

Name:

Matrikelnummer:

Test zu Grundlagen der Programmierung

Leitung: Susanne Guth/Michael Hahsler

31. Jänner 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.
- Beschriften Sie jede Seite des Lösungstextes mit Ihrem Namen und Ihrer Matrikelnummer
- 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	

Name:

Matrikelnummer:

1) Grundkonzepte von OO

a) *Zählen Sie die wichtigsten Grundkonzepte der Objektorientierung auf (mindestens 4)*

Punkte
2

Klassen/Objekte
Nachrichten
Vererbung
Polymorphismus
Kapselung
Interface

Je 1 Punkt

b) *Erklären Sie schlagwortartig 2 der Grundkonzepte genauer.*

Punkte
4

Je Konzept 2 Punkte

Name:

Matrikelnummer:

2) Datentypen und Variablen

a) Welche primitiven Datentypen werden in Java unterstützt?(Nennen Sie mindestens 5)

Punkte
1

Int , short, long
Boolean
Float, double
char

1 Punkt bei 5

b) Deklarieren Sie mindestens 4 Variablen mit unterschiedlichen Datentypen

Punkte
2

2 Variablen / Punkt

c) Welchen Datentyp können folgende Werte haben?

Punkte
3

0.0	Double (float)
"Hallo"	String
"h"	String (nicht char!)
21	Int od. short od. long
-931	Int od. short od. long
false	Boolean
2 Richtige / Punkt	

Name:

Matrikelnummer:

3) Operatoren

a) Was ist der Unterschied zwischen dem Operator = und dem Operator ==? Geben Sie je ein Beispiel.

Punkte
3

= ist der Zuweisungsoperator (1 Pkt)

```
int a = 100;
```

== ist der Vergleichsoperator (1 Pkt)

```
a==90
```

Für beide Beispiele 1 Pkt

b) Was ergeben folgende Ausdrücke (Wert und Datentyp):

Punkte
3

```
int a=9;  
int b=4;  
double c=3.0;  
boolean d = true;
```

Ausdruck	Wert	Datentyp
a/b	2	int
a/c	3.0	double
d && false	false	boolean
a>b	true	boolean
a!=b	true	boolean
(int) c*a	27	Int 2 Richtige / Pkt

Name:

Matrikelnummer:

4) Klassen

a) **Was ist ein Konstruktor? Wann und wozu wird er benötigt?**

Punkte
2

Besondere *Methode*, die bei der *Erstellung einer Instanz* aufgerufen wird.
Die *Instanzvariablen* können mit dem Konstruktor *initialisiert* werden.

...

Je kursives Wort 1 Pkt.

b) **Wann und wofür verwenden Sie folgende Java Schlüsselwörter?
Antworten Sie in Stichworten und geben Sie je ein Beispiele (Code).**

Punkte
2

`public`

Definiert das Interface einer Klasse oder eines Pakets. (1 Pkt)

Bsp: (1 Pkt)

```
public void doSomething(){...}  
public int c;  
public class SomeClass {...}
```

`private`

Interne Methoden und Variablen der Klasse: (1 Pkt)

- Versteckt (Data Hiding),
- können von aussen nicht verwendet werden,
- gehört nicht zum Interface

Bsp: (1 Pkt)

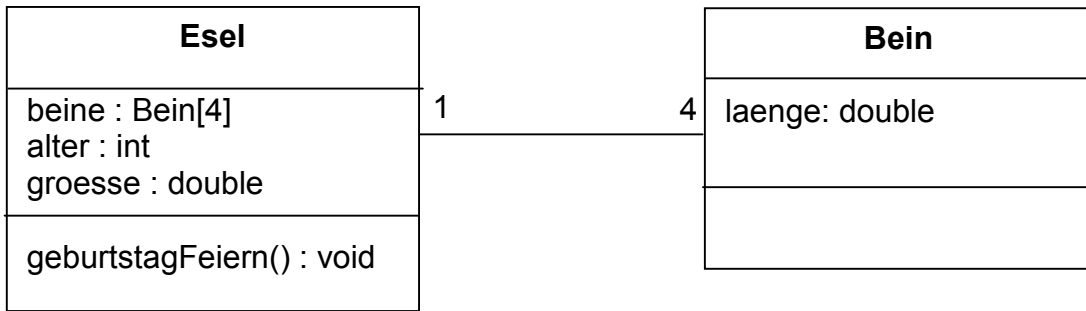
```
private double d;  
private int doSomethingSecret() {...}
```

Punkte
2

Name:

Matrikelnummer:

