Sigrid Weil

Klausur SS 2024
- Musterlösung -

Aufgabe 1 (10 P)

Gegeben sei der folgende Java-Code:

```
1  public static int fibonacci(int n) {
2
3      int fnminus2 = 1;
4      int fnminus1 = 1;
5      int fn =1;
6
7      for(int i = 3; i <= n; i++) {
8          fn = fnminus1+fnminus2;
9          fnminus2 = fnminus1;
10         fnminus1 = fn;
11     }
12     return fn;
13 }
```

Lösung 1

a) Tracetabelle: (5 P)

pro Fehler -1 P

Zeile	n	fn	fnminus2	fnminus1	i	i ≤ n
1	4					
3			1			
4				1		
5		1				
7					3	true
8		2				
9			1			
10				2		
7					4	true
8		3				
9			2			
10				3		
7					5	false

Returnwert: 3

b) Äquivalente do-while-Schleife: (5 P)

pro Fehler -1 P

```
public static int dowhilefibonacci(int n){
    int fnminus2 = 1;
    int fnminus1 = 1;
    int fn =1;
    int i=3;
    do{
        if(i>n) break;
        fn = fnminus1+fnminus2;
        fnminus2 = fnminus1;
        fnminus1 = fn;
        i++;
    }
    while (i<=n);
    return fn;
}
```

Aufgabe 2 (20 P)
Wir wollen die Personalabteilung eines Unternehmens mit neuer Software unterstützen:

- Die Personalabteilung verwaltet zu jedem Mitarbeitenden den Namen und das Geburtsjahr. Mitarbeiter sind also spezielle Personen.
  Den Code der Klasse `Person` finden Sie im Anhang.
- Jeder Mitarbeiter bekommt eine (fortlaufend automatisch generierte) Personalnummer `persNr`, die durch eine get-Methode erfragt werden kann.
- Die `toString`-Methode liefert einen String, der den Namen und die Personalnummer enthält.
- Außerdem gibt es für jeden Mitarbeiter eine Methode `gehalt()`, die das Gehalt (als `double`-Wert) liefert.

Bei der Gehaltsberechung wird allerdings zwischen Tarifangestellten und Leitenden Angestellten unterschieden:

Für Tarifangestellte („TarifMA“) gilt:

- Tarifangestellte erhalten eine Gehaltsstufe (das ist ein ganzzahliger Wert zwischen 1 und 12), die bei der Erzeugung eine Objektes angegeben wird.
- Es gibt ein monatliches Grundgehalt, das für alle gleich ist (1250.00 Euro).
- Zum Grundgehalt werden monatlich pro Gehaltsstufe 300.00 Euro aufgeschlagen.
- Grundgehalt und Stufenaufschlag können sich ändern (zB durch Tarifverhandlungen), die Veränderungen wirken sich dann natürlich auf alle Mitarbeiter des Unternehmens aus.
- Manche Mitarbeiter erhalten zusätzlich noch eine individuell vereinbarte Provision. Bei Einstellung eines neuen Mitarbeiters ist die Provision immer = 0.
- Definieren Sie getter- und setter für die Größen Grundgehalt, Stufenaufschlag, Provision.
- Definieren Sie eine get-Methode für die Gehaltsstufe.
- Definieren Sie eine Methode `aufsteigen()`, durch die sich die Gehaltsstufe eines Mitarbeiters um 1 erhöht (falls er nicht bereits in der höchsten Gehaltsstufe 12 ist).

Für Leitende Angestellte („Manager“) gilt:

- Leitende Angestellte sind keiner Gehaltsstufe zugeordnet.
- Leitende Angestellte werden auch nicht nach Tarif bezahlt, sondern erhalten ein Gehalt, das bei der Einstellung (Erzeugung eines Objektes) frei vereinbart wird.
- Auch das Gehalt von Managern kann sich ändern.

Erstellen Sie Java-Klassen, die die oben genannten Forderungen erfüllen.
Nutzen Sie die Konzepte von Vererbung und Abstraktion!

(Sie können davon ausgehen, dass alle Methoden nur mit zulässigen Parameterwerten aufgerufen werden. Eine Fehlerbehandlung ist hier nicht nötig.)

