

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 int anzBloecke(char[][] rrc) {
2      // Sie duerfen sich darauf verlassen, dass rrc "rechteckig" ist,
3      // d.h. dass alle Komponenten von rrc Reihungen gleicher Laenge sind.
4      // Wie oft kommt in rrc ein "Block" vor, der aus den folgenden 4 Zeichen
5      // (auf zwei untereinanderliegenden Zeilen) besteht:
6      //
7      // ...ab...
8      // ...cd...
9      //
10     // Diese Funktion liefert die Antwort. Beispiele:
11     //
12     // char[][] rrcA = { {'x','x','x','a','b'},
13     //                   {'a','b','x','c','d'},
14     //                   {'c','d','a','b','x'},
15     //                   {'x','x','c','d','x'},
16     //                   };
17     //
18     // char[][] rrcB = { {'1','2','3','a','b','c','d'},
19     //                   {'4','5','c','d','c','d','6'},
20     //                   };
21     //
22     // char[][] rrcC = { {}, {}, {}, };
23     //
24     // anzBloecke(rrcA) ist gleich 3
25     // anzBloecke(rrcB) ist gleich 0
26     // anzBloecke(rrcC) ist gleich 0
27     // ...
28 } // anzBloecke

```

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

```

1  static public boolean istKleiner(Long[] rl, ArrayList<Long> ali) {
2      // Liefert true genau dann wenn
3      // rl weniger Komponenten enthaelt als ali oder wenn
4      // rl gleich viele Komponenten enthaelt wie ali und jede Komponente
5      // von rl kleiner ist als die entsprechende Komponente von ali.
6      // Beispiele:
7      //
8      // Long[] rl1 = {new Long(10), new Long(20)};
9      // Long[] rl2 = {new Long(20), new Long(30)};
10     // Long[] rl3 = {new Long(10), new Long(20), new Long(30)};
11     // ArrayList<Long> ali1 = new ArrayList<Long>(Arrays.asList(rl1));
12     // ArrayList<Long> ali2 = new ArrayList<Long>(Arrays.asList(rl2));
13     // ArrayList<Long> ali3 = new ArrayList<Long>(Arrays.asList(rl3));
14     //
15     // istKleiner(rl1, ali2) ist gleich true
16     // istKleiner(rl1, ali3) ist gleich true
17     // istKleiner(rl2, ali1) ist gleich false
18     // istKleiner(rl3, ali3) ist gleich false
19     // ...
20 } // istKleiner

```

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

```
1      StringBuilder sb1 = new StringBuilder("AB");
2      StringBuilder sb2 = new StringBuilder("CD");
3      StringBuilder sb3 = sb2;
4
5      sb2.append("EF");
6      sb3.append("GH");
7      sb1 = sb2;
```

Stellen Sie die Variablen sb1, sb2, sb3 *zweimal* als Bojen dar, und zwar so, wie sie in folgenden Momenten aussehen:

3.1. Nachdem der Ausführer die Zeile 3 fertig ausgeführt hat.

3.2. Nachdem der Ausführer die Zeile 7 fertig ausgeführt hat.

**Aufgabe 4 (15 Punkte):** Angenommen, Sie haben (in einem Java-Programm) eine Variable namens mr vereinbart, die auf ein JFrame-Objekt zeigt, etwa so:

```
17      JFrame mr = new JFrame("Mein Rahmen!");
```

Vereinbaren Sie jetzt ein MouseListener-Objekt namens ml und melden Sie es bei mr an.

Die Methode mousePressed in diesem Objekt ml soll bewirken, dass ein String der Form "(xxx, yyy)" zur Standardausgabe ausgegeben wird. Dabei sollen xxx und yyy die Koordinaten des Punktes sein, an dem der Benutzer auf einen Mausknopf gedrückt hat.

Die übrigen Methoden im Objekt ml (d.h. alle ausser mousePressed) sollen nichts machen.

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

```
1      // Schleife 5.1: -----
2      int n = -5;
3      for (int i=0; i<=10; i+=n) {
4          n +=2;
5          p(i + " ");
6      }
7      pln();
8
9      // Schleife 5.2: -----
10     int[] rei = {1, 0, 3, 2};
11     for (int i : rei) rei[i] = 4-i;
12     pln(Arrays.toString(rei));
13
14     // Schleife 5.3: -----
15     StringBuilder[] sbr = {
16         new StringBuilder("A+"),
17         new StringBuilder("B+"),
18         new StringBuilder("C+"),
19     };
20
21     for (int i=1; i<sbr.length; i++) {
22         sbr[i].insert(0, sbr[i-1]);
23     }
24
25     for (StringBuilder sb : sbr) p(sb);
26     pln();
27
28     // Schleife 5.4: -----
29     for (char c1='H'; c1>='A'; c1-=2) {
30         for (char c2=c1; c2>='A'; c2--) p(c2);
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. Warum hat man in fast alle Programmiersprachen das Konzept eines *Typs* eingebaut?