5) Klassendefinitionen



a) **Implementieren Sie die Klasse Bein. Im Konstruktor soll die Länge des Beins festgelegt werden.**

Punkte
2

```
class Bein {
    double laenge;
    Bein(double laenge) {
        this.laenge=laenge;
    }
}
```

Definition 1 Pkt
Konstruktor 1Pkt

b) **Implementieren Sie die Klasse Esel mit Konstruktor. Die Klasse Esel benutzt die Klasse Bein (im UML-Klassendiagramm durch die Linie zwischen Esel und Bein dargestellt; keine Vererbung!). Die Methode geburtstagFeiern() erhöht das Alter des Esels um ein Jahr.**

Punkte
4

```
class Esel {
    int alter;
    double groesse; // in cm
    Bein[] beine; // Alternativ nicht als Array sondern bein1, bein2,...

    Esel(int alter, double groesse) {
        this.alter=alter;
        this.groesse=groesse;
        beine = new Bein[4];
        for(int i=0; i<beine.length;i++){
            beine[i] = new Bein(60);
        }
    }

    void geburtstagFeiern() {
        alter++;
    }
}
```

Klassendefinition 1 Pkt
Beziehung (Fettdruck) 2 Pkt

1 Pkt

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

Name:

Matrikelnummer:

6) Methoden

```
class Summierer {  
    int sum (int i1, int i2) { return i1+i2; }  
    double sum (double d1, double d2) { return d1+d2; }  
    int sum (double d3 ,double d4) { return (int) d3+d4; }  
    double sum (double d1, double d2, double d3) { return d1+d2+d3; }  
}
```

a) **Warum kann die Klasse Summierer nicht kompiliert werden? Die Fehlermeldung des Kompilers lautet:** Summierer.java:4:

sum(double,double) is already defined in Summierer

Woran liegt das? Welche Methoden erzeugen den Fehler?

Punkte

4

2 Methoden mit der *gleichen Signatur* `sum(double, double)`
Beim *Überladen von Methoden* muss die Signatur (Name + Parameteranzahl/Parametertypen) *eindeutig* sein
(2 Pkt)

Fehlerhafte Methoden: **(2Pkt)**

```
double sum (double d1, double d2) { return d1+d2; }  
int sum (double d3 ,double d4) { return (int) d3+d4; }
```

Beide haben die Signatur `sum(double, double)`

Da die Parameternamen und der Rückgabewert irrelevant sind

b) **Geben Sie 2 zusätzliche Methoden mit dem Namen `sum` an, die in der Klasse Summierer gültig definiert werden können.**

Punkte

2

Bsp: **(je 1 Pkt)**

```
float sum (float f1, float f2) { return f1+f2; }  
int sum (int i1, int i2, int i3) { return i1+i2+i3; }
```

Name:

Matrikelnummer:

7) Aufruf von Methoden

a) Was ist die Ausgabe des folgenden Programms?

Punkte
3

```
class AClass {
    public static int addieren (int a, int b) {
        int c=a+b;
        a++; b=b+a;
        System.out.println("addieren: a="+a+", b="+b+" und c="+c);
        return c;
    }

    public static void main(String[] args) {
        int a=12;
        int b=31;
        int c=addieren(a,19);
        System.out.println("main: a="+a+", b="+b+" und c="+c);
    }
}
```

Ausgabe:

addieren: a=13, b=32 und c=31 (1 Pkt für die Parameterübergabe)

main: a=12, b=31 und c=31 (2 Punkte für die Rückgabe + Verständnis von lokalen Variablen)

b) Was ist die Ausgabe des folgenden Programms?

```
class BClass {
    public static int addieren (int[] a) {
        int sum =a[1]+a[2];
        a[1]++; a[2]++;
        int c = 9;
        System.out.println("addieren: a={" +a[1]+" , "+a[2]+"} und c="+c);
        return sum;
    }

    public static void main(String[] args) {
        int a[] = {1,2,3};
        int c=addieren(a);
        System.out.println("main: a={" +a[1]+" , "+a[2]+"} und c="+c);
    }
}
```

Ausgabe:

addieren: a={3, 4} und c=9 (2 Punkte Verständnis von Parameterübergabe + Index von Arrays)

main: a={3, 4} und c=5 (1 Punkt Verständnis von Arraynamen)

Punkte
3

Name:

Matrikelnummer:

