassen und Beziehungen im Text. Bei den Klassen genügt die Kurzdarstellung (keine Eigenschaften bzw. Methoden). Bei den Beziehungen genügt der Typ, Zahlenangaben (Kardinalitäten) sind nur für „*“ erforderlich:

- In einem Einkaufszentrum gibt es verschiedene Stockwerke, wobei jedes Stockwert aus mehreren Geschäften besteht. Ein Geschäft ist z.B. ein Restaurant oder ein Schuhgeschäft. Jedes Geschäft besteht aus einer Anzahl Mitarbeitern. Ein Manager leitet ein Geschäft. Beim Sommerfest des Einkaufszentrum gibt es mehrere Attraktionen, welche von verschiedenen Geschäften gesponsert werden.

Siehe https://www.dropbox.com/s/yp284oudfq5cles/Screenshot%202017-07-21%2020.21.05.png?dl=0

BEISPIEL AUFGABE 3

- Geben Sie zu folgender Beschreibung ein UML-Klassendiagramm an. Achten Sie auf korrekte Attribute mit Datentyp, Methodennamen, Assoziationen und Kardinalitäten.
- „Eine Firma ist an beliebig vielen Standorten vertreten. Jeder Standort besteht aus mindestens einem Gebäude mit einer Hausnummer. In jedem Gebäude sind mehrere Büros, jedoch immer nur eine Kantine. Die Büros sind durch- nummeriert und haben ein Namensschild an der Tür. Des weiteren gibt es Angestellte, die sich in die Kategorien Chef, Führungspersönlichkeiten und Arbeiter aufteilen. Dabei müssen an jedem Standort ein Chef, zwei bis zehn Führungspersonen und mindestens drei Arbeiter vorhanden sein, um einen reibungslosen Ablauf zu gewährleisten. Jeder Angestellte besitzt seine eigene ID. Die Angestellte haben unterschiedliche Tätigkeiten. Der Chef entscheidet, die Führungsperson leitet und die Arbeiter arbeiten. Die Firma stellt viele Produkte her, die von den Arbeitern produziert werden.“

25

Ebenfalls online, vergleiche dhbw-stuttgart.de/~unterstein

BEISPIEL AUFGABE 3

- Geben Sie zu folgender Beschreibung ein UML-Klassendiagramm an. Achten Sie auf Klassen, Attribute, Assoziationen und Kardinalitäten.
- Orchester haben einen Namen und einen Heimatort. Sie bestehen aus einem Dirigenten und zwischen zehn und 50 Musikern. Sowohl Musiker als auch Dirigenten verfügen über einen Namen, ein Geburtsdatum und eine Adresse. Jeder Musiker spielt ein Instrument. Jedes Instrument gehört zu einer Instrumentengruppe (Streichinstrument, Holzbläser, Blechbläser,….). Orchester beherrschen Musikstücke (beschrieben durch ihren Titel und ihre Länge). Jedes Musikstück wurde von einem Komponisten (Name, Geburts- und ggf. Todesdatum) geschrieben. Musikstücke benötigen Instrumente in bestimmter Anzahl (Konzert No. 5 braucht z.B. 3 Flöten, 2 Geigen, …). Orchester bringen ein oder mehrere Musikstücke bei Konzerten zur Aufführung. Konzerte sind beschrieben durch Datum, Anfangszeit und Ort. Sie werden von Personen (Name, Adresse) besucht.

BEISPIEL AUFGABE 3

- Geben Sie zu folgender Beschreibung ein UML-Klassendiagramm an. Achten Sie auf korrekte Attribute mit Datentyp, Assoziationen und Kardinalitäten.
- Die 8 Planeten unseres Sonnensystems (außer der Erde) werden in innere und äußere Planeten unterteilt. Sie unterscheiden sich hauptsächlich durch ihre mittlere Entfernung zur Sonne und ihren Durchmesser. Den Tag, an dem Sonne, innerer Planet und Erde eine Linie bilden, nennt man Konjunktionsdatum, den Tag, an dem Sonne, Erde und äußerer Planet eine Linie bilden, Oppositionsdatum. Neben den bekannten großen Planeten schwirren noch tausende Kleinplaneten zwischen Mars und Jupiter umher. Ihre Bahnen sind teilweise so lang gestreckt, dass sie die Bahnen von bis zu 5 anderen Planeten kreuzen können. Generell haben Planeten einen eindeutigen Namen, bis zu 30 Monde und bestehen aus fester oder gasförmiger Oberfläche, Atmosphäre (jeweils bestehend aus mehreren chemischen Elementen) und evtl. einem oder mehreren Ringen (charakterisiert anhand des Durchmessers).

