Klausur: WS2009/10 (I), Kurs: MD1, PR1 Konzepte (Prof. Dr. Schimkat), Datum: Di., 03.02.10

Vorname: Nachname: Matrikelnummer:

Dies ist mein letzter Prüfungsversuch O ja O nein

Aufgabe	1	2	3	4	5	6	7	Gesamt	LV-Note
Punkte max:	5	20	20	12	13	20	10	100	XXXXXX
Punkte	5	18	20	12	13	12	10	90	1,0

Hinweise:

1) Beschriften Sie alle verwendeten Blätter nur einseitig -> die Rückseite ist freizulassen.
2) Falls Sie Teile Ihrer Lösung auf Extrablätter schreiben:
   Verwenden Sie für jede Aufgabe ein eigenes Extrablatt.
3) Auf zusätzlichen Blättern lassen Sie links, rechts, oben und unten mindestens je 2 cm Rand und schreiben Sie in die obere rechte Ecke Ihren Namen und Ihre Matrikelnummer.
4) Es sind die in der Lehrveranstaltung eingeführten Fachbegriffe zu verwenden.
5) Ausschweifende oder unnötig komplizierte Erklärungen führen zu Punktabzug.
6) Die Lösungen sind programmiertechnisch sinnvoll, verständlich und den jeweiligen Konventionen entsprechend auszuführen. Unnötig komplizierte oder unsinnige Programmkonstrukte führen zu Punktabzug.
7) Kommunikation jeder Art ist nur mit dem Dozenten gestattet.

Aufgabe 1 (5 Punkte): Halten Sie die oben genannten Formalien (Hinweis 1 bis 3) ein. 5

Aufgabe 2 (20 Punkte):
Geben Sie jeweils eine vollständige Deklaration für die folgenden Methoden an:

a) [5 Punkte]

```
/**
* Der Rückgabewert gibt an, ob drei aufeinanderfolgende Zahlen a, b, c der Größe nach
* ansteigend angeordnet sind. Bsp.:
* b = isOrderedAscending(-1, 3, 8); // b wird true zugewiesen
* m = isOrderedAscending( 1, 1, 2); // m wird false zugewiesen
* k = isOrderedAscending(-1, 2, 0); // k wird false zugewiesen
*/
public static boolean isOrderedAscending(int a, int b, int c)
```

b) [7 Punkte]

```
/**
* Gibt die Anzahl der Indices zurück, für die Array a und Array b den gleichen Wert
* enthalten. Bsp.:
* equalCount (new int[] {1, 2, 3, 4}, new int[] {1, 1, 1, 4}) gibt 2 zurück
* equalCount (new int[] {1, 2, 3, 4}, new int[] {0, 1, 2, 3}) gibt 0 zurück
Beachte: Die Methode soll für alle möglichen Parameterwerte korrekt funktionieren und
keine Exceptions werfen!
 */
public static int equalCount (int [] a, int [] b)
```

c) [8 Punkte]

```
/** Bestimmt den Mittelwert aller Zahlen in einer Textdatei, in der ganze Zahlen,
Gleitkommazahlen und Texte in beliebiger Reihenfolge und jeweils durch Leerraum
getrennt stehen. Ist die Datei nicht vorhanden oder nicht lesbar oder hat die Datei das
falsche Format, wird der Wert 0 zurückgegeben. Beachte: Die Methode soll für alle
möglichen Parameterwerte korrekt funktionieren und keine Exceptions werfen!
 */
public static double meanValueOf(File path)
```

Aufgabe 3 (20 Punkte):

Gegeben sei:

```
public abstract class BasicGewaechshaus {
      protected int t;  // Temperatur in °Celsius;
      protected int lf; // relative LuftFeuchtigkeit in %;

      public BasicGewaechshaus (int t, int lf) {
            this.t = t;
            this.lf = lf;
      }
      protected String getAirConditions() {
            return lf + "% @ " + t + "°C";
      }
      protected abstract String sizeToString();
      protected abstract String getPlantType();
      public String toString () {
            return "Gewaechshaus fuer " + getPlantType()
                  + " [" + sizeToString() + ", "  + getAirConditions()+ "]";
      }
}

public class Dimension {
      public int width;
      public int length;
      public Dimension (int width, int length) {
            this.width  = width;
            this.length = length;
      }
}
```

Geben Sie die Deklaration einer Klasse `Gewaechshaus` an, die von `BasicGewaechshaus` erbt und die nicht abstrakt ist. Die Klasse `Gewaechshaus` besitzt zwei Instanzvariablen vom Typ `String` bzw. `Dimension`, in der die `pflanzenart` bzw. die `groesse` gespeichert werden.

