Sigrid Weil

Klausur SS 2020
- Musterlösung -

Einführung in die Programmierung

Aufgabe 1 (15 P)
Implementieren Sie eine Klasse `Artikel` mit folgenden Eigenschaften: Artikel haben

- eine Bezeichung `bez` vom Typ `String`,
- eine Artikelnummer `nr` vom Typ `int` und
- einen (Netto-)Preis `nettoPreis` vom Typ `double`.
- Die Bezeichung und der Netto-Preis werden bei Erzeugung eines Objektes als Parameter übergeben.
- Die Artikelnummer wird bei Erzeugung eines Objektes automatisch fortlaufend vergeben.
- Außerdem gibt es einen Mehrwertsteuersatz (`mwst` vom Typ `double`), der zu Beginn (für alle Artikel) bei 19 Prozent, also 0.19 liegt.
- Der Mehrwertsteuersatz kann durch eine Methode `setMwst()` geändert werden. Er ändert sich dann für alle Artikel auf den übergebenen neuen Wert.
- Für Artikel gibt es eine Methode `bruttoPreis()`, die aus dem Netto-Preis und dem aktuell gültigen Mehrwertsteuersatz den Brutto-Preis berechnet.
  (Die Berechnungsformel dafür lautet bruttoPreis $=$ nettoPreis $\cdot\,(1 +$ mwst$)$)
- Die `toString`-Methode soll Artikelnummer und Bezeichnung liefern.
- Nur der Konstruktor und die Methoden sollen von außerhalb der Klasse zugreifbar sein.

Lösung 1

- InstanzAttribut, 1P
- InstanzAttribut 1P
- Instanzattribut. 1P
- Params in Konstruktor 1P
- Klassenattribut + incr im Konstruktor 2P
- Klassenattr. 2P
- Klassenmeth 2P
- InstanzMeth 2 P
- `toString`-Methode 1P
- private/public 2P

```java
public class Artikel {
        // Klassenattribute
        private static int nextNr = 1;
        private static double mwst = 0.19;

        // Instanzattribute
        private String bez;
        private int nr;
        private double nettoPreis;

        // Konstruktor
        public Artikel(String bez, double preis) {
                this.bez = bez;
                this.nettoPreis = preis;
                this.nr = nextNr++;
        }

        // Klassenmethode
        public static void setMwst(double neu) {
                mwst = neu;
        }
```

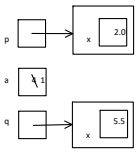
```
        // Instanzmethoden
        public double bruttoPreis() {
                return this.nettoPreis * (1 + mwst);
        }

        public String toString() {
                return this.nr + " " + this.bez;
        }

}
```

Aufgabe 2 (5+10 P)
Betrachten Sie die Klasse `A` und die main-Methode wie unten angeben.

a) Zeichnen Sie die Speicherbild-Situation nach der zweiten der vier Anweisungen der main-Methode, also an der mit „(*)“ gekennzeichneten Stelle.

b) Vervollständigen Sie die unten angedeutete Trace-Tabelle um entsprechende Spaltenüberschriften. Tragen Sie die Werte aller vorkommenden Variablen und die Änderung dieser Werte nach den jeweiligen Anweisungen ein.

Lösung 2
a) 2P für p und q als Refernzenvar.
1P für a als simpleType-Var
2P für korrekte Inhalte



Konzept Seite 1

b) 2 P für korrekte Spalten
2P pro korrekte Zeile

Anweisung	a	p.x	q.x
`A p = new A(4, 2.0);`	4	2.0	
`A q = new A(1, 5.5);`	1		5.5
`p.incr();`	2	3.0	
`q.incr();`	3		6.5

Aufgabe 3 (5+2+4+4 P)

a) Schreiben Sie eine (statische) Methode `teilSumme()`, die ein `double`-Array und zwei Positionsnummern i und j (vom Typ `int`) als Eingabeparameter erhält.

Die Methode soll die Teilsumme der Array-Einträge von Position i (einschließlich) bis Position j (ausschließlich) berechnen und zurückgeben.

b) Definieren Sie eine Fehlerklasse `FalscheParameterException`, die von `Exception` abgeleitet wird und Fehlermeldungen mit dem Fehlertext „unzulaessige Parameter“ erzeugt.

c) Falls für die Methode aus a) die Eingabeparameter i oder j unzulässige Werte darstellen, soll die Methode eine `FalscheParameterException` auslösen.
Unzulässige Werte sind solche, die Array-Grenzen überschreiten oder für die $i \geq j$ gilt.

d) Schreiben Sie eine (statische) Methode `testTeilSumme`, in der die Methode `teilSumme` für das Array $\{1.0, 2.0, 3.0\}$ und beliebige Parameter i, j aufgerufen wird. Gegebenenfalls auftretende Exceptions sollen durch einen Exception-Handler abgefangen werden.

