

Vorname (bitte deutlich und lesbar)

Nachname (bitte deutlich und lesbar)

Matrikel-Nr (bitte deutlich und lesbar)

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

Schreiben Sie jede Lösung auf die Vorderseite eines *neuen Blattes* (und lassen Sie die Rückseiten Ihrer Lösungsblätter *leer*). Diese Klausur besteht aus 6 Aufgaben.

Aufgabe 1 (20 Punkte): Schreiben Sie eine Methode entsprechend der folgenden Spezifikation:

```
1  static public int anzAB(ArrayList<String[]> alsr) {
2      // Wie oft kommt "AB" in den String-Komponenten der Reihungskompo-
3      // nenten der Sammlung alsr vor?. Diese Funktion liefert die Antwort.
4      ...
5  } // anzAB
```

Aufgabe 2 (20 Punkte) Schreiben Sie eine Methode entsprechend der folgenden Spezifikation:

```
1  static int[] histoZiffern(String s) {
2      // Liefert eine Reihung der Laenge 10, in der steht, wie oft die
3      // Ziffern '0' bis '9' in s vorkommen. Ist erg die Ergebnisreihung,
4      // so gilt:
5      // erg[0] enthaelt die Anzahl der Zeichen '0' in s
6      // erg[1] enthaelt die Anzahl der Zeichen '1' in s
7      // ...
8      // erg[9] enthaelt die Anzahl der Zeichen '9' in s
9      //
10     // Beispiel:
11     // int[] ergA = histoZiffern("88ab12de211fg88");
12     // ergA ist dann gleich {0, 3, 2, 0, 0, 0, 0, 0, 4, 0}
13     ...
14 } // histoZiffern
```

Wichtige Anforderung: *Ziffern* und *nicht-Ziffern* im Parameterstring *s* dürfen Sie natürlich mit einer *if*-Anweisung unterscheiden. Aber welche Komponente der Ergebnisreihung *erg* um 1 erhöht werden muss, sollten Sie mit Hilfe einer einfachen *Index-Rechnung* festlegen (wie bei der Übung Histogramme), und *nicht* mit einer komplizierten *if*- oder *switch*-Anweisung!

Aufgabe 3 (15 Punkte): Betrachten Sie die folgenden Befehle:

```
1  Object[] obr = null;
2  obr = new Object[3];
3  obr[0] = obr;
4  obr[2] = obr;
```

Stellen Sie die Variable *obr* *dreimal* als Boje dar, und zwar so wie sie in den folgenden Momenten aussieht:

Moment 1: Wenn die **Zeile 1** fertig ausgeführt ist

Moment 2: Wenn die **Zeile 2** fertig ausgeführt ist

Moment 3: Wenn die **Zeile 4** fertig ausgeführt ist

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

```

1  static class K1 { ... }
2  static class K2 extends K1 { ... }
3  static class K3 extends K2 { ... }
4  static class K4 extends K1 { ... }
5
6  static K1 v1 = new K1();
7  static K2 v2 = new K2();
8  static K3 v3 = new K3();
9  static K4 v4 = new K4();
10
11 static K1 v12 = v2;
12 static K1 v13 = v3;
13
14 static K2 v22 = v2;
15 static K2 v23 = v3;
16
17 static K3 v33 = v3;

```

Füllen Sie die folgende Tabelle (mit "JA" und "NEIN") aus:

Befehl	Zur Compile-Zeit		Zur Laufzeit
	Ist die Zuweisung erlaubt?	Ist der Cast erlaubt?	Funktioniert der Cast?
<code>v4 = (K4) v33;</code>			
<code>v3 = (K4) v13;</code>			
<code>v1 = (K2) v13;</code>			
<code>v1 = (K3) v12;</code>			
<code>v2 = (K3) v23;</code>			
<code>v2 = (K1) v22;</code>			

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

```

1  // Schleife 5.1:
2  int n = -2;
3  for (int i=0; i<=5; i+=n) {
4      p(i + " ");
5      n++;
6  }
7  pln();
8
9  // Schleife 5.2:
10 int[] rei = {10, 20, 30};
11 for (int g : rei) g++;
12 for (int g : rei) p(g + " ");
13 pln();
14

```

```
15 // Schleife 5.3:
16 StringBuilder[] sbr = {
17     new StringBuilder("A"),
18     new StringBuilder("B"),
19     new StringBuilder("C"),
20 };
21
22 for (StringBuilder sb : sbr) sb.append("X ");
23 for (StringBuilder sb : sbr) p(sb);
24 pln();
25
26 // Schleife 5.4:
27 for (char c1='A'; c1<='H'; c1+=2) {
28     for (char c2='A'; c2<=c1; c2++) {
29         p(c2);
30     }
31     pln();
32 }
```

Die Namen `p` und `pln` sind auch hier Abkürzungen für die Namen `System.out.print` und `System.out.println`.

