

Probeklausur – Java  
Einführung in die Informatik

Wintersemester 2017/2018

Musterlösung

Hinweis: Diese Probeklausur ist eine kleine Aufgabensammlung, die etwa dem Schwierigkeitsgrad der schriftlichen Prüfung des Moduls Einführung in die Informatik entspricht. Die hier zur Verfügung gestellten Aufgaben decken jedoch nicht alle behandelten Themenbereiche ab. Darüber hinaus wird es in der Klausur Wissensaufgaben geben, welche aus allen Themen und sämtlichen Teilen der Veranstaltung (Vorlesung, Tutorium und Hausaufgaben) kommen können. In den Klausuren für dieses Semester wird die Gewichtung ca. 60 % Programmieren und 40 % Rechneraufbau sein.

**Aufgabe 1 (Allgemeine Fragen (Java)).**

In dieser Aufgabe ist jeweils genau eine Antwort richtig, welche Sie ankreuzen sollen. Kreuzen Sie pro Teilaufgabe nur ein Kästchen an. Eine richtige Antwort ergibt einen Punkt, eine falsche 0 Punkte. Es gibt keine Minuspunkte. Um ein versehentlich gesetztes Kreuz wieder zu löschen, füllen Sie das jeweilige Kästchen aus und zeichnen ein leeres daneben. Ein Beispiel:

<i>Fragetext</i>	<u>Korrektur</u> →	<i>Fragetext</i>
<input checked="" type="checkbox"/> Antwort 1		<input type="checkbox"/> ■ Antwort 1
<input type="checkbox"/> Antwort 2		<input type="checkbox"/> Antwort 2
<input type="checkbox"/> Antwort 3		<input checked="" type="checkbox"/> Antwort 3
<input type="checkbox"/> Antwort 4		<input type="checkbox"/> Antwort 4

1. Welcher der folgenden Datentypen ist **kein** primitiver Datentyp?

- int
- double
- String
- byte

2. Welche der folgenden Anweisungen gibt einen Compilerfehler?

- `int[] d = {0,1,2,3,4,5,6,7};`
- `int[] c = {};`
- `double[] b = new int[2];`
- `String[] a;`

3. Wie oft wird die folgende Schleife durchlaufen?

```
int a = 10;
boolean run = true;
while(run){
    if(a%7 == 0){
        run = !run;
    }
    a--;
}
```

- 0
- 3
- 4
- 5

4. Welcher der folgenden Ausdrücke wird garantiert zu **true** ausgewertet?

- `(a < b) && (b < c)`
- `!(b || a)`
- `!!(a || !a)`
- `(a >= b && b >= a)`

5. Welche Aussage zu objektorientierter Programmierung in Java ist **falsch**?

- Java erlaubt es, Konstruktoren mit dem Sichtbarkeitslevel `private` zu versehen.
- Java erlaubt das Erben von mehreren Superklassen.
- Von einer Klasse dürfen im allgemeinen beliebig viele andere Klassen erben.
- Eine abstrakte Klasse kann statische Methoden enthalten.

6. Welche Aussage bezüglich des zweidimensionalen Arrays `int[] [] a = {{1,2,3},{4,5,6}}` ist korrekt?
- `a[0].length == 2` ergibt `true`
  - `a.length == 2` ergibt `false`
  - `a == b` ergibt `false` für `int[] [] b = {{1,2,3},{4,5,6}}`;
  - `int b = a[a.length]`; gibt keinen Compilerfehler
7. Welche Aussage über Konstruktoren in Java ist falsch?
- Konstruktoren haben immer den gleichen Name wie die Klasse
  - Der Konstruktor einer Superklasse kann in einer Subklasse durch `super()` aufgerufen werden
  - Es können mehrere Konstruktoren in einer Klasse definiert werden
  - In jeder Klasse muss ein Konstruktor explizit definiert sein
8. Wir haben die folgende Beziehung gegeben `public class Motorrad extends Fahrzeug{...}`?  
Was entspricht dem Prinzip der Polymorphie?
- `Fahrzeug f = new Fahrzeug();`
  - `Motorrad moto = new Object();`
  - `Fahrzeug f = new Motorrad();`
  - `Motorrad moto = new Fahrzeug();`
9. Welche Aussagen über Interfaces ist falsch ?
- Die Sichtbarkeit der Methoden des Interfaces entspricht immer der des Interfaces
  - Eine Klasse die ein Interface implementiert muss alle Methoden des Interfaces implementieren, damit Objekte der Klasse erzeugt werden können
  - Klassen können jeweils nur ein Interface implementieren
  - Ein Interface kann als Datentyp verwendet werden

