

Test zu Grundlagen der Programmierung

Leitung: Michael Hahsler

21. November 2003

| | |
|-----------------------|--|
| Name | |
| Matrikelnummer | |
| Unterschrift | |

Bitte kreuzen Sie das Studium an, für das Sie diese Prüfung ablegen:

- Bakkalaureat Wirtschaftsinformatik (LVP)
- SBWL Wirtschaftsinformatik (alt, VO)
- SBWL Informationswirtschaft (alt, AG)
- SBWL Informationswirtschaft (neu, PI)

Diese Angabe umfasst inklusive Deckblatt 11 Seiten.

- Bitte lesen Sie die Angaben aufmerksam und sorgfältig durch.
- Beachten Sie bei Ihrer Beantwortung, dass nur **einwandfrei lesbare** und **eindeutige Antworten** bewertet werden können.
- Verwenden Sie dazu einen nicht radierbaren Kugelschreiber.
- Für diese Prüfung sind **keinerlei Unterlagen** erlaubt!
- Bitte beantworten Sie die Fragen in den dafür vorgesehen Bereichen. Alle anderen Notizen und dergleichen können bei der Beurteilung nicht berücksichtigt werden.
- und nehmen Sie das Fragenheft nicht auseinander!

Für die Bearbeitung der Fragen haben Sie **60 Minuten** Zeit. Insgesamt können Sie bei dieser Prüfung **60 Punkte** erreichen, kalkulieren Sie also pro Punkt eine Minute Bearbeitungszeit.

Viel Erfolg!

| | |
|--------|--|
| Punkte | |
| Note | |

1) Grundkonzepte der Softwareentwicklung

a) *Erläutern sie den Unterschied zwischen Syntax und Semantik.*

Punkte
1

Syntax: Die Regeln für das Bilden **gültiger Sätze** aus diesen Symbolen (= "Grammatikregeln").

Semantik: Die Bedeutung der gültigen Sätze.

1 Punkt

b) *Welche Rolle spielen Syntax und Semantik bei der Compilierung und bei der Ausführung von Java Programmen?*

Punkte
2

Compiler kann nur **Syntax** überprüfen (Compiler-Fehler).
Wenn das Programm ausgeführt wird, entscheidet die **Semantik** was passiert und ob das Programm das **richtige Ergebnis** liefert.

2 Punkte

c) *Wie und wozu wird in Java "Byte Code" eingesetzt?*

Punkte
3

Java kombiniert **Compiler und Interpreter**

1 Punkt

Zunächst wird die Quelldatei mit Hilfe des Java Compilers in Bytecode übersetzt. Der Bytecode **kann noch nicht direkt ausgeführt** werden.

Dazu wird im zweiten Schritt ein Java

Bytecode Interpreter eingesetzt. Jedes System hat seinen eigenen Java Bytecode Interpreter, die „**Java Virtual Machine**“ (JVM)

1 Punkt

Dadurch ist Java Byte Code **plattformunabhängig!**

1 Punkt

2) Primitive Datentypen und Operatoren

a) Welche der folgenden Aussagen sind richtig? Kreuzen Sie die richtige(n) Aussage(n) an.

Punkte
3

- In Java werden einige Typumwandlungen automatisch vorgenommen.
- Variablennamen (Bezeichner) dürfen im gesamten Java Programm nur einmal deklariert werden, da sie eindeutig sein müssen.
- Alle Variablen bekommen in Java bei der Deklaration einen bestimmten Wert zugewiesen.
- Die Division der int-Werte $15/4$ ergibt in Java den gerundeten int-Wert 4.
- Der Wert "a" ist in Java vom Datentyp `char`.
- Variablen (primitive Datentypen) werden in Java bei der ersten Verwendung vom Compiler automatisch deklariert.
- Der Vergleichsoperator ist nur für den Vergleich primitiver Datentypen geeignet.
- Konstanten werden in Java mit dem Schlüsselwort `final` deklariert.
- Das Ergebnis von Vergleichsoperatoren ist vom Datentyp `boolean`.

Pro falscher Antwort 0.5 Punkte Abzug!

b) Was bewirken folgende Operatoren bei primitiven Datentypen? Beschreiben sie die Wirkung und geben Sie je ein Beispiel in Java Code an.