6.2. Übersetzen Sie jeden der folgenden beiden Java-Befehle ins Deutsche:

```
1    ... 2 * (x + y) / 17 ...  
2    x = 2 * (x + y) / 17;
```

6.3. Mit welchen zusammengesetzten Anweisungen kann der Programmierer bewirken, dass die darin enthaltenen Anweisungen *weniger als einmal* ausgeführt werden?

6.4. In der folgenden Klasse `Carola` werden 4 Elemente vereinbart.

Geben Sie von jedem der Elemente folgende Informationen an:

seinen *Namen*, seine *Erreichbarkeit*, seine *Aspektzugehörigkeit* und seine *Art*

(z.B. "fritz ist ein öffentliches Klassen-Attribut" oder "corinna ist eine private Objekt-Methode" oder so ähnlich).

```
3 class Carola {  
4     protected static int otto;  
5     void machWas() {System.out.println("Hallo");};  
6     class Carl {};  
7     static public interface Sonja {void tuNix();};  
8 }
```

6.5. Wie heißen die obersten *drei Ausnahmeklassen* (eine Klasse und zwei Unterklassen)?

6.6. Geben Sie Vereinbarungen an die bewirken, dass der Name `karlHeinz` *überladen* wird.

**Anmerkung:** *Überladen* (engl. to overload) ist viel einfacher als *überschreiben* (engl. to override) und man sollte die beiden Begriffe möglichst selten miteinander verwechseln.

**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 int anzBloেকে(char[][] rrc) {
2      // Sie dürfen sich darauf verlassen, dass rrc "rechteckig" ist,
3      // d.h. dass alle Komponenten von rrc Reihungen gleicher Länge sind.
4      // Wie oft kommt in rrc ein Block mit folgenden char-Werten vor:
5      //
6      // ...ab...
7      // ...cd...
8      //
9      // Diese Funktion liefert die Antwort. Beispiele:
10
11     int anz = 0;
12
13     for (int zeil=0; zeil<rrc.length-1; zeil++) {
14         for (int spa=0; spa<rrc[zeil].length-1; spa++) {
15             if (rrc[zeil+0][spa+0] == 'a' &&
16                 rrc[zeil+0][spa+1] == 'b' &&
17                 rrc[zeil+1][spa+0] == 'c' &&
18                 rrc[zeil+1][spa+1] == 'd')
19                 {
20                     anz++;
21                 }
22         }
23     }
24     return anz;
25 }
26 // anzBloেকে
```

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

```
1  static public boolean istKleiner(Long[] rl, ArrayList<Long> ali) {
2      // Liefert true genau dann wenn
3      // rl weniger Komponenten enthält als ali oder wenn
4      // rl gleich viele Komponenten enthält wie ali und jede Komponente
5      // von rl kleiner ist als die entsprechende Komponente von ali.
6
7      if (rl.length < ali.size()) return true;
8      if (rl.length > ali.size()) return false;
9
10     for (int i=0; i<rl.length; i++) {
11         if (rl[i] >= ali.get(i)) return false;
12     }
13     return true;
14
15 } // istKleiner
```

**Lösung 3 (15 Punkte):** Betrachten Sie die folgenden Befehle:

