

Vorname

Nachname

Matrikel-Nr

Diese Klausur ist mein dritter Prüfungsversuch (bitte ankreuzen): Ja ☐ Nein ☐

Schreiben Sie jede Lösung auf die Vorderseite eines *neuen Blattes* (und lassen Sie die Rückseiten Ihrer Lösungsblätter *leer*). Die Aufgaben 5 und 6 stehen *auf der Rückseite* dieses Blattes!

Aufgabe 1 (15 Punkte): Schreiben Sie eine Methode entsprechend der folgenden Spezifikation:

```
1  static public int quersumme(int zahl, short B) {
2      // Welche Quersumme hat die zahl als B-er-Zahl (d.h. wenn man sie
3      // im Zahlensystem mit der Basis B darstellt)? Diese Funktion liefert
4      // die Antwort.
5      // Falls die zahl negativ ist, wird die Quersumme von -zahl als
6      // Ergebnis geliefert.
7      // Falls die Basis B nicht im Bereich 2 bis Short.MAX_VALUE liegt,
8      // wird -1 als Ergebnis geliefert.
9
10     // Beispiele:
11     // short b01 = 1;           // Eine zu kleine Basis
12     // short b02 = 2;           // Die kleinste zulaessige Basis
13     // short b10 = 10;
14     // short b16 = 16;
15     // short bmax = Short.MAX_VALUE; // Die groesste zulaessige Basis
16     //
17     // quersumme(+123, b10) ist gleich 6
18     // quersumme(-123, b10) ist gleich 6
19     // quersumme( +51, b10) ist gleich 6
20     // quersumme(  +7, b02) ist gleich 3
21     // quersumme(  +8, b02) ist gleich 1
22     // quersumme( +16, b16) ist gleich 1
23     // quersumme( +16, bmax) ist gleich 16
24     // quersumme(+123, b01) ist gleich -1 (weil b01 zu klein ist)
```

Aufgabe 2 (15 Punkte) Schreiben Sie zwei Methoden entsprechend den folgenden Spezifikationen:

```
1  static public boolean istGrossbuchstabe(char zeichen) {
2      // Liefert true, wenn zeichen ein Grossbuchstabe ist.
3      // Als Grossbuchstaben gelten hier die Zeichen 'A', 'B', ..., 'Z'.
4      // Der Rumpf dieser Methode darf nur einfache Anweisungen enthalten!
5      ...
6  } // istGrossbuchstabe
7
8  static public int anzahlGrossbuchstaben(ArrayList<String> als) {
9      // Liefert die Anzahl der Grossbuchstaben ('A', 'B', ..., 'Z')
10     // die in den Komponenten von als enthalten sind.
11     // Beispiel:
12     // ArrayList<String> a01 = new ArrayList<String>();
13     // als.add("Hallo Harry!");
14     // als.add("Wie geht's?");
15     // als.add("Danke, gut!");
16     //
17     // Fuer die Sammlung a01 gilt jetzt:
18     // anzahlGrossbuchstaben(a01) ist gleich 4 (H, H, W und D).
19     ...
20 } // anzahlGrossbuchstaben
```

Natürlich sollen Sie in der Methode `anzahlGrossbuchstaben` die Methode `istGrossbuchstabe` aufrufen.

Aufgabe 3 (15 Punkte): Stellen Sie die folgende Reihung `r2s` als Boje dar und beantworten die nachfolgenden Fragen:

```
1  String[][] r2s = {
2      {"AB", null, "CD"},
3      null,
4      {null},
5      {}
6  };
```

Frage 3.1: Welche Länge hat die Reihung `r2s`?

Frage 3.2: Welche Länge hat die Reihung `r2s[0]`?

Frage 3.3: Welche Länge hat die Reihung `r2s[2]`?

Frage 3.4: Welche Länge hat die Reihung `r2s[3]`?