Punkte
3

| Ausdruck | Wirkung | Beispiel |
|----------|---------------------------|----------------------------------|
| + | Addition | 3+8 |
| == | Vergleich | 4==8 |
| && | log. und | true && false |
| * | Multiplikation | 23.5 * 11.8 |
| -= | Subtraktion mit Zuweisung | int z = 9; z -= 3; |
| = | Zuweisung | int a=9; 2 Richtige / Pkt |

3) Ablaufsteuerung

a) **Schreiben Sie eine `for`-Schleife, die von 50 bis 100 alle geraden Zahlen ausgibt! Also: 50, 52, 54, ..., 98, 100 (Beistriche können entfallen)**

Punkte
2

```
for (int i=50; i<=100; i+=2) {  
    System.out.print(i+", ");  
}
```

1 Punkt Syntax

1 Punkt Endbedingung

b) **Simulieren sie den Ablauf folgender Schleife. Schreiben Sie die Ausgabe in das Antwortkästchen!**

Punkte
4

```
int a=0, b=50;  
while(a>100 || b>=-50) {  
    System.out.println(a+"\t"+b);  
    b--; a++;  
}
```

```
0    50  
1    49  
2    48  
...  
50   0  
..  
100  -50  
101  -51  
... Endlosschleife
```

4 Punkte

4) Methoden

a) *Wie unterscheidet sich die Parameterübergabe an Methoden bei primitiven Datentypen und bei Objekt-Datentypen?*

Punkte
2

Primitive Datentype: Wert wird **kopiert**.

Objekt Datentypen: Übergabe **einer Referenz auf die Instanz** aus der aufrufenden Methode. Problem: Veränderungen der Instanz in der Methode!

je 1 Punkt

b) *Fügen Sie in die Klasse `MainPerson` die fehlende Methode `printName` ein (wird von der `main`-Methode aufgerufen). Die Methode soll den Wert der Instanzvariable `name` des Parameters ausgeben.*

Punkte
4

```
class Person {
    String name;
    Person (String name) { this.name=name; }
}

class MainPerson {
    public static void main (String [] args) {
        Person me = new Person("michael");
        printName (me);
    }
}
```

```
public static void printName(Person einePerson) {
    System.out.println(einePerson.name);
}
```

2 Punkte Signatur, Übergabe der Person
2 Punkte Ausgabe mit Punkt-Operator

5) Klassen

a) **Wozu werden in Java Klassendefinitionen benötigt und aus welchen Elementen bestehen Klassendefinitionen?**

Punkte
2

Die Klassendefinition ist der Bauplan für Instanzen. Jedes Java Programm besteht aus Klassen, aus denen beim Ablauf Instanzen erzeugt werden.

1 Punkt

Klassendefinitionen bestehen aus dem Klassennamen, Klassen- und Instanzvariablen und den Methoden.

1 Punkt

b) **Geben Sie ein kurzes Beispiel einer Klassendefinition für eine Klasse, die eine Zeichenkette speichern und ausgeben kann (mit Konstruktor) in Java Code.**

Punkte
2

```
class AClass {
    String b;

    Aclass (String bb) {
        b=bb;
    }

    print() {
        System.out.println(b);
    }
}
```

2 Punkte

c) **Welche der folgenden Aussagen sind richtig? Kreuzen Sie die richtige(n) Aussage(n) an.**

Punkte
2

- Instanzen werden in Java bei der Deklaration der Instanznamensvariablen erzeugt.
- Die spezielle Variable `this` beinhaltet immer die Referenz auf die aktuelle Instanz.
- Klassenvariablen werden mit dem Schlüsselwort `protected` deklariert.
- Konstruktoren können in Java genau so wie andere Methoden überladen werden.
- Jede Instanz besitzt eigene Werte für die Instanzvariablen.
- Durch das Schlüsselwort `new` wird der Konstruktor einer Klasse aufgerufen.

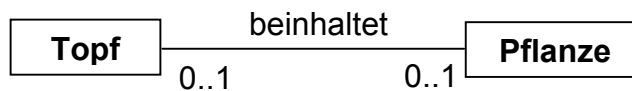
Pro falscher Antwort 0.5 Punkte Abzug!

6) Klassendefinitionen und Beziehungen zwischen Klassen

Implementieren Sie die Klassen Topf und Pflanze für folgendes Main-Programm:

```
class MeinBuero {
    public static void main (String [] args) {
        Topf meinTopf = new Topf(25.0, 30.0); /* Durchmesser und Höhe des
Topfes (in cm) */
        Pflanze meinePflanze = new Pflanze("Rosenstock",30.0); /*
Pflanzenart und Größe (in cm)*/
        meinTopf.einpflanzen(meinePflanze);
        meinTopf.giessen(); /* ruft bei der eingepflanzten Pflanze die
Methode gießen auf und daraufhin wächst die Pflanze um 0.1 cm */
    }
}
```

Klassendiagramm:



a) Implementierung: Klasse Pflanze

Punkte
3

```
class Pflanze {
    String art;
    double groesse;

    Pflanze(String dieArt, double dieGroesse) {
        art=dieArt; groesse=dieGroesse;
    }

    void giessen() {
        groesse += 0.1;
    }
}
```