```

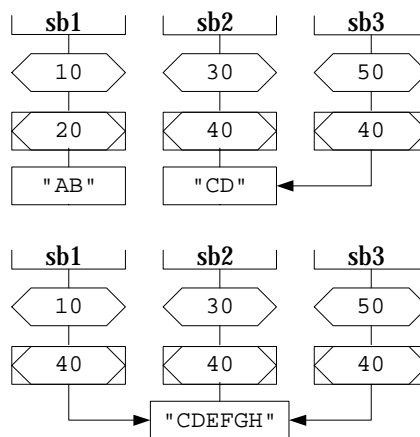
1      StringBuilder sb1 = new StringBuilder("AB");
2      StringBuilder sb2 = new StringBuilder("CD");
3      StringBuilder sb3 = sb2;
4
5      sb2.append("EF");
6      sb3.append("GH");
7      sb1 = sb2;

```

Stellen Sie die Variablen sb1, sb2, sb3 *zweimal* als Bojen dar, und zwar in folgenden Momenten:

3.1. Nachdem der Ausführer die Zeile 3 fertig ausgeführt hat.

3.2. Nachdem der Ausführer die Zeile 7 fertig ausgeführt hat.

**Lösung 4 (15 Punkte):** Angenommen, Sie haben (in einem Java-Programm) eine Variable namens mr vereinbart, die auf ein JFrame-Objekt zeigt, etwa so:

```

17      JFrame mr = new JFrame("Mein Rahmen!");

```

Vereinbaren Sie jetzt ein MouseListener-Objekt namens ml und melden Sie es bei mr an.

Die Methode mousePressed in diesem Objekt ml soll bewirken, dass ein String der Form "(xxx, yyy)" zur Standardausgabe ausgegeben wird. Dabei sollen xxx und yyy die Koordinaten des Punktes sein, an dem der Benutzer auf einen Mausknopf gedrückt hat.

Die übrigen Methoden im Objekt ml (d.h. alle ausser mousePressed) sollen nichts machen.

```

18      MouseListener ml = new MouseAdapter() {
19          public void mousePressed(MouseEvent me) {
20              println("(" + me.getX() + ", " + me.getY() + ")");
21          }
22      };
23
24      mr.addMouseListener(ml);

```

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

```

**Die Ausgaben der Schleifen:**

```

// Schleife 5.1: 0 -3 -4 -3 0 5
// Schleife 5.2: [1, 3, 3, 1]
// Schleife 5.3: A+A+B+A+B+C+
// Schleife 5.4:
HGFEDCBA
FEDCBA
DCBA
BA

```

**Lösung 6: (15 Punkte)**

6.1. Warum hat man in fast alle Programmiersprachen das Konzept eines Typs eingebaut?

Um bestimmter Flüchtigkeitsfehler des Programmierers automatisch (d.h. billiger) auffinden zu können.

6.2. Übersetzen Sie jeden der folgenden beiden Java-Befehle ins Deutsche:

```
1    ... 2 * (x + y) / 17 ...
2    x = 2 * (x + y) / 17;
```

1 Berechne den Wert des Ausdrucks  $2 * (x + y) / 17$

2 Berechne den Wert des Ausdrucks  $2 * (x + y) / 17$  und tue ihn (den Wert) in die Variable x.

6.3. Mit welchen zusammengesetzten Anweisungen kann der Programmierer bewirken, dass die darin enthaltenen Anweisungen *weniger als einmal* ausgeführt werden?

Mit der if-Anweisung und der switch-Anweisung.

6.4. In der folgenden Klasse Carola werden 4 Elemente vereinbart.

Geben Sie von jedem der Elemente

seinen *Namen*, seine *Erreichbarkeit*, seine *Aspektzugehörigkeit* und seine *Art* an

(z.B. "fritz ist ein öffentliches Klassen-Attribut" oder "corinna ist eine private Objekt-Methode" oder so ähnlich).

```
3 class Carola {
4     protected static int otto;
5     void machWas() {System.out.println("Hallo");};
6     class Carl {};
7     static public interface Sonja {void tuNix();};
8 }
```

otto ist ein geschütztes Klassen-Attribut,

machWas ist eine paketweit erreichbare Objekt-Methode.

Carl ist eine paketweit erreichbare Objekt-Klasse (oder: nicht-statische Klasse).

Sonja ist ein öffentliche Klassen-Schnittstelle (oder: statische Schnittstelle).

6.5. Wie heißen die obersten drei Ausnahmeklassen (eine Klasse und zwei Unterklassen)?

Throwable, Exception, Error

6.6. Geben Sie Vereinbarungen an die bewirken, dass der Name karlHeinz *überladen* wird.

Anmerkung: *Überladen* (engl. to overload) ist viel einfacher als *überschreiben* (engl. to override) und man sollte die beiden Begriffe möglichst selten miteinander verwechseln.

```
9     void karlHeinz(int n) {p1n(n);}
10    int  karlHeinz(String s) {return s.length();}
```