**Aufgabe 2 (Bedingte Anweisungen).**

1. Betrachten Sie die folgende Java-Methode

```
static boolean f(boolean x, boolean y) {
    if(x) {
        return y;
    } else if(y) {
        return x;
    } else {
        return x;
    }
}
```

- (a) Stellen Sie eine Tabelle auf, die für alle möglichen Wertepaare  $(x, y)$  der Parameter den zugehörigen Rückgabewert der Methode `f()` angibt.

**Lösung:**

x	y	f(x, y)
true	true	true
true	false	false
false	true	false
false	false	false

- (b) Implementieren Sie eine möglichst einfache Methode `boolean f2(boolean x, boolean y)`, die ohne `if`-Anweisung auskommt und für alle möglichen Wertekombinationen von `x` und `y` die gleichen Werte wie die Methode `f()` zurückliefert.

**Lösung:**

```
static boolean f2(boolean x, boolean y) {
    return x && y;
}
```

2. Betrachten Sie das folgende Punktesystem zur Notenvergabe:

Punkte	Note
36-40	1
31-35	2
26-30	3
21-25	4
0-20	5

Implementieren Sie eine Methode `int getNote(int punkte)`, die als Wert die Note der übergebenen Punkte zurückliefert. Liefern Sie den Wert `-1` zurück, falls der Parameter `punkte` keine gültige Punktezahl darstellt.

**Lösung:**

siehe nächste Seite.

```
static int getNote(int punkte) {  
    if (punkte > 40 || punkte < 0){  
        return -1;  
    } else if (punkte > 35) {  
        return 1;  
    } else if (punkte > 30) {  
        return 2;  
    } else if (punkte > 25) {  
        return 3;  
    } else if (punkte > 20) {  
        return 4;  
    } else {  
        return 5;  
    }  
}
```

**Aufgabe 3 (Schleifen).**

1. Welchen Wert hat `k` nach Ausführung der folgenden Anweisungen?

```
int k = 2;
for(int i = 2; i < 7; i += 2) {
    k *= 2;
}
```

**Lösung:**

`k = 16`

2. Wandeln Sie die folgende `for`-Anweisung in eine `while`-Anweisung um.

```
for(int i = 533; i > -12; i -= 7) {
    System.out.println(i);
}
```

**Lösung:**

```
int i = 533;
while(i > -12) {
    System.out.println(i);
    i -= 7;
}
```

3. Implementieren Sie eine statische Methode `public static int[] threeDecimals(int[] zahlen)`, die ein `int`-Array zurückgibt. Dieses Array soll mit den ersten drei Einträgen von einem übergebenem `int`-Array `zahlen` besetzt werden. Sollte einer der ersten drei Einträge von `zahlen` entweder nicht existieren oder keine Dezimalziffer (die nur zwischen 0 und 9 liegen dürfen) enthalten, soll das zurückzugebene Array an dieser Stelle mit Nullen aufgefüllt werden.

**Lösung:**

```
public static int[] threeDecimals(int[] zahlen){
    int[] decimals = new int[3];
    int counter = 0; // count names
    for(int i = 0; i < zahlen.length && counter < 3; i++){
        // check if decimal
        if(zahlen[i] >= 0 && zahlen[i] < 10){
            decimals[counter] = zahlen[i];
            counter++;
        }
    }
    return decimals;
}
```

**Aufgabe 4 (Schleifen (MC)).**

Es soll die Java-Methode `sumArray` vervollständigt werden, welche die Summe der Zahlen in dem übergebenen Array `zahlen` zurückgibt. Betrachten Sie dazu den folgenden Quellcode:

```
0 public static int sumArray(int zahlen[]){
1     int sum = 0;
2     for(___ (1) ___; ___ (2) ___; ___ (3) ___){
3         ___ (4) ___
4     }
5     return sum;
6 }
```

Geben Sie an, was bei den Stellen \_\_\_ (1) \_\_\_ bis \_\_\_ (4) \_\_\_ eingetragen werden soll.