Die Klasse `Gewaechshaus` soll einen vollständigen Konstruktor, einen Kopierkonstruktor und einen weiteren Konstruktor mit der Parameterliste `(int t, int lf, String plantType, int breite, int laenge)` besitzen. Die Ausgabe eines `Gewaechshaus`objekts in der Art `System.out.println (gewaechshaus);` soll eine Ausgabe nach dem Muster

`grosses Gewaechshaus fuer Tomaten [20 X 100 qm, 33% @ 24°C]` erzeugen. Hat das Gewaechshaus eine Groesse (=laenge*breite) von mehr als 10 Quadratmetern, dann gilt es als grosses Gewaechshaus, ansonsten als kleines Gewaechshaus, dann soll die Ausgabe nach dem Muster

`kleines Gewaechshaus fuer Moehren [1 X 2 qm, 41% @ 22°C]` erzeugt werden.

Vermeiden Sie unnötige Codevervielfachungen.

// Anwendungsbeispiel: *Die Anweisungen*

```
Dimension flaecheMoehrenbeet = new Dimension(1, 2);
BasicGewaechshaus tomatenPlantage
                  = new Gewaechshaus(24, 33, "Tomaten", new Dimension(20, 100));
BasicGewaechshaus moehrenGarten
                  = new Gewaechshaus(22, 41, "Moehren", flaecheMoehrenbeet);
System.out.println(tomatenPlantage);
System.out.println(moehrenGarten);
```

// erzeugen die Ausgabe:

```
grosses Gewaechshaus fuer Tomaten [20 X 100 qm, 33% @ 24°C]
kleines Gewaechshaus fuer Moehren [1 X 2 qm, 41% @ 22°C]
```

Aufgabe 4 (12 Punkte):

Vereinfachen Sie die Codefragmente a) bis c) so weit wie möglich, aber ohne die Funktionalität zu verändern:

a) [3]
```
if (markerflag == false) {
      markerflag = true;
} else {
      markerflag = false;
}
```

b) [5]
```
// Beachte: term kann auch einen negativen Wert haben!
public static int newCount (int count, int term) {
      for (int i = 0; i < term; i++) {
            count = count + 1;
      }
      return count;
}
```

c) [4]
```
private void runSubMenu (char zeichen) {
      switch (zeichen)  {
      case 'a':   runSubMenuA ();
                  break;
      case 'A':   runSubMenuA ();
                  break;
      case 'b':   runSubMenuB ();
                  break;
      case 'B':   runSubMenuB ();
                  break;
      }
}
```

13

Aufgabe 5 (13 Punkte):

a) [4] Nennen Sie die vier grundlegenden Klassen des IO-Systems von Java:

java.io. InputStream java.io. Writer

java.io. OutputStream java.io. Reader

4

b) [3] Ordnen Sie den folgenden mit printf() verwendbaren Umwandlungszeichen die richtige Bedeutung (durch Ankreuzen) zu:

d	O depressed	O double	X decimal	O direct
f	X fixed	O formal	O fine	O float
s	O sign	O subclass	X string	O supertype

3

c) [4] Geben Sie für die folgenden Methoden der Klasse Object jeweils den vollständigen Methodenkopf an:

public boolean equals (Object o);

public int hashCode ()

4

d) [2] Wofür steht die Abkürzung DRY?

2

D on't R epeat Y ourself

Aufgabe 6 (20 Punkte):

12

Geben Sie ohne Begründung an, welche der folgenden Aussagen zutreffen und welche nicht zutreffen:

	trifft zu	trifft nicht zu
Eine abstrakte Klasse kann nicht mit new instanziiert werden	X	O
Eine abstrakte Klasse hat immer mindestens eine abstrakte Methode	O	X
Exceptions sind Interfaces	O	X
Arrays sind Referenztypen	X	O
Ein Objekt kann mehrere Typen haben	X	O
Ein Package kann mehrere Interfaces implementieren	X	O
Eine lokale Variable darf nicht protected sein	X	O
Die Klasse aller Objekte ist zur Compilezeit bekannt	X	O
Genaugenommen kann keine Variable ein Objekt speichern	X	O
Der Standardkonstruktor hat den Rückgabetyp void	O	X

f

f

Aufgabe 7 (10 Punkte):

