

# Probeklausur: Programmierung WS04/05

Name:	
Matrikelnr.:	

## Hinweise zur Bearbeitung

- Nimm Dir für diese Klausur ausreichend Zeit, und Sorge dafür, dass Du nicht gestört wirst.
- Die Klausur ist für 90 Minuten angesetzt, versuche diese Zeit einzuhalten. Am Ende der Klausur solltest Du Dir einen kleinen Puffer lassen, um noch einmal alle Antworten durchzulesen.
- Lies Dir die Aufgabenstellungen aufmerksam und sorgfältig durch.
- Während der richtigen Klausur sind keine weiteren Unterlagen oder Hilfsmittel erlaubt. Versuche deshalb auch die Probeklausur zu bearbeiten, ohne Deine Notizen, Lehrbücher oder die API zu konsultieren.
- Die maximal zu erreichende Punktzahl ist 60.



## 1 Fragen zum Allgemeinen Verständnis (15 Punkte)

1. Im Folgenden sind vier verschiedene Implementierungen der Methode `umfang()` einer Klasse `Kreis` gegeben. Welche der Implementierungen funktionieren fehlerfrei mit dem gegebenen Hauptprogramm? Gib bei den nicht funktionierenden Implementierungen jeweils in maximal einem Satz an, wieso sie nicht funktionieren.

Listing 1: KreisTest.java

```
1 package Irgendwo;
2 import Geometrie.*;
3
4 public class KreisTest {
5     public static void main(String [] args) {
6         Kreis k = new Kreis(5);
7         double u = k.umfang()
8     }
9 }
```

Listing 2: Kreis.java

```
1 package Geometrie;
2 public class Kreis{
3     protected double r;
4     public Kreis(double r) {
5         this.r = r;
6     }
7
8     /* umfang() */
9 }
```

- (a) `protected double umfang() {  
 return 2*r*Math.PI; }`
- (b) `public static double umfang() {  
 return 2*r*Math.PI; }`
- (c) `public double umfang() {  
 return 2*r*Math.PI; }`
- (d) `public static double umfang(double r) {  
 return 2*r*Math.PI; }`

Name:

Matrikelnr.:

---

- (a) *Funktioniert nicht. Auf Methoden mit Sichtbarkeit `protected` kann ausserhalb des Pakets nur aus Klassen zugegriffen werden, die `Kreis` erweitern.*
- (b) *Funktioniert nicht. In einer statischen Methode kann man nicht auf Instanzvariablen (z.B. `r`) zugreifen.*
- (c) *Funktioniert.*
- (d) *Funktioniert nicht. Die Methode wird in `KreisTest` ohne Parameter aufgerufen, in der Signatur wird aber ein Parameter verlangt.*

2. Es seien die folgenden Klassendeklarationen und Objektinstanziierungen gegeben.

```
class ErsteKlasse { int a; }
class ZweiteKlasse extends ErsteKlasse { int b; }
class DritteKlasse extends ZweiteKlasse { int c; }
```

```
ErsteKlasse eins = new ErsteKlasse();
ZweiteKlasse zwei = new ZweiteKlasse();
DritteKlasse drei = new DritteKlasse();
```

Welche der folgenden Aussagen sind dann richtig?

- `drei instanceof ErsteKlasse` ergibt `true`.
  - `eins = drei;` ist zulässig.
  - `zwei.a = eins.a;` ist *nicht* zulässig.
  - `zwei = (DritteKlasse) eins;` ist zulässig.
3. Welches Java-Keyword beinhaltet die Referenz auf das aktuelle Objekt? Gib ein Beispiel für die typische Benutzung dieses Keywords an.

`this`.

Zum Beispiel wird `this` oft in Setter-Methoden benutzt, um explizit die Objektvariable zu referenzieren und diese von einem Parameter zu unterscheiden:

```
public void setName (String name) { this.name =
name; }
```

Name:

Matrikelnr.:

4. Bei den folgenden Behauptungen gibt ein richtig angekreuztes Kästchen 1 Punkt, ein falsch angekreuztes führt zu 1 Punkt Abzug.

Behauptung	Ja	Nein
Alle Java-Klassen haben eine gemeinsame direkte oder indirekte Oberklasse.	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Eine Java-Klasse kann mehrere Konstruktoren besitzen.	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Abstrakte Klassen müssen mindestens eine abstrakte Methode besitzen.	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Unter einer „überladenen“ Methode versteht man zwei Methoden gleichen Namens in derselben Klasse, mit gleichen Rückgabetypen und unterschiedlichen Parametern.	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Eine Klasse kann beliebig viele Schnittstellen implementieren.	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Eine Klasse kann von beliebig vielen Klassen erben.	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Die Anweisungsfolge <code>int[] a = new int[10]; a[10]=10;</code> wird vom Compiler akzeptiert.	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Die Anweisungsfolge <code>int[] a = new int[10]; a[10]=10;</code> liefert eine Ausnahme zur Laufzeit.	<input checked="" type="checkbox"/>	<input type="checkbox"/>
<code>static</code> -Methoden können nur aus <code>static</code> -Methoden aufgerufen werden.	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Ein Konstruktor kann alle nichtprivaten Methoden der aktuellen Klasse und ihrer Oberklasse aufrufen.	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Name:

Matrikelnr.:

---

## 2 Grafische Oberflächen und Eventmodell (5 Punkte)

Betrachte die abgebildete Benutzeroberfläche bestehend aus einer Anzeige und einem Knopf.