1. Stelle \_\_\_ (1) \_\_\_:

- `int i=0`
- `int i=1`
- `i<zahlen.length`
- `i++`

2. Stelle \_\_\_ (2) \_\_\_:

- `i<zahlen.length`
- `i<<zahlen.length`
- `i>zahlen.length`
- `i>=zahlen.length`

3. Stelle \_\_\_ (3) \_\_\_:

- `i++`
- `i+=zahlen.length`
- `i<zahlen.length`
- `i=1`

4. Stelle \_\_\_ (4) \_\_\_:

- `sum += zahlen[i];`
- `sum++;`
- `sum = zahlen[i];`
- `sum = zahlen[i]++;`



**Aufgabe 5 (Rekursion).**

Betrachten Sie die rekursiv definierte Funktion für alle  $n \geq 3$

$$f(3) = 1$$

$$f(4) = 2$$

$$f(n) = 2 \cdot f(n-1) + 3 \cdot f(n-2) \quad \text{für } n > 4$$

Implementieren Sie eine rekursive Java-Methode, die den Wert  $f(n)$  zurückliefert. Die Variable  $n$  wird hierbei als Parameter der Methode übergeben. Für alle nicht definierten  $n$  soll 0 zurückgegeben werden. Die Deklaration von lokalen Variablen und Verwendung von Schleifen ist nicht erlaubt. Um welche Art von Rekursion handelt es sich? Begründen Sie Ihre Antwort.

**Lösung:**

```
static int f(int n) {
    if(n < 3) {
        return 0;
    } if(n == 3) {
        return 1;
    } else if(n == 4) {
        return 2;
    } else {
        return 2*f(n-1) + 3*f(n-2);
    }
}
```

Art der Rekursion: Nichtlineare/kaskadenförmige/baumartige Rekursion, da es mehr als einen rekursiven (nicht konstanten) Aufruf pro Methodenaufruf gibt.

## Aufgabe 6 (Klassen und Objekte).

1. Ergänzen Sie die folgende Java-Klasse wie in den Kommentaren im Quellcode beschrieben.

**Lösung:**

```
public class C {
    private int a;

    public C(int a) {
        // Initialisiere Attribut mit uebergebenem Parameter
        this.a = a;
    }

    public void f() {
        a *= 2;
        System.out.println("a = " + a);
    }

    public static void main(String[] args) {
        // Erzeuge ein Objekt der Klasse C. Dabei soll der Wert des
        // Attributs a mit 5 initialisiert werden.
        C c = new C(5);

        // Rufe die Methode f() auf.
        c.f();
    }
}
```

2. Gegeben ist die Klasse Fortbewegungsmittel.

```
public class Fortbewegungsmittel {
    // Attribute
    private int speed;
    private String besitzer;

    // Konstruktor
    public Fortbewegungsmittel(String besitzer){
        this.besitzer = besitzer;
        speed = 0;
    }

    // weitere Methoden

    // Getter für speed
    public int getSpeed(){
        return speed;
    }

    // Setter für speed
    public void beschleunigen(int deltaSpeed){
        speed += deltaSpeed;
    }
}
```

Implementieren Sie eine Klasse `Fahrrad`, welche von `Fortbewegungsmittel` erbt. Die Zugriffskriterien für Konstruktoren, Methoden und Attribute sind wie in der Superklasse. Die Klasse `Fahrrad` soll folgenden Anforderungen genügen:

- (a) Sie besitzt zwei zusätzliche Attribute: `gang` vom Typ `int` sowie `locked` vom Typ `boolean`.
- (b) Sowohl die Geschwindigkeit (`speed`) als auch der derzeitige Gang (`gang`) dürfen nicht negativ werden.
- (c) Die Klasse besitzt einen parametrisierten Konstruktor, der Parameter zum Setzen der Attribute `besitzer`, `gang` und `locked` übergeben bekommt. Die Geschwindigkeit soll (wie in `Fortbewegungsmittel`) mit 0 initialisiert werden.
- (d) Die Klasse besitzt einen weiteren parameterlosen Konstruktor, der die Attribute `gang` und `speed` mit 0, das Attribut `locked` mit `false` und den Besitzer mit dem String "Erika Mustermann" initialisiert.
- (e) Um die zuvor erwähnten Eigenschaften (siehe b)) des Attributs `speed` zu erfüllen, muss die Methode `beschleunigen(...)` der Superklasse überschrieben werden.
- (f) Eine Setter-Methode für `gang` soll implementiert werden, welche den Eigenschaften des Attributs (siehe b)) gerecht wird.