Aufgabe 6: (15 Punkte)

6.1. Betrachten Sie die folgende Klassen-Vereinbarung:

```
1 class Carola {
2     private int otto;
3     static public void machWas(double d) {...}
4     public Carola() {...}
5     protected String toString() {...}
6     Carola(int i) {...}
7     private static double emil = 3.5;
8     static String text() {...}
9 }
```

Geben Sie von jedem Element dieser Klasse folgende Informationen an:

- den Namen des Elements
- seine Erreichbarkeit
- seine Aspektzugehörigkeit
- seine Art

Beispiel: `tuNichts`: private Klassen-Methode

6.2. Welche drei *Arten von Befehlen* haben wir unterschieden? Geben Sie zu jeder Art von Befehl ein *Beispiel* an.

6.3. Aus wie vielen und welchen Teilen besteht der Ausdruck `b1 && b2 || b3 && b4`?

6.4. Wo (in einem Java-Programm) darf man eine `continue`-Anweisung verwenden und was bewirkt sie?

6.5. Was ist (beim Programmieren von Grabos in Java) ein *Behälter*?

6.6. Geben Sie eine allgemeine Definition für den Begriff *Oberart von Ereignissen* an und nennen Sie zwei Beispiele für solche *Oberarten von Ereignissen*.

Beurteilung dieser Klausur:

A1	
A2	
A3	
A4	
A5	
A6	
Summe	
Note	
Datum	

Korrigierte Beurteilung:

A1	
A2	
A3	
A4	
A5	
A6	
Summe	
Note	
Datum	

Lösung 1 (20 Punkte): Schreiben Sie eine Methode entsprechend der folgenden Spezifikation:

```

1  static public int anzAB(ArrayList<String[]> alsr) {
2      // Wie oft kommt "AB" in den String-Komponenten der Reihungskompo-
3      // nenten der Sammlung alsr vor?. Diese Funktion liefert die Antwort.
4
5      int anz = 0;
6
7      for (String[] sr : alsr) {
8          for (String s : sr) {
9              for (int i=0; i<s.length()-1; i++) {
10                 if (s.charAt(i) == 'A' && s.charAt(i+1) == 'B') anz++;
11             }
12         }
13     }
14     return anz;
15 } // anzAB

```

Lösung 2 (20 Punkte) Schreiben Sie eine Methode entsprechend den folgenden Spezifikationen:

```

1  static int[] histoZiffern(String s) {
2      // Liefert eine Reihung der Laenge 10, in der steht, wie oft die
3      // Ziffern '0' bis '9' in s vorkommen. Ist erg die Ergebnisreihung,
4      // so gilt:
5      // erg[0] enthaelt die Anzahl der Zeichen '0' in s
6      // erg[1] enthaelt die Anzahl der Zeichen '1' in s
7      // ...
8      // erg[9] enthaelt die Anzahl der Zeichen '9' in s
9      //
10     // Beispiel:
11     // int[] ergA = histoZiffern("88ab12de211fg88");
12     // ergA ist dann gleich {0, 3, 2, 0, 0, 0, 0, 0, 4, 0}
13
14     int[] erg = new int[10];
15
16     for (int i=0; i<s.length(); i++) {
17         char c = s.charAt(i);
18         if ('0' <= c && c <= '9') {
19             erg[c - '0']++;
20         }
21     }
22
23     return erg;
24 } // histoZiffern

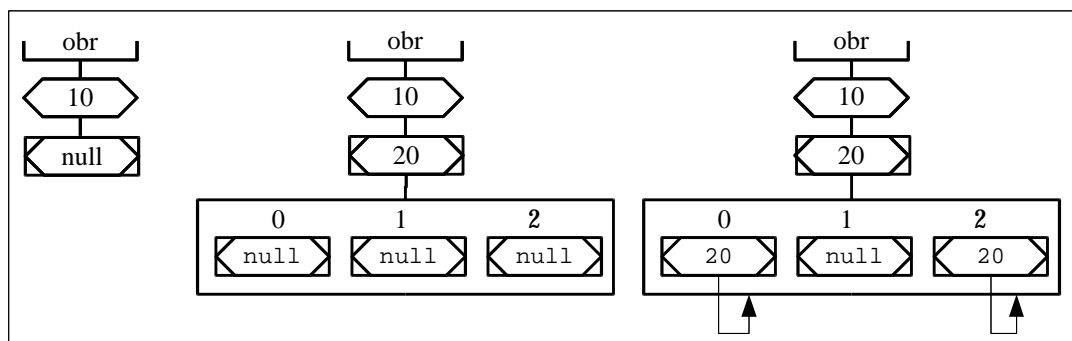
```