Lösung 2

```
public abstract class Mitarbeiter extends Person {

        private static int anz = 0;

        private int persNr;

        public Mitarbeiter(String name, int gebJahr) {
                super(name, gebJahr);
                persNr = anz++;
        }

        public abstract double gehalt();

        public String toString() {
                return name() + " " + persNr;
        }
}


public class TarifMA extends Mitarbeiter {

        // Klassenattribute
        private static double grundgehalt = 1250.0;
        private static double stufenaufschlag = 300.0;

        // Instanzattribute
        private int stufe;
        private double provision;

        // Konstruktor
        public TarifMA(String name, int gebJahr, int stufe) {
                super(name, gebJahr);
                this.stufe = stufe;
                provision = 0.0;
        }

        // Klassenmethoden
        public static void setGrundgehalt(double neu) {
                grundgehalt = neu;
        }

        public static void setStufenaufschlag(double neu) {
                stufenaufschlag = neu;
        }

        public static double getGrundgehalt() {
                return grundgehalt;
        }

        public static double getStufenaufschlag() {
                return stufenaufschlag;
        }

        // Instanzmethoden
        public void setProvision(double p) {
                provision = p;
        }
```

```
        public void aufsteigen() {
                if (stufe < 12)
                        stufe++;
        }

        public double gehalt() {
                return grundgehalt + stufe*stufenaufschlag + provision;
        }
}


public class Manager extends Mitarbeiter {

        private double gehalt;

        public Manager(String name, int gebJahr, double gehalt) {
                super(name, gebJahr);
                this.gehalt = gehalt;
        }

        @Override
        public double gehalt() {
                return gehalt;
        }

}
```

Aufgabe 3 (18 P)

In einem Computerspiel können pro Spiel eine beliebige Anzahl (int-Wert) an Punkten erreicht werden.
Im Highscore werden die jeweils 10 besten Ergebnisse in absteigend sortierter Reihenfolge gespeichert.

a) Definieren Sie eine Klasse `Spiel` mit einem statischen Attribut `int[] highscore`

b) Definieren Sie eine (statische) Methode `eintragen()`. Diese Methode erhält einen int-Wert (ein Spielergebnis) als Eingabe. Falls das Ergebnis zu den bisher erreichten TopTen gehört, wird es in den Highscore an die richtige Position eingetragen. Alle schlechteren Ergebnisse rutschen um eine Position nach hinten bzw. fallen ganz aus dem Highsore heraus. (Möglicherweise gleiche Ergebnisse werden auch mehrfach im Highscore aufgelistet.)

c) Eine Methode `ausgabe` gibt den aktuellen Highscore aus. Falls noch nicht mindestens 10 Spiele gespielt wurden, werden nur die bis dahin erreichten Ergebnisse ausgegeben.

Lösung 3

```
public class Spiel {

        private static int[] highscore = new int[11];
        private static int anzSpiele = 0;

        public static void eintragen(int erg) {
                highscore[anzSpiele] = erg;
                int i = anzSpiele;
                while (i > 0 && highscore[i-1] < erg) {
                        highscore[i] = highscore[i-1];
                        highscore[i-1] = erg;
                        i--;

                }

                if (anzSpiele < 10) {
                        anzSpiele++;
                }

        }

        public static void ausgabe() {
                for (int i = 0; i < anzSpiele; i++)
                        System.out.print( highscore[i] + "\t");
                System.out.println();
        }
}
```

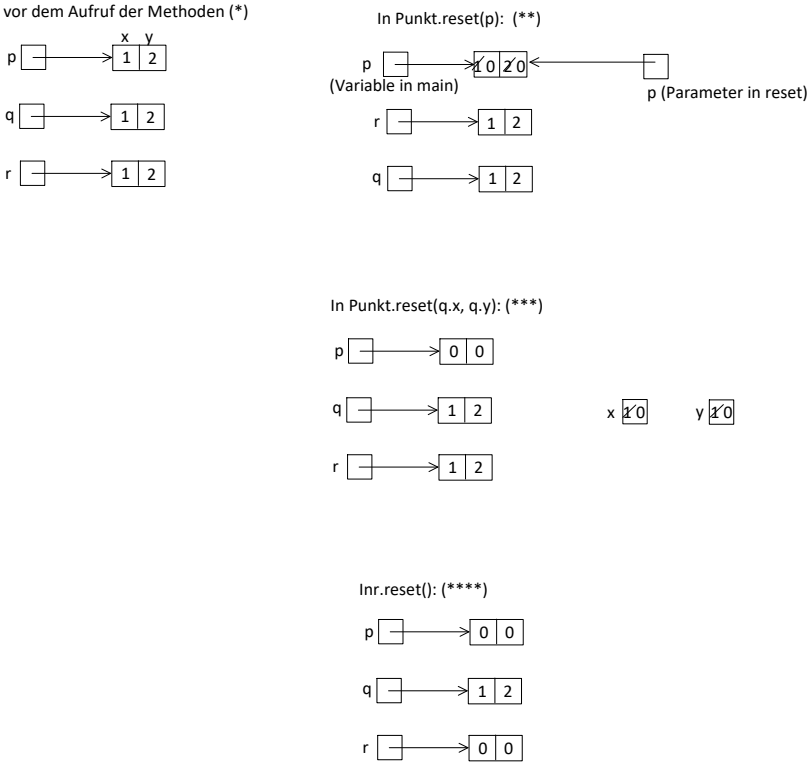
Aufgabe 4 (10 P)
Was ist die Ausgabe des Programms?

