DHBW Stuttgart
Duale Hochschule
Baden-Württemberg

PROGRAMMIEREN IN Java

TINF13 - Sommersemester 2014

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)
- Wissensabfrage: Ausformulierung (8-10 Fragen, 48 Punkte)
- Modellierung: Kurze Aufgabe (16 Punkte)
- Codeanalyse: Ausgabe bestimmen (3-4 Aufgaben,16 Punkte)

BEISPIEL AUFGABE 1

- Welche Antwort ist richtig? 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 des deklarierenden Objektes.

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.

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 nicht verändert.
  - [ ] Es passiert nichts.

BEISPIEL AUFGABE 1

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

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 und welche unterschiedlichen Transporteinheiten gibt es?
- Nennen Sie für jede Transporteinheit und jede Transportrichtung ein Beispiel.

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. Desweiteren 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 korrekte Attribute mit Datentyp, Methodennamen, 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, Methodennamen, 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 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);
  }
}
```

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");
    }
  }
}
```