Betrachten Sie das folgende Java-Programm:

```
public class Exa extends Exception {
}
public class Exb extends Exa {
}
public class Exc extends Exb {
}
//-----------------------------------------
public class Aufgabe07 {
      public static void main(String[] sonja) {
            for (int n = 1; n <= 3; n++) {
                  try {
                        met01(n);
                  } catch (Exa exa) {
                        System.out.println("R ex:A");
                  } finally {
                        System.out.println("S ----");
                  }
            }
      }
      // ----------------------------------
      private static void met01(int n) throws Exa {
            try {
                  met02(n);
                  System.out.println("T ----");
            } catch (Exc exc) {
                  System.out.println("U ex:C");
            } catch (Exb exb) {
                  System.out.println("V ex:B");
                  throw exb;
            }
      }
      // ----------------------------------
      protected static void met02(int n) throws Exa {
            try {
                  switch (n) {
                  case 1:
                        throw new Exa();
                  case 2:
                        throw new Exb();
                  case 3:
                        throw new Exc();
                  }
            } catch (Exc exc) {
                  System.out.println("W ex:C");
                  throw exc;
            }
      }
}
//-----------------------------------------
```

Geben Sie an, was dieses Programm zur Standardausgabe (d.h. zum Bildschirm) ausgibt.

R ex: A
S ----
V ex: B
R ex: A
S ----
W ex: C
U ex: C
S ----

|| 1. Schleifendurchlauf
|| 2. Schleifendurchlauf
|| 3. Schleifendurchlauf

10

2.) a,

```
public static boolean isOrderedAscending(int a,
                    int b, int c) {
        return a < b && b < c;
}
```

5

b,

```
public static int equalCount (int []a, int []b){
        if( a == null || b == null ){
                return 0;
        }
        int laenge= a.length;
        if( laenge > < b.length){
                laenge = b.length;
        }
        int count = 0;
        for( int i=0; i < laenge; i++) {
                if( a[i] == b[i] ){
                        count ++;
                }
        }
}
```

7

2.) c)

```
public static double meanValueOf( File path){
    if( path == null){
        return 0;
    }
    Scanner fin = null;
    int sume = 0;
    int cont = 0;
    try {
        fin = new Scanner (new BufferedReader(
                    new FileReader( path) ));
        while( fin.hasNext() ){
            if(fin.hasNextInt() || fin.hasNextDouble()){
                sume += fin.next();
            } else {
                cont++;
                fin.next();
            }
        } catch(IOException e){
            return 0;
        }
        finally{
            try{
                fin.close();
            } catch (IOException e){
                // return 0;
            }
        }
}
```

isDirectory()

Double()
(Try/fehler)

Zu 2.) c)

```
if( count == 0) {
    return 0;
}
else{ sum
    return sum / count;
}
```

+1

6

3.)

```
public class Gewaechshaus extends BasicGewaechshaus{

    ~~pcd Stri~~
    protected String pflanzenart;
    protected Dimension groesse;
    public Gewaechshaus (int t, int lf, String pa,
                         Dimension gr) {
        super (t, lf);
        this.pflanzenart = new String(pa);
        this. groesse = new Dimension (gr.width,
                                       gr. length);
    }
    public Gewaechshaus( Gewaechshaus other) {
        this(other. t, other. lf, other. pflanzenart,
             other. groesse);
    }
    public Gewaechshaus(int t, int lf, String plantType,
                        int breite, int laenge) {
        this (t, lf, new String( plantType), new
              Dimension( breite, laenge));
    }
    public String toString () {
        if((groesse * breite) > 10) {
            return "~~grote~~ grosses " + super.toString();
        }
        return "kleines " + super. toString();
    }
```

Zu 3.)

```
protected String sizeToString() {
    return groesse.width + " X " +
           groesse.length + " qm";
}
protected String getPflanzType() {
    return pflanzenart;
}
}
```

✓

20

4.) a)

```
markerflag = !markerflag;
```

3

b)

```
public static int newCount(int count, int term){
    if( term < 0 ) {
        return count;
    }
    return count + term;
}
```

5

c)

```
private void runSubMenu( char zeichen ) {
    switch ( Character.toLowerCase(
                zeichen ) {
        case 'a': runSubmenuA();
                  break;
        case 'b': runSubMenuB();
                  // (break;) // break nicht
    }                          // notwendig
}
```

✓

4

/12