Erläutern Sie durch Speicherbilder, wie es dazu kommt. In den Speicherbildern sollten Sie die Situation *vor dem Aufruf* und *innerhalb* der aufgerufenen Methoden, also an den (*) - (****) markierten Stellen darstellen.

Lösung 4
Die Ausgabe des Programms ist

```
(0, 0)
(1, 2)
(0, 0)
```

Die zweite Methode ist nicht geeignet, denn es werden nur Kopien der Werte von q.x und q.y an de lokalen Variablen x und y übergeben.



Speicherbild Seite 1

Aufgabe 5 (12 P)
Wir wollen mit Artikeln (vom Typ String) handeln:

- Ein Kunde möchte einen Artikel kaufen. Dazu bestellt er den Artikel bei seinem Lieblings-Lieferanten.
- Der Lieferant fordert seinerseits den Hersteller auf, den Artikel zu liefern.
- Der Hersteller liefert aber nur Hosen. Alle anderen Artikel sind „zur Zeit nicht lieferbar“.

Im folgenden sehen Sie entsprechende Klassendefinitionen. Ergänzen Sie den angegeben Code:

a) Definieren Sie eine Klasse `NichtlieferbarException` als Unterklasse von `Exception`.

b) Der Hersteller löst bei Artikeln, die nicht „hose“ sind, eine `NichtlieferbarException` mit entsprechendem Fehlertext aus.

c) Der Lieferant propagiert die Fehlermeldung - sofern sie auftritt - an den Kunden.

d) Der Kunde fängt die Fehlermeldung ab und gibt ggf. den Fehlertext aus.

Sie können für die Aufgabenteile b) - d) den Code direkt auf der folgenden Seite an den entsprechenden Stelle ergänzen. Ansonsten geben Sie die Nummern der Zeilen an, die Sie verändern möchten.

Lösung 5

```
public class NichtLieferbarException extends Exception {

   public NichtLieferbarException() {
      super();
   }

   public NichtLieferbarException(String txt) {
      super(txt);
   }
}

public class Artikel {
   // unveraendert
}

public class Hersteller {

   Artikel hose = new Artikel();

   public Artikel liefern(Artikel artikel) throws NichtLieferbarException
   {
      if (artikel.equals(hose))
         return new Artikel();
      else
         throw new NichtLieferbarException(artikel + " zur Zeit nicht
             lieferbar.");
   }
}

public class Lieferant {

   Hersteller h;

   public void bestellen(Artikel artikel) throws NichtLieferbarException
   {
      h.liefern(artikel);
   }
}

public class Kunde
{

   private Lieferant bmazon;

   public void kaufen(Artikel artikel)
   {
      try {
         bmazon.bestellen(artikel);
      }
      catch (NichtLieferbarException e) {
         System.out.println(e.getMessage());
      }
   }
}
```

Aufgabe 6 (12 P)
Betrachten Sie den Java-Code auf der folgenden Seite.

a) Definieren Sie ein Interface `ClassName` (mit der/den notwendige/n Methode/n.)

b) Ergänzen Sie die Klassen `A`, `B` und `C` so, dass sie das Interface `className` implementieren und dass jedes `x` auf die Nachricht `myClassName()` im Rumpf der Methode `test()` mit seinem richtigen Klassennamen antwortet.

c) Ergänzen Sie die Klasse `Test` um eine main-Methode, in der `test()` dreimal aufgerufen wird: Jeweils für ein Objekt der Klasse `A`, eines der Klasse `B` und eines von `C`.