**Lösung:**

```

0 // Die Kommentare sollen zum Verstaendnis dienen und sind in der Klausur nicht
1 // gefordert. Gefordert ist nur gute Lesbarkeit und ein Verstaendnis des Codes.
2
3 public class Fahrrad extends Fortbewegungsmittel{
4     // zusaetzliche Attribute fuer Fahrrad
5     private int gang;
6     private boolean locked;
7
8     // parametrisierter Konstruktor
9     public Fahrrad(String besitzer, int gang, boolean locked){
10         super(besitzer); // speed wird automatisch auf 0 gesetzt
11
12         if(gang < 0){
13             gang = 0; // falls gang negativ, wird Variable Null gesetzt
14         }
15         this.gang = gang;
16
17         this.locked = locked;
18     }
19
20     // parameterloser Konstruktor
21     public Fahrrad(){
22         this("Erika Mustermann", 0, false);
23     }
24
25     // weitere Methoden
26
27     // Setter für gang
28     public void setGang(int gang){
29         // nur wenn gang positiv oder 0, wird Variable neu gesetzt
30         if(gang >= 0){
31             this.gang = gang;
32         }
33         // alternativ: gang auf 0 setzen bei negativen Uebergabewerten
34     }
35
36     // ueberschriebene Methode bschleunigen
37     // damit Fahrrad NICHT rueckwaerts fahren kann
38     public void beschleunigen(int deltaSpeed){
39         if(getSpeed()+deltaSpeed < 0){
40             // setzt speed auf 0 in Superklasse, damit nicht negativ
41             super.beschleunigen(-getSpeed());
42         }else{

```

```
43         super.beschleunigen(deltaSpeed);
44     }
45     // alternativ: speed bei negativen Werten nicht veraendern
46     // (siehe Setter fuer gang)
47 }
48 }
```

**Aufgabe 7 (Zahlensysteme).**

1. Rechnen Sie die Zahl  $423_{(6)}$  in eine Zahl zur Basis 16 um. Der Lösungsweg muss erkennbar sein.

**Lösung:**

$$423_{(6)} = 4 \cdot 6^2 + 2 \cdot 6^1 + 3 \cdot 6^0 = 144 + 12 + 3 = 159_{(10)}$$

$$159 : 16 = 9 \text{ Rest } 15$$

$$9 : 16 = 0 \text{ Rest } 9$$

$$\Rightarrow 423_{(6)} = 9F_{(16)}$$

2. Berechnen Sie die folgenden Aufgaben, dargestellt in Basis 10, unter Verwendung der Zweierkomplementdarstellung mit 4 Bit. Der Lösungsweg muss erkennbar sein. Bei welchen Aufgaben findet ein Über- bzw. Unterlauf statt und woran erkennen Sie diesen jeweils?

(a)  $6 + (-2)$

(b)  $-7 + (-2)$

(c)  $4 + 7$

**Lösung:**

(a)  $6_{(10)} = 110_{(2)} = 0110_{(2K)}$

über Hinzufügen des Vorzeichenbits 0 und Auffüllen mit 0-en (für positive Zahlen)

$2_{(10)} = 10_{(2)} = 0010_{(2K)} \Rightarrow -2_{(10)} = 1110_{(2K)}$

über Darstellung der positiven Zahl, Invertierung dieser und Addition von 1 (für negative Zahlen)

$$\begin{array}{r|cccc} & 0 & 1 & 1 & 0 \\ + & 1_1 & 1_1 & 1_1 & 0 \\ \hline & 0 & 0 & 1 & 0 & 0 \end{array}$$

Ergebnis:  $6 + (-2) = 4$ : Korrekt, da die ersten beiden Bits im (erweiterten) Ergebnis gleich sind.

(b)  $7_{(10)} = 0111_{(2K)} \Rightarrow -7_{(10)} = 1001_{(2K)}$

$2_{(10)} = 0010_{(2K)} \Rightarrow -2_{(10)} = 1110_{(2K)}$

$$\begin{array}{r|cccc} & 1 & 0 & 0 & 1 \\ + & 1_1 & 1_1 & 1_1 & 0 \\ \hline & 1 & 0 & 1 & 1 & 1 \end{array}$$

Ergebnis:  $-7 + (-2) = 7$ : Unterlauf, da die ersten beiden Bits im (erweiterten) Ergebnis 10 sind.

