

Test zu Grundlagen der Programmierung

Leitung: Susanne Guth/Michael Hahsler

28. Februar 2003

Name	
Matrikelnummer	
Unterschrift	

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

- Bakkalaureat Wirtschaftsinformatik
- SBWL Wirtschaftsinformatik
- SBWL Informationswirtschaft

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 von Objektorientierung

a) Erklären Sie schlagwortartig was Nachrichten und Methoden sind und wie beide zusammenhängen.

Punkte
2

Nachrichten sind Signale von einem Objekt an ein anderes.
Nachrichten lösen Methoden des anderen Objekts aus.

Methoden stellen das Verhalten von Objekten dar. Methoden bestehen aus O-Name, Methode und Übergabeparameter.

Je 1 Punkt

b) Erklären Sie schlagwortartig Polymorphismus und beschreiben Sie ein Anwendungsbeispiel (kein Code).

Punkte
4

Griechisch: Vielgestaltigkeit
Gemeinsames Interface von verschiedenen Objekten (in Vererbungshierarchie)
Dynamic (late) Binding
Polymorphie funktioniert nur mit Vererbung und Upcasting

Je 1 Punkt (max. 2 Punkte)

Beispiel 2 Punkte

2) Datentypen und Variablen

a) Was bedeutet der Begriff "Datentyp" und wozu gibt es Datentypen?

Punkte
1

Der Datentyp bezeichnet die Art der Daten (Zahlen, Zeichen, Wahrheitswerte)

Grund: Überprüfung sinnvoller Operationen, Reservierung von Speicherplatz, da unterschiedliche Datentypen unterschiedlich viel Speicher benötigen

1 Pkt

b) Was ist der Unterschied zwischen impliziter und expliziter Typumwandlung? Geben sie je ein Beispiel in Java-Code.

Punkte
3

Implizite Umwandlung: Java erweitert Datentypen, damit Operationen möglich sind. Erweiterungen sind nur zu jeweils "weiteren" Datentypen möglich (z.Bsp: int -> long od. int -> double)

Bsp: 3.8+4 ... hier wird automatisch 4 (int) in 4.0 (double) für die Addition erweitert.

1.5 Punkt

Explizite Umwandlung: (Cast) Der Programmierer gibt eine Umwandlung im Code an.

Bsp:

```
double b = 8.9;
```

```
int a = (int) b; ... Die Kommastellen gehen verloren! (Alternativ: int a = int(b);)
```

1.5 Punkt

c) Was bedeuten die unterstrichenen Schlüsselworte in folgendem Code?

Punkte
2

```
public class MyMath {  
    private final double E = 2,718;  
    // hier kommt mehr Code der Klasse  
}
```

private... Der **Zugriff** auf E ist auf die Klasse **beschränkt**.

final ... E wird als **Konstante** deklariert. E kann im Programm nicht mehr verändert werden.

Je 1 Pkt

3) Operatoren

a) Was bewirkt der Operator + bei den Datentypen *int* bzw. *String*?
Geben Sie je ein Beispiel (Code).

Punkte
3

+ int -> Addition (1 Pkt)

```
int a = 100+23;
```

+ String -> Verkettung (1 Pkt)

```
String b = "hallo" + " wie geht es";
```

Für beide Beispiele 1 Pkt

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

Punkte
3

- Zu den logische Operatoren gehören auch ! und &&.
- Der Operator + kann als unärer Operator eingesetzt werden.
- Mehrere Operator in einem Ausdruck werden in Java immer von links nach rechts evaluiert.
- Das Ergebnis des Ausdrucks $5/2$ ist vom Datentyp *int*.
- Das Ergebnis des Ausdrucks $3.0+4$ ist eine ganze Zahl vom Datentyp *int*.
- && ergibt wahr, wenn beide Operanden wahr sind.
- || ergibt wahr, wenn beide Operanden wahr sind.
- Das Ergebnis des Ausdrucks $5/2$ ist eine Gleitkommazahl.
- Der Operator == hat als Ergebnis einen Wahrheitswert.

Pro falscher Antwort 1 Punkt Abzug!

4) Klassen

a) **Was sind Instanzvariablen? Geben Sie den Java-Code für eine Klasse an, in der mindestens 3 Instanzvariablen mit unterschiedlichem Datentyp deklariert werden.**

Punkte
4

Instanzvariablen speichern den Zustand von Instanzen (Objekten).
Jede erzeugte Instanz besitzt eine eigene Kopie der Variable.