Aufgabe 4 (15 Punkte): Betrachten Sie die folgenden Klassenvereinbarungen:

```
1  class K1 {
2      static private int sum1 = 0;
3      static private int sum2 = 0;
4      private int zal1;
5      private int zal2;
6
7      public K1(int zal1, int zal2) {
8          this.zal1 = zal1;
9          this.zal2 = zal2;
10         sum1 += zal1;
11         sum2 += zal2;
12     }
13
14     public K1() {}
15 }
16
17 class K2 {
18     static public void main(String[] _) {
19         K1 ob1 = new K1(10, 20);
20         K1 ob2 = new K1();
21         K1 ob3 = new K1();
22
23         K1 ob4 = new K1(30, 40);
24         K1 ob5 = new K1();
25
26     } // main
27 }
```

4.1. Wie viele *Klassenelemente* werden in der Klasse `K1` vereinbart?

4.2. Wie viele *Elemente* werden in jedes `K1`-Objekt eingebaut?

4.3. Wie viele *Konstruktoren* werden in der Klasse `K1` vereinbart?

4.4. Sei `M1` der Moment, in dem der Ausführer die Zeile 21 fertig ausgeführt hat. Wie viele Module existieren in diesem Moment `M1` und wie heißen diese Module?

4.5. Wie viele `int`-Variablen existieren im Moment `M1` und wie heißen diese Variablen mit vollen Namen?

4.6. Sei `M2` der Moment, in dem der Ausführer die Zeile 24 fertig ausgeführt hat. Wie viele Module existieren in diesem Moment `M2` und wie heißen diese Module?

4.7. Wie viele `int`-Variablen existieren im Moment `M2` und wie heißen diese Variablen mit vollen Namen?

Aufgabe 5 (15 Punkte): Geben Sie von jeder der folgenden 4 Schleifen an, was sie zum Bildschirm ausgibt.

```
1      final int Z = 3;
2
3      // Schleife 5.1.:
4      for (int i=1; i<=Z; i++) {
5          for (int j=i; j>=1; j--) p("XX");
6          pln();
7      }
8
9      // Schleife 5.2:
10     for (int i=0; i<2*Z; i++) {
11         for (int j=i%Z+1; j>=1; j--) p("XX");
12         pln();
13     }
14
15     // Schleife 5.3:
16     for (int i=1; i<=Z; i++) {
17         for (int j=i-1; j>=1; j--) p("00");
18         pln("XX");
19     }
20
21     // Schleife 5.4:
22     for (int i=1; i<=Z; i++) {
23         for (int j=Z-i; j>=1; j--) p("00");
24         for (int j=1; j<=i+(i-1); j++) p("XX");
25         for (int j=Z-i; j>=1; j--) p("00");
26         pln();
27     }
```

Die Prozedur `p` gibt (wie in zahlreichen Beispielprogrammen) ihren `String`-Parameter zum Bildschirm aus, *ohne* den Bildschirmzeiger (engl. `cursor`) zum Anfang der folgenden Zeile vorzuschieben. Die parameterlose Prozedur `pln` schiebt den Bildschirmzeiger zum Anfang der nächsten Zeile vor.

Aufgabe 6: (15 Punkte)

6.1. Wie Sie wahrscheinlich wissen, gehören (im Zusammenhang mit Grabo-Programmen) die Ereignis-Art `mouseWheelMoved`, die Ereignis-Oberart `Mausrad-Ereignis` und die Schnittstelle `MouseWheelListener` zusammen (die Ereignis-Art *gehört zur* Ereignis-Oberart und die Schnittstelle ist der Ereignis-Oberart *zugeordnet*). Geben Sie eine weitere Ereignis-Art, Ereignis-Oberart und Schnittstelle an, die genauso zusammengehören.

6.2. Erläutern Sie kurz, zu welchen *Listener-Schnittstelle* es eine *Adapter-Klasse* gibt und zu welchen nicht.

6.3. Geben Sie mit dem Befehl `System.out.print` ein *anonymes Objekt* aus. Den Typ des Objekts dürfen Sie frei wählen.

6.4. Welche der folgenden Sätze treffen zu und welche nicht?

6.4.1. *Prozesse* (eines Betriebssystems) werden immer *gleichzeitig* und *Fäden* (engl. threads of control) werden ebenfalls immer *gleichzeitig* ausgeführt.

6.4.2. *Prozesse* werden immer *gleichzeit* und *Fäden* werden immer *nebenläufig* zueinander ausgeführt.