Erweitere das folgende Programm in den Zeilen 6 und 18, sowie in der Methode `actionPerformed`, so dass bei jedem Drücken des Knopfes "OK" der Zähler im Textfeld um 1 erhöht wird.

Name:

Matrikelnr.:

---

Listing 3: ZaehlKnopf.java

```
1 import java.awt.*;
2 import java.awt.event.*;
3 import javax.swing.*;
4
5 public class ZaehlKnopf extends JPanel
6 implements ActionListener
7 {
8     protected int zaehler = 0;
9     protected JTextField anzeige;
10    protected JButton knopf;
11
12    ZaehlKnopf() {
13        super(new FlowLayout());
14        setSize(new Dimension(100,100));
15        anzeige = new JTextField(3);
16        anzeige.setText(""+zaehler);
17        JButton knopf = new JButton("OK");
18        knopf.addActionListener(this);
19        add(anzeige);
20        add(knopf);
21    }
22
23    public void actionPerformed(ActionEvent evt) {
24        zaehler++;
25        anzeige.setText(""+zaehler);
26    }
27
28
29    public static void main(String [] args) {
30        JFrame frame = new JFrame("Zaehlknopf");
31        frame.setDefaultCloseOperation(
32            JFrame.EXIT_ON_CLOSE);
33        frame.setContentPane(new ZaehlKnopf());
34        frame.pack();
35        frame.setVisible(true);
36    }
37 }
```

Name:

Matrikelnr.:

---

### 3 Zuweisungen und Operatoren (10 Punkte)

Bei den folgenden Fragen seien jeweils die Variablen wie folgt vorausgesetzt:

```
int a = 2;  
int b = 10;  
double c = 3.0;  
double d = 1.0;  
boolean e = false;
```

Berechne die folgenden Ausdrücke und gib für jeden die Reihenfolge an, in der die Operatoren gemäß den Präzedenzen in Java ausgewertet werden. Gib außerdem den Wert und den Datentyp des Ergebnisses an. In der ersten Zeile der Tabelle ist ein Beispiel angegeben.

Ausdruck	1	2	3	4	5	Wert	Datentyp
Beispiel: $a = 3 + 4 * 2$	*	+	=			11	int
$3 * 8 / 3 / 2$	*	/	/			4	int
$(int) - (c / ++d)$	++	/	-	(int)		-1	int
$c = b \% a$	%	=				0.0	double
$a ++ - b * c$	++	*	-			-28.0	double
$e = 16 >> 1 >> 2 \% 2 == 8$	%	>>	>>	==	=	true	boolean

## 4 Sprachkonstrukte in Java (10 Punkte)

1. Welche Bildschirmausgabe erzeugt der folgende Java-Code?

```
String text="test";

int i=0;

while (i<text.length()) {
    if (i % 2 != 0)
        System.out.print(text.charAt(i));
    else
        System.out.print(
            (""+text.charAt(i)).toUpperCase());
    i++;
}

TeSt
```

2. Ergänze die folgende for-Schleife so, dass die Variable a nach Ausführung der Schleife die Summe der Zahlen von 1 bis 10 enthält:

```
int a = 0;

for (int zaehler = 1; zaehler <= 10; zaehler++) {
    a+=zaehler;
}

oder:

int a = 0;

for (int zaehler = 0; zaehler <10; zaehler++) {
    a += zaehler+1;
}
```

**Name:**

**Matrikelnr.:**

---

3. Gegeben seien die beiden folgenden für positive ganze Zahlen (größer 0) definierten Funktionen:

$$g(n) = n - 1$$

$$h(n) = n! \quad (\text{gemeint ist die Fakultätsfunktion: } n! = 1 \cdot 2 \cdot \dots \cdot n)$$

sowie die Funktion

$$f(m, 0) = m \quad \text{für } n = 0$$

$$f(m, n) = f(h(g(n)), g(n)) \quad \text{für } n > 0$$

Schreibe jeweils eine Java-Methode, die  $g$ ,  $h$  und  $f$  implementiert. Gehe dabei davon aus, dass nur korrekte Parameter übergeben werden.

```
public static int g ( int n ) {  
  
    return n-1;  
}  
  
public static int h ( int n ) {  
  
    int fak=1;  
    for (int i=1; i<= n; i++) fak *=i;  
    return fak;  
  
}  
  
public static int f ( int m, int n ) {  
  
    if (n==0) return m;  
    else return f(h(g(n)),g(n));  
  
}
```

## 5 Objektorientiertes Programmieren (10 Punkte)

Definiere eine Java-Klasse zur Simulation der Senderauswahl bei einem Fernseher. Der Fernseher hat eine feste Zahl an Speicherplätzen (z.B. 30), die beginnend mit 0 durchgehend numeriert sind.