2 Pkt

Bsp:

```
class MyClass {  
    int a;  
    float b;  
    char c;  
}
```

2 Pkt

b) **Schreiben Sie einen Konstruktor mit Parametern, der die 3 Instanzvariablen aus dem vorigen Beispiel bei der Instanzierung mit Werten initialisiert.**

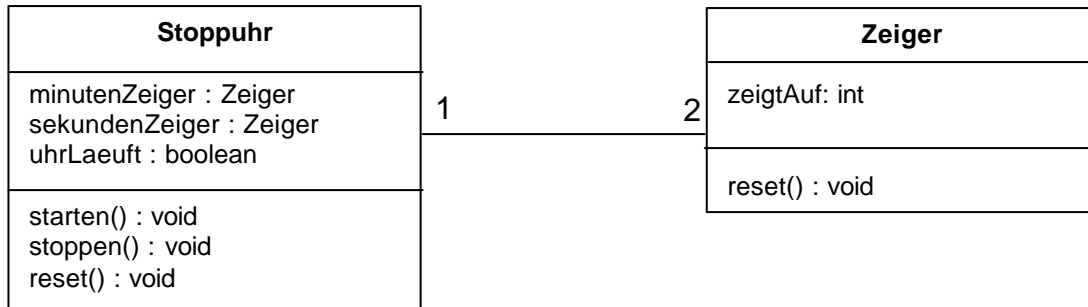
Punkte
2

In class MyClass:

```
MyClass(int a, float b, char c) {  
    this.a = a;  
    this.b = b;  
    this.c = c;  
}
```

2 Pkt

5) Klassendefinitionen



- a) **Implementieren Sie die Klasse Zeiger. Die Instanzvariable `zeigtAuf` gibt die abgelaufenen Sekunden bzw. Minuten an (0-60). Die Methode `reset()` setzt den Zeiger wieder auf 0.**

Punkte
2

```

class Zeiger {
    int zeigtAuf;
    void reset() {
        zeigtAuf=0;
    }
}
  
```

Definition 1 Pkt
Methode 1Pkt

- b) **Implementieren Sie die Klasse Stoppuhr. Der Konstruktor soll die 2 benötigten Zeiger erzeugen (im UML-Klassendiagramm durch die Linie zwischen Stoppuhr und Zeiger dargestellt, keine Vererbung!). Die Methoden `starten()` und `stoppen()` verändern lediglich den Wahrheitswert der Variable `uhrLaeuft`. `reset()` setzt beide Zeiger auf 0.**

Punkte
4

```

class Stoppuhr {
    Zeiger minutenZeiger;
    Zeiger sekundenZeiger;
    boolean uhrLaeuft;

    Stoppuhr() {
        minutenZeiger = new Zeiger();
        sekundenZeiger = new Zeiger();
    }

    void starten() { uhrLaeuft=true; }
    void stoppen() { uhrLaeuft=false; }
    void reset() {
        minutenZeiger.reset();
        sekundenZeiger.reset();
    }
}
  
```

1 Pkt Klassendef
1 Pkt Verbindung zu Zeiger (fett)
1 Pkt starten/stoppen
1 Pkt reset

Hinweis: Die Datentypen sind im Diagramm nach dem Doppelpunkt angegeben.

6) Methoden

a) **Aus welchen Teilen besteht die Deklaration (Signatur) einer Methode? Geben Sie ein Beispiel in Java-Code.**

Punkte
2

Sichtbarkeit/Zugriff, Rückgabewert, Name, Parameterliste mit Datentypen

Bsp: public boolean youngerThan(double age1, double age2)

2 Pkt

b) **Schreiben Sie eine Methode mit dem Namen `istGroesser`, damit folgender Code ein sinnvolle Ausgabe liefert:**

Punkte
4

```
double a= 25.5;
if (istGroesser(a, 15.0)) {
    System.out.println(a+" ist groesser als 15");
}else{
    System.out.println(a+" ist kleiner oder gleich 15");
}
```

```
boolean istGroesser(double a, double b) {
    if (a>b) { return ture; } else { return false; }
}
```

1 Pkt ... Rückgabewert

1 Pkt ... Parameter

2 Pkt ... if mit return

7) Methoden II

Gegeben ist folgende Klasse:

```
public class Zahl {
    public double zahl;
    public Zahl(double zahl) { this.zahl = zahl; }
}

public class Taschenrechner {
    public Zahl addieren(Zahl a, Zahl b) {
        return new Zahl(a.zahl+b.zahl);
    }
    // weiterer Code ausgelassen
}
```

a) Erzeugen Sie in der main-Methode folgender Klasse zwei Instanzen der Klasse Zahl mit den Werten 49 und 8. Verwenden Sie dann die Klasse Taschenrechner, um die zwei Zahlen zu addieren. Das Ergebnis soll in einer 3. Instanz von Zahl gespeichert werden.