8) Ablaufsteuerung

Weisen Sie jeweils der Variable a ihre Matrikelnummer zu (vernachlässigen Sie dabei die führenden Nullen; Matrikelnummer 0052880 => int a= 52880;) und schreiben Sie den Output der Code-Fragmente in die Kästchen. Achtung: Prüfen Sie die Endbedingungen der Schleifen sorgfältig!

a) while-Schleife

```
int a=_____;  
while (a>=100) {  
    System.out.println(a);  
    a = a/10;  
}
```

Punkte
2

Bsp: int a = 152277;

Ausgabe:

152277
15227
1522
152

(so oft bis die Ausgabe nur noch 3 Stellen hat: **2 Punkte**)

b) for-Schleife

```
int a=_____;  
for (int i=a; i<10000000; a++) {  
    System.out.println(i+" Meine Matrikelnummer ist "+a);  
}
```

Punkte
2

Bsp: int a = 152277;

Ausgabe:

152277. Meine Matrikelnummer ist 152277
152277. Meine Matrikelnummer ist 152278 (1 Pkt)

.
. .
. . .

Endlosschleife, da i in der Schleife nicht verändert wird! (1 Pkt)

c) do-while-Schleife

```
int a=_____;  
int b=10;  
do {  
    int i=2;  
    b=b*i;  
    System.out.println(a);  
    i++; a--;  
} while (a>=1000 && b<=50);
```

Punkte
2

Bsp: int a = 152277;

Ausgabe:

152277
152276
152275 (3 Zeilen da b 3x (solange <= 50) erhöht wird, **2 Punkte**)

Name:

Matrikelnummer:

9) Strings, Arrays und Objekt-Datentypen

a) **Deklarieren Sie ein Array vom Datentyp `double` mit der Länge 10. Weisen Sie danach an der 4.Stelle den Wert 4.99 zu.**

Punkte
2

```
double[] meinArray = new double[10];    (1 Pkt)
```

```
meinArray[3] = 4.99;                    (1 Pkt für den richtigen Index)
```

b) **Wie unterscheiden sich primitive Datentypen von Objekt-Datentypen beim Vergleichsoperator? Erklären Sie anhand des folgenden Beispiels:**

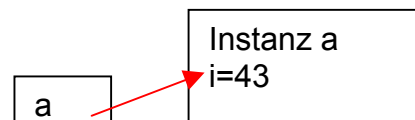
Punkte
4

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

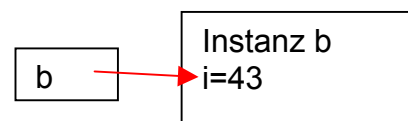
```
Simple a = new Simple(43);  
Simple b = new Simple(43);  
boolean result = (a==b);
```

`a==b` vergleicht die Referenzen auf die 2 Instanzen (die Namensvariablen der Instanzen) der Klasse Simple.

Es werden nur die zwei Zeiger verglichen, die auf unterschiedliche Instanzen zeigen, nicht aber ob die Instanzen die gleichen Werte (in diesem Fall 43) enthalten
(2 Pkt)



`result` bekommt den Wert `false` **(2 Pkt)**



Bei primitiven Datentypen werden die Werte verglichen.

Bsp:

```
int a =9;
```

```
boolean result2 = (a==9); // ergibt true
```

Name:

Matrikelnummer:

10) Wiederverwendung

```
abstract class Pflanze {
    protected double groesse; // Groesse in cm
    Pflanze() {
        System.out.println("Ah, endlich lebe ich!");
    }
    void wieGrossBistDu() {
        System.out.println("Ich bin " + groesse + "cm");
    }
    abstract void wachsen(double zeit); // Zeit in Tagen
}

class Rose extends Pflanze {
    Rose() {
        super();
        System.out.println("Es ist gut eine Rose zu sein.");
    }
    void wachsen(double zeit) {
        groesse += zeit * 0.5;
    }
}
```

- a) **Die main-Funktion in der Klasse Garten verwendet die oberhalb angegebenen Klassen. Vervollständigen Sie das Programm. Es soll eine Instanz vom Typ Rose erzeugen und folgende Nachrichten (Messages) an die Instanz schicken: Eine Woche wachsen; die Grösse der Rose (Instanz) abfragen.**

Punkte
3

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

        Rose eineRose = new Rose();
        eineRose.wachsen(7);
        eineRose.wieGrossBistDu();

        (je 1 Pkt)

    }
}
```

- b) **Was wird bei den einzelnen Befehlen am Bildschirm ausgegeben?**

Punkte
3

```
Äusgabe:
Ah, endlich lebe ich!
Es ist gut eine Rose zu sein.
Ich bin 3.5cm

(je 1 Pkt)
```