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

```

1  Object[] obr = null;
2  obr = new Object[3];
3  obr[0] = obr;
4  obr[2] = obr;

```



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

```

1  static class K1 { ... }
2  static class K2 extends K1 { ... }
3  static class K3 extends K2 { ... }
4  static class K4 extends K1 { ... }
5
6  static K1 v1 = new K1();
7  static K2 v2 = new K2();
8  static K3 v3 = new K3();
9  static K4 v4 = new K4();
10
11 static K1 v12 = v2;
12 static K1 v13 = v3;
13
14 static K2 v22 = v2;
15 static K2 v23 = v3;
16
17 static K3 v33 = v3;

```

Füllen Sie die folgende Tabelle (mit "JA" und "NEIN") aus:

	Zur Compile-Zeit		Zur Laufzeit
Befehl	Ist die Zuweisung erlaubt?	Ist der Cast erlaubt?	Funktioniert der Cast?
v4 = (K4) v33;	Ja	Nein	--
v3 = (K4) v13;	Nein	Ja	--
v1 = (K2) v13;	Ja	Ja	Ja
v1 = (K3) v12;	Ja	Ja	Nein
v2 = (K3) v23;	Ja	Ja	Ja
v2 = (K1) v22;	Nein	Ja	--

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

```
1  // Schleife 5.1:
2  int n = -2;
3  for (int i=0; i<=5; i+=n) {
4      p(i + " ");
5      n++;
6  }
7  pln();
8
9  // Schleife 5.2:
10 int[] rei = {10, 20, 30};
11 for (int g : rei) g++;
12 for (int g : rei) p(g + " ");
13 pln();
14
15 // Schleife 5.3:
16 StringBuilder[] sbr = {
17     new StringBuilder("A"),
18     new StringBuilder("B"),
19     new StringBuilder("C"),
20 };
21
22 for (StringBuilder sb : sbr) sb.append("X ");
23 for (StringBuilder sb : sbr) p(sb);
24 pln();
25
26 // Schleife 5.4:
27 for (char c1='A'; c1<='H'; c1+=2) {
28     for (char c2='A'; c2<=c1; c2++) {
29         p(c2);
30     }
31     pln();
32 }
```

Die Ausgaben der Schleifen:

```
// Schleife 5.1:
0 -1 -1 0 2 5
```

```
// Schleife 5.2:
10 20 30
```

```
// Schleife 5.3:
AX BX CX
```

```
// Schleife 5.4:
A
ABC
ABCDE
ABCDEFG
```

Lösung 6: (15 Punkte)**6.1. Betrachten Sie die folgende Klassen-Vereinbarung:**

```
1 class Carola {  
2     private int otto;  
3     static public void machWas(double d) {...}  
4     public Carola() {...}  
5     protected String toString() {...}  
6     Carola(int i) {...}  
7     private static double emil = 3.5;  
8     static String text() {...}  
9 }
```

Geben Sie von jedem Element dieser Klasse folgende Informationen an:

- den Namen des Elements
- seine Erreichbarkeit
- seine Aspektzugehörigkeit
- seine Art

otto : privates Objekt-Attribut
machWas : öffentliche Klassen-Methode
toString : geschützte Objekt-Methode
emil : privates Klassen-Attribut
text : paketweit erreichbare Klassen-Methode

6.2. Welche drei *Arten von Befehlen* haben wir unterschieden? Geben Sie zu jeder Art von Befehl ein *Beispiel* an.

Vereinbarung : **int otto = 17;**
Ausdruck : **otto + 1**
Anweisung : **otto = otto + 1;**

6.3. Was kann der Programmierer mit Hilfe einer *Prozedur* bewirken? Oder: Was bewirkt jeder Aufruf einer Prozedur (ganz allgemein und abstrakt gesagt)?

Dass die Inhalte bestimmter Wertebehälter verändert werden.

6.4. Was kann der Programmierer mit Hilfe einer *Funktion* bewirken? Oder: Was bewirkt jeder Aufruf einer Funktion (ganz allgemein und abstrakt gesagt)?

Dass ein Wert berechnet (und als Ergebnis geliefert) wird.

6.5. Aus wie vielen und welchen Teilen besteht der Ausdruck `b1 && b2 || b3 && b4`?

Aus 3 Teilen: Teil1: `b1 && b2` Teil2: `||` Teil3: `b3 && b4`

6.6. Wo (in einem Java-Programm) darf man eine `continue`-Anweisung verwenden und was bewirkt sie?

Eine `continue`-Anweisung darf man (nur) in Schleifen verwenden. Sie beendet die aktuelle Rumpfausführung (nicht die Ausführung der Schleife).