6.4.3. *Prozesse* werden immer *nebenläufig* zueinander und *Fäden* werden immer *gleichzeitig* ausgeführt.

6.4.4. *Prozesse* werden immer *nebenläufig* zueinander und *Fäden* werden ebenfalls immer *nebenläufig* zueinander ausgeführt.

6.4.5. *Fäden* werden manchmal *gleichzeitig* und manchmal *nacheinander* ausgeführt.

6.5. Wie Sie wahrscheinlich wissen, beschreibt die kontextfreie Grammatik der Sprache Java nicht genau die Sprache Java, sondern eine etwas andere Sprache Java' ('Java Strich'). Erläutern Sie kurz, welche Worte zu der Sprache Java' gehören.

6.6. Betrachten Sie die folgende Regel einer kontextfreien Grammatik:

FieldDecl : {"static" | "final" | "public" | "private"} Type Var {"", " Var"} ";"

Welche der folgenden Symbolfolgen kann man mit dieser Regel aus dem Zwischensymbol (engl. non-terminal symbol) FieldDecl ableiten und welche nicht?

6.6.1. "static" "final" "public" Type Var ";"

6.6.2. "final" "static" "public" Type Var ";"

6.6.3. Type Var

6.6.4. "private" "private" Type Var ";"

6.6.5. "static" Type Var ", " Var ", " Var ", "

Lösung 1 (15 Punkte): Schreiben Sie eine Methode entsprechend der folgenden Spezifikation:

```
1  static public int quersumme(int zahl, short B) {
2      // Welche Quersumme hat die zahl als B-er-Zahl (d.h. wenn man sie
3      // im Zahlensystem mit der Basis B darstellt)? Diese Funktion liefert
4      // die Antwort.
5      // Falls die zahl negativ ist, wird die Quersumme von -zahl als
6      // Ergebnis geliefert.
7      // Falls die Basis B nicht im Bereich 2 bis Short.MAX_VALUE liegt,
8      // wird -1 als Ergebnis geliefert.
9
10     if (B < 2) return -1;
11
12     int erg = 0;
13     int ziff;
14     zahl=Math.abs(zahl);
15
16     while (true) {
17         if (zahl == 0) break;
18         ziff = zahl % B;
19         zahl = zahl / B;
20         erg += ziff;
21     }
22
23     return erg;
24 } // quersumme
```

Lösung 2 (15 Punkte) Schreiben Sie zwei Methoden entsprechend den folgenden Spezifikationen:

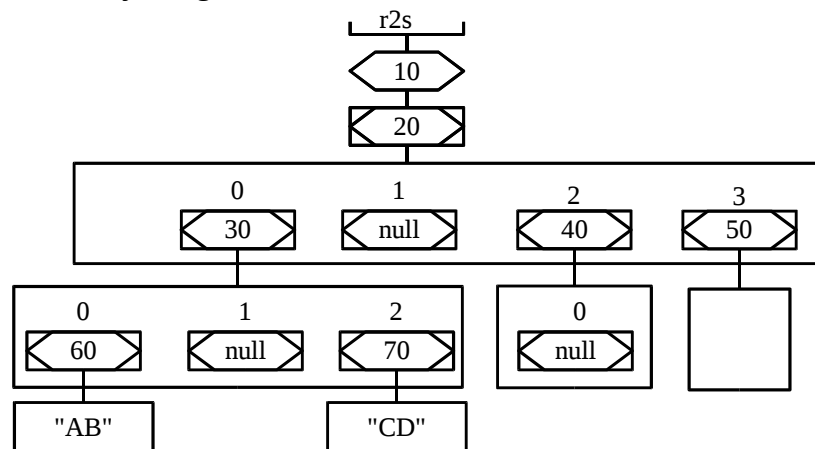
```
1  static public boolean istGrossbuchstabe(char zeichen) {
2      // Liefert true, wenn zeichen ein Grossbuchstabe ist.
3      // Als Grossbuchstaben gelten hier die Zeichen 'A', 'B', ..., 'Z'.
4
5      return 'A' <= zeichen && zeichen <= 'Z';
6  } // istGrossbuchstabe
7
8  static public int anzahlGrossbuchstaben(ArrayList<String> als) {
9      // Liefert die Anzahl der Grossbuchstaben ('A', 'B', ..., 'Z')
10     // die in den Komponenten von als enthalten sind.
11     int erg = 0;
12     for (String s: als) {
13         for (int i=0; i<s.length(); i++) {
14             if (istGrossbuchstabe(s.charAt(i))) erg++;
15         }
16     }
17     return erg;
18 } // anzahlGrossbuchstaben
```