Definition und Variablen 1 Pkt

Konstruktor 1 Pkt

Methode 1Pkt

b) Implementierung der Klasse Topf

```
class Topf {  
    double durchmesser;  
    double hoehe;  
  
    Pflanze diePflanze;  
  
    Topf(double derDurchmesser, double dieHoehe) {  
        durchmesser=derDurchmesser;  
        hoehe=dieHoehe;  
    }  
  
    void einpflanzen(Pflanze einePflanze) {  
        diePflanze=einePflanze;  
    }  
  
    void giessen() {  
        diePflanze.giessen();  
    }  
}
```

Definition und Variablen (durchm., hoehe) 1 Pkt
Konstruktor 1 Pkt

Methode einpflanzen + Variable diePflanze 2 Punkte
Methode giessen 2 Punkte

7) Wiederverwendung

a) Was ist der Unterschied zwischen normaler Vererbung, Vererbung bei abstrakten Klassen und der Implementierung von Interfaces?

Punkte
3

Bei der Vererbung bekommt die Subklasse alle Eigenschaften (Var. und Methoden) der Superklasse. Die Superklasse kann auch instanziiert werden.

Abstrakte Klassen definieren für einige Methoden nur das Interface und nicht den Code. Subklassen müssen diese fehlenden Methoden implementieren. Abstrakte Klassen können nicht instanziiert werden.

Ein Interface in Java ist die Möglichkeit, zu einer oder mehreren Klassen eine Schnittstelle ohne Implementierung zu definieren. Alle Methoden sind automatisch public

Je 1 Pkt

b) In der Java 2 Plattform API Dokumentation finden Sie folgenden Eintrag:

Punkte
3

The screenshot shows the Java API documentation for the `Float` class. Three parts are highlighted with dotted boxes and numbered:

- 1:** The package name `java.lang` and the class name `Class Float`.
- 2:** The class hierarchy showing inheritance: `java.lang.Object` (parent), `java.lang.Number` (parent of `Float`), and `java.lang.Float` (child of `Number`).
- 3:** The class declaration: `public final class Float`, `extends Number`, and `implements Comparable`.

Below the code, there is a description: "The `Float` class wraps a value of primitive type `float` in an object. An object of type `Float` contains a single field whose type is `float`."

Was bedeuten die Informationen in den Kästchen 1-3?

1 Klasse Float in Package java.lang.
2 Float ist abgeleitet von Number ist abgeleitet von Object

1 Punkt

3 public -> gehört zum Interface des Pakets java.jang.
final -> Klasse kann nicht abgeleitet werden
extends -> abgeleitet von Number
implements -> implementiert das Interface Comparable

2 Punkte

8) Exceptions und Packages

a) Was sind Exceptions in Java?

Punkte
1

Im Ablauf eines Programms können unvorhergesehen Probleme auftreten. Die JavaVM führt dann eine Ausnahmebehandlung (Exception) durch.

b) Geben Sie je ein Beispiele in Java Code, die eine *ArithmeticException* und eine *ArrayIndexOutOfBoundsException* auslösen können:

Punkte
3

ArithmeticException

```
int n =0;  
int r=6/n;
```

ArrayIndexOutOfBoundsException

```
int [] ar = {1,2,3,4};  
int i=9;  
  
System.out.println(ar[i]);
```

c) Was ist ein Package und wie wird in einem Package das Interface definiert?

Punkte
2

Verschiedene, funktional zusammengehörende Klassen werden in Packages integriert.

Die Klassen und Methoden, die von außerhalb des Pakets verwendet werden sollen, müssen als `public` deklariert werden, da sie sonst nur innerhalb des Paketes sichtbar sind.

je 1 Pkt

9) Arrays und Methoden

- a) **Schreiben Sie eine Methode, die als Parameter ein int-Array bekommt, die Reihenfolge der Elemente umdreht und das neue Array als Rückgabewert zurückgibt. Also z.B. aus einem Array mit den Elementen 1, 2, 3, 4, 5 das Array mit den Elementen 5, 4, 3, 2, 1 macht.**

Punkte
6

```
public static int [] reverse (int [] einArray) {
    int [] neuesArray= new int[einArray.length];

    for (int i=0; i<einArray.length; i++) {
        neuesArray[einArray.length-1-i] = einArray[i];
    }

    return neuesArray;
}
```

1 Punkt Signatur
1 Punkt Rückgabe / Rückgabewert
1 Punkt neues Array deklarieren
1 Punkt Schleife
2 Zuweisung mit Index

- b) **Schreiben Sie die Main-Methode, die die gerade implementierte Methode verwendet um ein Array umzuordnen und das neue Array ausgibt.**

```
public static void main (String [] args) {

    int [] array1 = {4, 5, 11, 9, -12, 8};
    int [] array2 = reverse(array1);

    for (int i=0; i<array2.length; i++) {
        System.out.println(array2[i]);
    }

}
```

1 Punkt Deklaration des Arrays
1 Punkt Aufruf der Methode
1 Punkt Ausgabeschleife

Punkte
3