BEISPIEL AUFGABE 3

- Geben Sie zu folgender Beschreibung ein UML-Klassendiagramm an. Beachtet werden nur Klassen und ihre Relationen. Achten Sie auf sinnvolle Assoziationen und Kardinalitäten. Ergänzen Sie sinnvolle Abstraktionen.
- Ein Bahnhof besteht aus Bahnsteigen und Geschäften. Bahnsteige bestehen wiederum aus Treppen, Aufzügen, Rolltreppe, Anzeigetafeln und bis zu zwei Bahngleisen. Geschäfte sind Beispielsweise Bäckereien, Supermärkte oder Autovermietung. Jeder Bahnhof hat beliebig viele Mitarbeiter und genau einen Chef. Mitarbeiter sind zum Beispiel Reinigungskräfte, Personal am Informationsschalter, Servicekräfte und die Frau welche die Durchsagen macht.

BEISPIEL AUFGABE 3

- Geben Sie zu folgender Beschreibung ein UML-Klassendiagramm an. Beachtet werden nur Klassen und ihre Relationen. Achten Sie auf sinnvolle Assoziationen und Kardinalitäten. Ergänzen Sie sinnvolle Abstraktionen.
- Ein MacBook besteht aus einem Deckel und einem Body. Im Deckel ist das Display und im Body ist die Tastatur, die Lautsprecher und bis zu zwei USB-Anschlüsse enthalten. Es gibt MacBook Pro und MacBook Air.

BEISPIEL AUFGABE 3

- Geben Sie zu folgender Beschreibung ein UML-Klassendiagramm an. Beachtet werden nur Klassen und ihre Relationen. Achten Sie auf sinnvolle Assoziationen und Kardinalitäten. Ergänzen Sie sinnvolle Abstraktionen.
- Ein Rechenzentrum besteht aus mehreren Räumen. Ein Raum besteht aus mehreren Racks und in einem Rack sind mehrere Server. In einem Server können bis zu 8 CPU, bis zu 8 Festplatten und bis zu 12 Speicherriegel verbaut werden. Es gibt HighAvailability-Speicherriegel, welche nur in spezielle Server passen.

BEISPIEL AUFGABE 4

Teil 4 – Analyse von Code-Fragmenten

Im Folgenden sind Code-Fragmente zu analysieren. Die Fragen gelten stets nur dann als korrekt beantwortet, wenn jede Antwort begründet wurde. Es kann bei den Code-Fragmenten stets davon ausgegangen werden, dass keine "einfachen" Fehler, wie ein fehlendes Semikolon, vorliegen.

Aufgabe 1 (4P)

Kompiliert der Code und falls ja, welche Ausgabe macht das folgende Programm?
Bitte begründen Sie ihre Antwort ausführlich.
Hinweis: Die Methode `System.out.println(boolean x)` akzeptiert einen boolschen Wert und gibt ihn entsprechend als String aus. Gegebenenfalls fehlende Import-Statements sind zu vernachlässigen.

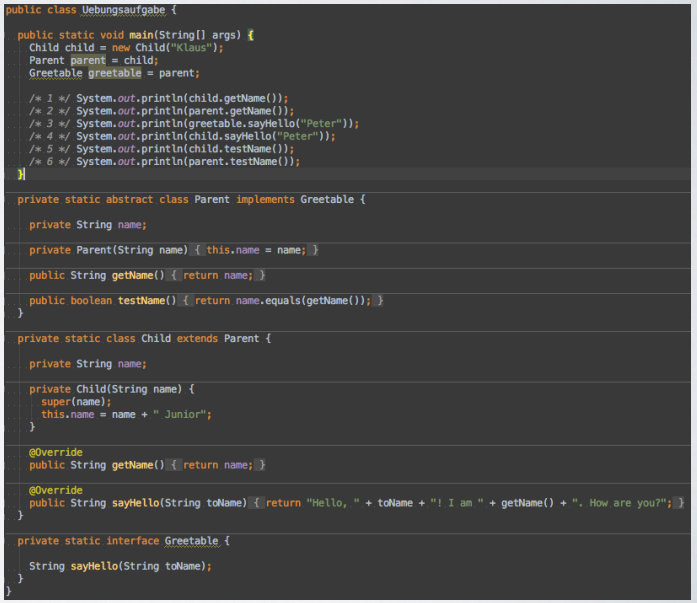
```
public class Indecisive {
    public static void main(String[] args) {
        System.out.println(decision());
    }

    static boolean decision() {
        try {
            return true;
        } finally {
            return false;
        }
    }
}
```

Rückgabe false, da der finally Block auf alle Fälle ausgewertet wird und auch den Rückgabewert überschreiben kann.