Lösung 3 (15 Punkte): Stellen Sie die folgende Reihung r2s als Boje dar:

```
1  String[][] r2s = {
2      {"AB", null, "CD"},
3      null,
4      {null},
5      {}
6  };
```

- | | |
|---|---|
| Frage 3.1: Welche Länge hat die Reihung r2s? | 4 |
| Frage 3.2: Welche Länge hat die Reihung r2s[0]? | 3 |
| Frage 3.3: Welche Länge hat die Reihung r2s[2]? | 1 |
| Frage 3.4: Welche Länge hat die Reihung r2s[3]? | 0 |

Die Reihung r2s als Boje dargestellt:



Lösung 4 (15 Punkte): Betrachten Sie die folgenden Klassenvereinbarungen:

```

1  class K1 {
2      static private int sum1 = 0;
3      static private int sum2 = 0;
4      private int zal1;
5      private int zal2;
6
7      public K1(int zal1, int zal2) {
8          this.zal1 = zal1;
9          this.zal2 = zal2;
10         sum1 += zal1;
11         sum2 += zal2;
12     }
13
14     public K1() {}
15 }
16
17 class K2 {
18     static public void main(String[] _) {
19         K1 ob1 = new K1(10, 20);
20         K1 ob2 = new K1();
21         K1 ob3 = new K1();
22
23         K1 ob4 = new K1(30, 40);
24         K1 ob5 = new K1();
25
26     } // main
27 }

```

4.1. Wie viele *Klassenelemente* werden in der Klasse K1 vereinbart? 2

4.2. Wie viele *Elemente* werden in jedes K1-Objekt eingebaut? 2

4.3. Wie viele *Konstruktoren* werden in der Klasse K1 vereinbart? 2

4.4. Sei M1 der Moment, in dem der Ausführer die Zeile 21 fertig ausgeführt hat. Wie viele Module existieren in diesem Moment M1 und wie heißen diese Module?

5 Module, 2 Klassen (K1 und K2) und 3 Objekte (ob1, ob2, ob3).

4.5. Wie viele `int`-Variablen existieren im Moment M1 und wie heißen diese Variablen mit vollen Namen?

8 `int`-Variablen: K1.sum1, K1.sum2, ob1.zal1, ob1.zahl2, ob2.zal1, ob2.zal2, ob3.zal1, ob3.zal2.

4.6. Sei M2 der Moment, in dem der Ausführer die Zeile 24 fertig ausgeführt hat. Wie viele Module existieren in diesem Moment M2 und wie heißen diese Module?

7 Module, 2 Klassen (K1 und K2) und 5 Objekte (ob1, ob2, ob3, ob4, ob5).

4.7. Wie viele `int`-Variablen existieren im Moment M2 und wie heißen diese Variablen mit vollen Namen?

12 `int`-Variablen: K1.sum1, K1.sum2, ob1.zal1, ob1.zahl2, ... ob5.zal1, ob5.zal2.

Lösung 5 (15 Punkte): Geben Sie von jeder der folgenden 4 Schleifen an, was sie zum Bildschirm ausgibt.

```

1      final int Z = 3;
2
3      // Schleife 5.1.:
4      for (int i=1; i<=Z; i++) {
5          for (int j=i; j>=1; j--) p("XX");
6          pln();
7      }
8
9      // Schleife 5.2:
10     for (int i=0; i<2*Z; i++) {
11         for (int j=i%Z+1; j>=1; j--) p("XX");
12         pln();
13     }
14
15     // Schleife 5.3:
16     for (int i=1; i<=Z; i++) {
17         for (int j=i-1; j>=1; j--) p("00");
18         pln("XX");
19     }
20
21     // Schleife 5.4:
22     for (int i=1; i<=Z; i++) {
23         for (int j=Z-i; j>=1; j--) p("00");
24         for (int j=1; j<=i+(i-1); j++) p("XX");
25         for (int j=Z-i; j>=1; j--) p("00");
26         pln();
27     }