(c)  $4_{(10)} = 0100_{(2K)}$

$7_{(10)} = 0111_{(2K)}$

$$\begin{array}{r|cccc} & 0 & 1 & 0 & 0 \\ + & 0 & 0_1 & 1 & 1 & 1 \\ \hline & 0 & 1 & 0 & 1 & 1 \end{array}$$

Ergebnis:  $4 + 7 = -5$ : Überlauf, da die ersten beiden Bits im (erweiterten) Ergebnis 01 sind.

**Aufgabe 8 (Fließkommazahlen).**

1. Addieren Sie die Zahlen  $45_{(10)}$  und  $0.0625_{(10)}$  im angegebenen Gleitkommaformat und konvertieren Sie das Ergebnis zurück in eine Dezimalzahl.

Vorzeichen	Exponent	Mantisse
1 Bit	4 Bits	5 Bits

Der Exzess beträgt:  $7_{(10)} = 0111_{(2)}$

**Lösung:**

- (a) Dualzahlen:  $45_{(10)} = 101101_{(2)}$ ,  $0.0625_{(10)} = 0.0001_{(2)}$   
 (b) Normalisierung:  $1.01101_{(2)} \cdot 10_{(2)}^{101}$ ,  $1.00000000_{(2)} \cdot 10_{(2)}^{-100}$   
 (c) Exponent in Exzessdarstellung:  $0101_{(2)} + 0111 = 1100_{Exzess}$ ,  $-0100_{(2)} + 0111 = 0011_{Exzess}$

Vorzeichen	Exponent	Mantisse
0	1100	01101
0	0011	00000

- (d) Addition:

- kleineren Exponenten anpassen:  $1.00000_{(2)} \cdot 10_{(2)}^{-100} = 0.00000_{(2)} \cdot 10_{(2)}^{101}$
  - Addition der Mantissen
- |   |         |
|---|---------|
|   | 1.01101 |
| + | 0.00000 |
|   | 1.01101 |
- Zusammenfügen zum Gleitkommaformat: 0 1100 01101

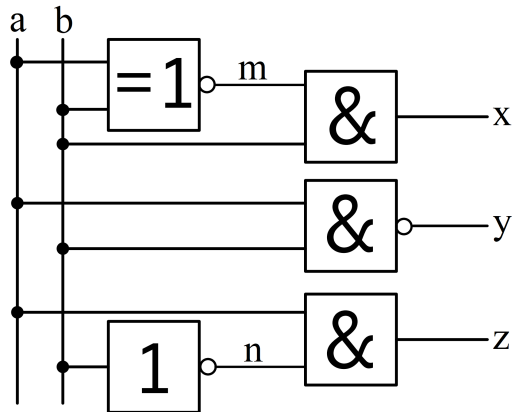
2. Wie groß ist der Rundungsfehler, der bei der Addition in der ersten Teilaufgabe entsteht. Geben Sie den Rundungsfehler als Dezimalzahl an.

**Lösung:**

Die Differenz der Exponenten ist  $9_{(10)}$ , daher fällt das Bit heraus und das Ergebnis ist wieder  $45_{(10)}$ . Der Fehler beträgt demzufolge  $0.0625_{(10)}$ .

**Aufgabe 9 (Schaltungen).**

1. Stellen Sie die Wertetabelle für folgende Schaltung auf:



(a und b sind die Eingänge, x, y und z sind die Ausgänge der Schaltung.)

**Lösung:**

Zunächst werden die Zwischenpunkte  $m := \overline{a \oplus b}$  und  $n := \overline{a}$  eingeführt. Dann ergibt sich folgende Wertetabelle:

a	b	m	n	x	y	z
0	0	1	1	0	1	0
0	1	0	1	0	1	0
1	0	0	0	0	1	1
1	1	1	0	1	0	0

2. Zeichnen Sie eine Schaltung, die der folgenden Wertetabelle entspricht.

a	b	c	y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

(a, b und c sind die Eingänge, y ist der Ausgang der Schaltung.)

**Lösung:**