Am Fernseher wird jeweils der Sender zum aktuell ausgewählten Speicherplatz angezeigt. Deine Klasse soll die folgenden Operation unterstützen:

1. Wechsel auf den nächsten Speicherplatz bzw. Sender (Zapping). Auf den letzten Speicherplatz folgt wieder der erste.
2. Speichern eines Sendernamens (z.B. ARD) für den aktuellen Speicherplatz.
3. Ausgabe des Sendernamens für den aktuellen Speicherplatz.
4. Suchen des Speicherplatzes zu einem Sendernamen; und Wechsel zu diesem Sender, falls ein solcher Speicherplatz existiert.

Schreibe eine passende Klassendefinition mit Instanzvariablen, geeignetem Konstruktor und den Signaturen für alle benötigten Methoden. Implementiere *eine* der Methoden nach Deiner Wahl.

Listing 4: KreisTest.java

```
1 public class Fernseher {
2
3     private int aktuellerSender;
4     private String [] senderKennungen;
5
6     Fernseher(int senderZahl) {
7         senderKennungen = new String[senderZahl];
8         for (int i=0; i<senderZahl; i++)
9             senderKennungen[i] = "";
10        aktuellerSender = 0;
11    }
12
13    /* nicht in der Aufgabe gefordert */
14    private boolean angeschaltet;
15    public void anschalten() { angeschaltet=true;}
16    public void ausschalten() { angeschaltet=false;}
17
18    public void zap() {
19        if (aktuellerSender == senderKennungen.length-1)
```

Name:

Matrikelnr.:

---

```
20     aktuellerSender = 0;
21     else
22         aktuellerSender++;
23     }
24
25     public void setzeKennung (String kennung) {
26         senderKennung[aktuellerSender] = kennung;
27     }
28
29     public String gibKennung () {
30         return senderKennung[aktuellerSender];
31     }
32
33     /*
34     * liefere gefundenen Speicherplatz zurueck ,
35     * oder -1 wenn Sender nicht gefunden
36     */
37     public int waehleSender (String kennung) {
38
39         for (int i=0; i<senderKennungen.length; i++) {
40             if (senderKennungen[i].equals(kennung)) {
41                 aktuellerSender = i;
42                 return i;
43             }
44         }
45         return -1;
46     }
47 }
```

*Gefordert war nur eine Implementation, z.B. von gibKennung(). Für die restlichen Methoden genügt die Angabe der Signatur:*

```
public void setzeKennung(String name)
```

## 6 Vererbung (10 Punkte)

1. Schreibe drei Klassen zu folgendem Problem: Es gibt Bücher, zu denen der Autorenname, der Titel und die Sprache festgehalten werden soll. Gedruckte Bücher und Elektronische Bücher sind Spezialisierungen von Büchern. Gedruckte Bücher haben zusätzlich eine Seitenzahl und sind entweder ein Taschenbuch oder nicht. Elektronische Bücher haben eine Dateigröße und ein Format.

Die Klassen sollen folgende **abstrakte** Methoden haben:

- Buch: drucken()
- ElektronischesBuch: speichern()
- GedrucktesBuch: kopieren()

*Achtung: Klassen müssen als abstrakt definiert werden !*

```
abstract class Buch {
    String autor;
    String sprache;
    String titel;
    abstract public void drucken();
}

abstract class GedrucktesBuch extends Buch {
    int seiten;
    boolean taschenbuch;
    abstract public void kopieren();
}

abstract class ElektronischesBuch extends Buch {
    int dateigroesse;
    String format;
    abstract public void speichern();
}
```

2. Es sei eine Klasse nicht-abstrakte `WebArtikel` als Erweiterung von `ElektronischesBuch` gegeben, sowie eine nicht-abstrakte Klasse `Lehrbuch`, die `GedrucktesBuch` erweitert.

Erstelle nun ein `Buch[]`-Array und fülle es mit vier Instanzen von Klassen, die Du in diesem Array speichern kannst.

**Name:**

**Matrikelnr.:**

---

```
Buch[] array = new Buch[4];  
array[0] = new WebArtikel();  
array[1] = new WebArtikel();  
array[2] = new Lehrbuch();  
array[3] = new Lehrbuch();
```

*Achtung: von den abstrakten Klassen Buch, GedrucktesBuch, ElektronischesBuch dürfen keine Instanzen erzeugt werden*

3. Welche Methode besitzen alle in (2) erzeugten Instanzen? Schreibe eine Schleife, die für alle Elemente des Arrays diese Methode ausführt.

```
drucken()  
  
for (int i=0; i<array.length; i++)  
array[i].drucken();
```

4. Wann wird festgestellt, aus welcher Klassendefinition die auszuführende Methode herangezogen wird? Wie nennt man das dahinterstehende Konzept?

*Zur Laufzeit des Programms. Polymorphie.*