(Sie *können* - müssen aber nicht - für die Aufgabenteile a) und c) eine gemeinsame Lösung abgeben.)

Lösung 3

a) 2P für Methodenkopf 3P für korrekte Summenberechnung

b) 2 P für ExceptionKlasse

c) 1P throws // 1P für if - throw new // 2 P richtige Bedingung im if

d) 1P für Kopf von testTeilSumme 2 P für try-catch
1P für Methodeaufruf von teilSumme

```
public static double teilSumme(double[] a, int i, int j)
                                        throws FalscheParameterException {
        if(i < 0 || j < 0 || i >= a.length || j > a.length || i >= j)
                throw new FalscheParameterException();
        double s = 0;
        for (int n = i; n < j; n++) {
                s += a[i];
        }
        return s;
}

public class FalscheParameterException extends Exception {

        public FalscheParameterException() {
                super("unzulaessige Parameter");
        }
}
```

Aufgabe 4 (3 + 12 P)
Gegeben sei das folgende Interface

```
public interface Turtle {

        void step(); // ein Schritt in Blickrichung

        void step(int n); // n Schritte in Blickrichtung

        void turnR(); // Vierteldrehung gegen den Uhrzeiger

        void turnL(); // Vierteldrehung mit dem Uhrzeiger
}
```

Nehmen Sie an, die Klasse `Roboter` implementiere dieses Interface.
Die Klasse verfüge über einen parameterlosen Konstruktor, der einen Roboter mit Startposition $(0, 0)$ und Blickrichtung „rechts“ erzeugt.

a) Definieren Sie ein Interface `TreppenTurtle`, das das Interface `Turtle` um eine Methode `void treppe(int n)` erweitert.

b) Die Methode `treppe(int n)` ermöglicht den Objekten, treppenförmige Wege zu laufen (siehe unten). Der Parameter n gibt die Anzahl der Stufen an.
Implementieren Sie eine Klasse `TreppenRoboter`, die von `Roboter` abgeleitet ist und zusätzlich das Interface `TreppenTurtle` implementiert.
(Beachten Sie bei der Implementierung, dass die Treppe eine rechts-Bewegung mehr erfordert als nach-oben-Bewegungen.)

Lösung 4

```
// 2 P

public interface TreppenTurtle extends Turtle{

        void treppe(int n);

}


public class TreppenRoboter extends Roboter implements TreppenTurtle{ // 2P

        public void treppe(int n) {
                int j = 1;                          // 2P für richitge Init.
                    und Verwendung
                for (int i = 1; i <= n; i++) { // 1 P für Schleifenkopf
                        this.step(j);
                        this.turnL();
                        this.step(j++);
                        this.turnR();
                        // 4 P für "Treppenverlauf"
                }
                this.step(j); // 1P für richitge Anzahl an Teilstücken
        }

}
```

Aufgabe 5 (4 + 6 P)

$$z(t) = z(t-1) + n(t-1)$$
$$n(t) = n(t-1) \cdot r \cdot \frac{p - z(t-1)}{p}$$

Die Startwerte für n und z sind $n_0 = n(0) = 1$ und $z_0 = z(0) = 0$.

Lösung 5

a) Für $r = 2$ und $p = 100$ ist

$$z_1 = z_0 + n_0 = n_0 = 1$$
$$n_1 = n_0 \cdot r \cdot \frac{p - z_0}{p} = n_0 \cdot 2 \cdot 1 = 2$$
$$z_2 = z_1 + n_1 = 3n_2 \qquad = n_1 \cdot r \cdot \frac{p - z_1}{p} = 2 \cdot 2 \cdot \frac{97}{100} = \frac{388}{100} = 3.88$$

b)

```
public class Corona {

        private static double p = 80000000; // Bevoelkerung
        private static double r = 2; // Basisreproduktionszahl

        // iterative Lösung
        public static double zahl(int t) {
                double z0 = 0; // Zahl der anfangs infizierten Personen
                double n0 = 1; // Zahl der Neuinfektionen
                double z ; // Zahl der insgesamt inifzierten Personen
                double n ; // Zahl der Neuinfektionen  1 Tag spaeter

                for (int i = 1; i <= t; i++) {
                        z = z0+n0;
                        n = n0*r*(p-z0)/p;
                        n0 = n;
                        z0 = z;
                }
                return z;
        }

        // rekursive Lösung, benötigt zwei Methoden
        public static double z(int t) {
                if(t == 0)
                        return 0;
                else
                        return z(t-1) + n(t-1);
        }

        public static double n(int t) {
                if (t == 0)
                        return 1.0;
                else
                        return n(t-1) * r * (p-z(t-1))/p;
        }
}
```