BEISPIEL AUFGABE 4

- Was gibt folgendes Programm aus? Begründen Sie Ihre Aussagen!



```
public class Uebungsaufgabe {

  public static void main(String[] args) {
    Child child = new Child("Klaus");
    Parent parent = child;
    Greetable greetable = parent;

    /* 1 */ System.out.println(child.getName());
    /* 2 */ System.out.println(parent.getName());
    /* 3 */ System.out.println(greetable.sayHello("Peter"));
    /* 4 */ System.out.println(child.sayHello("Peter"));
    /* 5 */ System.out.println(child.testName());
    /* 6 */ System.out.println(parent.testName());
  }

  private static abstract class Parent implements Greetable {

    private String name;

    private Parent(String name) { this.name = name; }

    public String getName() { return name; }

    public boolean testName() { return name.equals(getName()); }
  }

  private static class Child extends Parent {

    private String name;

    private Child(String name) {
      super(name);
      this.name = name + " Junior";
    }

    @Override
    public String getName() { return name; }

    @Override
    public String sayHello(String toName) { return "Hello, " + toName + "! I am " + getName() + ". How are you?"; }
  }

  private static interface Greetable {

    String sayHello(String toName);
  }
}
```

Klaus Junior
Klaus Junior
Hello Peter, I am Klaus Junior …
Hello Peter, I am Klaus Junior …
false (Klaus != Klaus Junior)
false (Klaus != Klaus Junior)

Begründung: getName gibt in allen Fällen immer Klaus Junior zurück, da immer ein Objekt gefragt wird, was sein konkreter Typ ist. Dies ist in allen Fällen Child. Von der Klasse Child wird immer ausgegangen und Methoden gesucht, auch wenn diese in der Super-Klasse Parent aufgerufen werden.

BEISPIEL AUFGABE 4

- Was gibt folgendes Programm aus? Begründen Sie Ihre Aussagen!

```
public class Uebungsaufgabe2 {

  public static void main(String[] args) {
    Foo foo = new Foo();

    /* 1 */ System.out.println(Base.SPECIAL_STRING);
    foo.manipulate();
    /* 2 */ System.out.println(Base.SPECIAL_STRING);
    /* 3 */ System.out.println(Foo.SPECIAL_STRING);
    Bar bar = new Bar();
    bar.doSomethingSpecial();
    /* 4 */ System.out.println(Base.SPECIAL_STRING);
    /* 5 */ System.out.println(Foo.SPECIAL_STRING);
    /* 6 */ System.out.println(Bar.SPECIAL_STRING);
  }

  private static class Base {

    static String SPECIAL_STRING = "special";
  }

  private static class Foo extends Base {

    void manipulate() {
      SPECIAL_STRING = "Foo goes here";
    }
  }

  private static class Bar extends Base {

    void doSomethingSpecial() {
      SPECIAL_STRING += ", and something special";
    }

  }
}
```


special
Foo goes here
Foo goes here
Foo goes here, and something special
Foo goes here, and something special
Foo goes here, and something special

Die statische Variable zählt für alle drei Klassen gleichermaßen und ist nur ein mal vorhanden. Jede Manipulation erfolgt auf dieser einen Variable.

BEISPIEL AUFGABE 4

- Was gibt folgendes Programm aus? Begründen Sie Ihre Aussagen!

```
public class Uebungsaufgabe3 {

  public static void main(String[] args) {
    List<Auto> autos = new ArrayList<>();
    autos.add(new Bmw3er());
    autos.add(new OpelAstra());
    autos.add(new Bmw3er());
    autos.add(new OpelAstraKombi());
    for (Auto auto : autos) {
      auto.blinkeRechts();
    }
  }

  private static abstract class Auto {

    void blinkeRechts() {
      System.out.println("blink blink links");
    }

    void blinkeLinks() {
      System.out.println("blink blink rechts");
    }
  }

  private static class Bmw3er extends Auto {

    @Override
    void blinkeRechts() {
      System.out.println("blinker kaputt");
    }
  }

  private static class OpelAstra extends Auto {

    @Override
    void blinkeRechts() {
      System.out.println("blink blink");
    }
  }

  private static class OpelAstraKombi extends OpelAstra {

    @Override
    void blinkeRechts() {
      super.blinkeRechts();
      System.out.println("blink blink blink");
    }
  }
}
```


blinker kaputt
blink blink
blinker kaputt
blink blink
blink blink blink

In der Vererbungshierarchie wird blinkeRechts jeweils überschrieben und genau wie bei der Aufgabe zuvor wird jedes Objekt nach seinem konkreten Typ befragt und von dort aus die Methoden gesucht.