```

Ausgabe der Schleife 5.1.:

```

XX
XXXX
XXXXXX

```

Ausgabe der Schleife 5.2.:

```

XX
XXXX
XXXXXX
XX
XXXX
XXXXXX

```

Ausgabe der Schleife 5.3.:

```

XX
00XX
0000XX

```

Ausgabe der Schleife 5.4.:

```

0000XX0000
00XXXXXX00
XXXXXXXXXX

```

Lösung 6: (15 Punkte)

6.1. Wie Sie wahrscheinlich wissen, gehören (im Zusammenhang mit Grabo-Programmen) die Ereignis-Art `mouseWheelMoved`, die Ereignis-Oberart `Mausrad-Ereignis` und die Schnittstelle `MouseWheelListener` zusammen (die Ereignis-Art *gehört zur* Ereignis-Oberart und die Schnittstelle ist der Ereignis-Oberart *zugeordnet*). Geben Sie eine weitere Ereignis-Art, Ereignis-Oberart und Schnittstelle an, die genauso zusammengehören.

`mouseClicked`, Maus-Ereignis, `MouseListener`
`mouseMoved`, Mausbewegungs-Ereignis, `MouseMotionListener`
`actionPerformed`, Aktions-Ereignis, `ActionListener`

6.2. Erläutern Sie kurz, zu welchen *Listener-Schnittstelle* es eine *Adapter-Klasse* gibt und zu welchen nicht.

Nur zu `Listener-Schnittstelle`, die mehr als eine Methode enthalten, gibt es eine Adapter-Klasse.

6.3. Geben Sie mit dem Befehl `System.out.print` ein *anonymes Objekt* aus. Den Typ des Objekts dürfen Sie frei wählen.

`System.out.println(new String("ABC"));`

6.4. Welche der folgenden Sätze treffen zu und welche nicht?

6.4.1. Prozesse (eines Betriebssystems) werden immer gleichzeitig und Fäden (engl. threads of control) werden ebenfalls immer gleichzeitig ausgeführt.

6.4.2. Prozesse werden immer gleichzeitig und Fäden werden immer nebenläufig zueinander ausgeführt.

6.4.3. Prozesse werden immer nebenläufig zueinander und Fäden werden immer gleichzeitig ausgeführt.

6.4.4. Prozesse werden immer nebenläufig zueinander und Fäden werden ebenfalls immer nebenläufig zueinander ausgeführt.

6.4.5. Fäden werden manchmal gleichzeitig und manchmal nacheinander ausgeführt.

Die Sätze 6.4.4. und 6.4.5. treffen zu, die übrigen Sätze treffen nicht zu.

6.5. Wie Sie wahrscheinlich wissen, beschreibt die kontextfreie Grammatik der Sprache Java nicht genau die Sprache Java, sondern eine etwas andere Sprache Java' ("Java Strich"). Erläutern Sie kurz, welche Worte zu der Sprache Java' gehören.

Zur Sprache Java' gehören alle korrekten Java-Programme und zusätzliche solche Worte, die ganz ähnlich aussehen wie Java-Programme, aber bestimmte Kontextbedingungen nicht erfüllen.

6.6. Betrachten Sie die folgende Regel einer kontextfreien Grammatik:

`FieldDecl : {"static" | "final" | "public" | "private"} Type Var {"", " Var"} ";"`

Welche der folgenden Symbolfolgen kann man mit dieser Regel aus dem Zwischensymbol (engl. non-terminal symbol) `FieldDecl` ableiten und welche nicht?

- | | |
|--|-----------------|
| 6.6.1. <code>"static" "final" "public" Type Var ";"</code> | ableitbar |
| 6.6.2. <code>"final" "static" "public" Type Var ";"</code> | ableitbar |
| 6.6.3. <code>Type Var</code> | nicht ableitbar |
| 6.6.4. <code>"private" "private" Type Var ";"</code> | ableitbar |
| 6.6.5. <code>"static" Type Var ", " Var ", " Var ", "</code> | nicht ableitbar |