Sie können für die Aufgabenteile b) und c) den Code direkt auf der folgenden Seite an den entsprechenden Stellen ergänzen. Ansonsten geben Sie die Nummern der Zeilen an, die Sie verändern möchten.

Lösung 6

```
public interface ClassName {
        void myClassName();
}


public class A implements ClassName {
        public void sagA() {
                System.out.println("A");
        }

        public void myClassName() {
                sagA();
        }
}


public class B implements ClassName {
        public void sagB() {
                System.out.println("B");
        }

        public void myClassName() {
                sagB();
        }
}


public class C implements ClassName {
        public void sagC() {
                System.out.println("C");
        }

        public void myClassName() {
                sagC();
        }
}

public class Test {

        public static void main(String[] args) {
                ClassName x;
                x = new A();
                test(x);
                x = new B();
                test(x);
                x = new C();
                test(x);
        }

        public static void test(ClassName x) {
                x.myClassName();
        }
}
```

Aufgabe 7 (18 P)
Im deutschen Bundestag mit seinen derzeit 736 Mitgliedern werden Mitglieder mit gleichen (Nach-)Namen nicht durch ihren Vornamen, das Geburtsjahr oder andere Merkmale unterschieden, sondern dadurch, dass der Nachname um eine Nummer erweitert wird.

a) Definieren Sie eine Unterklasse `MdB` von `Person`, die als weiteres Attribut einen int-Wert `nr` (mit entsprechendem getter) hat. Der Defaultwert für dieses Attribut ist 0, es kann aber durch eine set-Methode geändert werden.

   Den Code der Klasse `Person` finden Sie im Anhang.

b) Die toString-Methode von MdB-Objekten liefert den Namen und - nur, falls der nr-Wert $\neq 0$ ist - den Wert von `nr`.

c) Definieren Sie in der Testklasse eine statische Methode `unify()`, die ein Array von MdB-Objekten erhält und die darin enthaltenen Objekte in folgender Weise „unifiziert“: Mehrere Objekte mit gleichem Namen erhalten jeweils einen um 1 höheren nr-Wert. (Der Name selbst wird nicht verändert!)

d) Definieren Sie (ebenfalls in der Testklasse) eine statische Methode `printA`, die das übergebene Array von MdB-Objekten ausgibt.

e) Ergänzen Sie die main-Methode an den markierten Stellen um die entsprechenden Methoden-Aufrufe.

Beispiel: Falls `init()` das Array `a` mit diesen Objekten belegt

Meier, 1970
Mueller, 1971
Schmitz, 1972
Meier, 1973
Meier, 1974
Mueller, 1975
Mueller, 1976
Schmitz, 1977
Schulz, 1978
Meier, 1979

sollte die Ausgabe des Arrays nach der „Unifizierung“ so aussehen:

Meier
Mueller
Schmitz
Meier1
Meier2
Mueller1
Mueller2
Schmitz1
Schulz
Meier3

Noch einmal der Hinweis: Das Attribut `name` eines MdB-Objektes soll nicht verändert werden, nur die Nummer `nr`. Die veränderte Ausgabe ergibt sich durch die `toString()`-Methode!

Lösung 7

```
public class MdB extends Person {

   private int nr;

   public MdB(String s, int n) {
      super(s, n);
      nr = 0;
   }

   public void setNr(int i) {
      this.nr = i;
   }

   public int nr() {
      return this.nr;
   }

   public String toString() {
      String s = this.name();
      if (this.nr > 0)
         s += this.nr;
      return s;
   }
}


public class Test {

   private static MdB[] a = new MdB[10];

   public static void main(String[] args) {
      init();
      unify(a);
      printA(a);
   }

   public static void init() {
      // wie oben
   }

   public static void printA(MdB[] a) {
      for (int i = 0; i < a.length; i++)
         System.out.println(a[i]);
   }

   public static void unify(MdB[] m) {
      int c;
      for (int i = 0; i < m.length; i++) {
         c = 1;
         for (int j = i+1; j < m.length; j++) {
            if (m[i].name().equals(m[j].name()) && m[i].nr() == 0) {
               m[j].setNr(c++);
            }
         }
      }
   }
}
```