Punkte
3

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

        Zahl a = new Zahl(49.0);
        Zahl b = new Zahl(8.0);           // 1 Pkt für Konstruktor

        Taschenrechner meinRechner = new Taschenrechner(); // 1 Pkt
        Zahl c=meinRechner.addieren(a,b);           // 1 Pkt

    }
}
```

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

Punkte
3

- Gibt es mehrere Methoden mit dem gleichen Namen (überladene Methode), wird immer die als letztes definierte Methode aufgerufen.
- Methoden können bei der Vererbung überschrieben werden.
- Arrays werden bei der Übergabe als Parameter kopiert.
- Private-deklarierte Methoden können nur von Methoden innerhalb des selben Pakets aufgerufen werden.
- Objekt-Datentypen werden als Parameter nur als Referenz an Methoden übergeben.
- Methoden mit dem Schlüsselwort `abstract` haben keinen konkreten Rückgabewert.
- Primitive Datentypen werden bei der Übergabe als Parameter kopiert.
- Methoden können in Java maximal einen Rückgabewert und 3 Parameter haben.
- Konstruktoren haben keinen Rückgabewert.

Pro falscher Antwort 1 Punkt Abzug!

8) Ablaufsteuerung

a) Die folgende Schleife soll von 100 auf 0 herunter zählen. Ergänzen Sie den fehlenden Code.

Punkte
2

```
for ( int i=_____ ; _____ ; _____) {  
    System.out.println(i);  
}  
  
int i=100 ; i>=0 ; i--
```

b) Schreiben Sie den folgenden Code so um, dass statt *switch* nur *if*-Anweisungen verwendet werden. Die Variable *note* ist als *int* definiert und kann Werte zwischen 1 und 5 annehmen.

```
switch(note) {  
    case 1: case 2: System.out.println("Gute Leistung!"); break;  
    case 3: case 4: System.out.println("Leistung ok!"); break;  
    case 5: System.out.println("Schlechte Leistung!"); break;  
    default: System.out.println("Ungültige Note!");  
}
```

Punkte
4

```
if (note ==1 || note ==2) { // auch möglich mit > oder >=  
    System.out.println("Gute Leistung!");  
}else if (note ==3 || note ==4) {  
    System.out.println("Leistung ok!");  
}else if (note == 5) {  
    System.out.println("Schlechte Leistung!");  
}else{  
    System.out.println("Ungültige Note!");  
}
```

Pro if 1 Pkt
Letztes else 1 Pkt

9) Strings, Arrays und Objekt-Datentypen

a) **Deklarieren Sie eine Datenstruktur, die 255 Familiennamen speichern kann. Schreiben Sie an die letzte Stelle in der Datenstruktur ihren eigenen Familiennamen.**

Punkte
3

```
String[] meinArray = new String[255];    (1 Pkt)
meinArray[254] = "MeinName";           (1 Pkt für den richtigen Index +
                                         1 Pkt für doppelte Anführungszeichen)
```

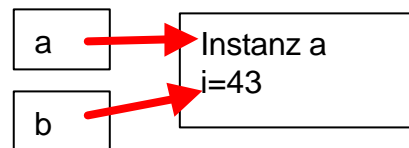
b) **Wie unterscheiden sich Objekt-Datentypen von primitive Datentypen beim Zuweisungsoperator? Erklären Sie anhand des folgenden Beispiels (die fettgedruckten Zeilen):**

Punkte
3

```
class Simple {
    int i;
    Simple (int i) {
        this.i=i
    }
}

Simple a = new Simple(43);
Simple b;
b=a;
a.i=10;
System.out.println("a: "+a.i+"; b:"+b.i);
```

b=a weist die Referenzen auf die Instanzen a auch der Namensvariable b zu. Beide Namen zeigen auf die selbe Instanz. (2 Pkt)



Die Instanzvariable von a wird auf 10 gesetzt. Da b und a auf die selbe Instanz weisen, ist die Ausgabe:

a: 10; b: 10 (1 Pkt)

Bei primitiven Datentypen werden die Werte zugewiesen und kopiert.

Bsp:
int a =43;
int b=a;



a und b sind verschiedene Variablen und voneinander unabhängig. a=10 verändert b nicht!

10) Wiederverwendung

```
abstract class Tier {
    public String rasse;
    Tier(String rasse) {
        this.rasse=rasse;
        System.out.println("Ich bin ein neues Tier und ich bin ein "+
            rasse+".");
    }
    abstract void geraeuschMachen();
}
```

- a) **Schreiben Sie die Klasse Hund, die die abstrakte Klasse Tier erweitert. Die Klasse Hund soll die Methode `geraeuschMachen()` so erweitern, dass auf dem Bildschirm der String "Wuff!" ausgegeben wird. Im Konstruktor der Klasse Hund soll die Rasse als Parameter angegeben werden können.**

Punkte
4

```
class Hund extends Tier {
    Hund(String Rasse) {
        super(Rasse);
    }
    void geraeuschMachen() {
        System.out.println("Wuff!");
    }
}
```

1 Pkt Klassendef.
2 Pkt Konstruktor
1 Pkt Methode

- b) **Was ist die Ausgabe folgender Main-Methode**

```
class EinHund{
    public static void main(String[] arg) {
        Hund benno = new Hund("Bernhardiner");
        benno.geraeuschMachen();
    }
}
```

Punkte
2

Ausgabe:
Ich bin ein neues Tier und ich bin ein Bernhardiner
Wuff!

(je 1 Pkt)