DHBW Stuttgart
Duale Hochschule
Baden-Württemberg

PROGRAMMIEREN IN Java

TINF15 - Sommersemester 2016

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, sonst nix

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

- 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.

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 1

- Was ist der Unterschied zwischen Java 8 streamed Collections #map und #forEach?
  - [ ] Es gibt keinen.
  - [ ] Das eine filtert und das andere gruppiert.
  - [ ] Das eine gibt etwas zurück und das andere nicht.

Korrekt: c
Die Funktion, welche in #map hineingereicht wird, wird auf jedem Element der Collection angewendet und erfordert einen Rückgabewert. Die Funktion, welche in #forEach hineingereicht wird, wird auf jedem Element der Collection angewendet und erfordert keinen Rückgabewert.

BEISPIEL AUFGABE 2

- Erläutern Sie das Konzept der Konstruktorverkettung und beschreiben Sie die möglichen Arten anhand eines Beispiels. Umfang etwa 4-5 Sätze plus Beispiel-Code

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!
```

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.“

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

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

```java
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);
  }
}
```

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";
    }

  }
}
```

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");
    }
  }
}
```