Programmierkurs
Wintersemester 2020/21
Übungsblatt 07 (21.12.2020)

M.Sc. Laslo Hunhold
Dept. Math./Inf., Abt. Informatik
Universität zu Köln

Abgabe für Rückmeldungen bis zum 11.01.2021 (08:00 Uhr), Besprechung ab dem 11.01.2021.

Aufgabe 07-0 (Schlüsselwort `this`)
Betrachten Sie folgende Klassen.

- ```
  public class Klasse1 {
      int wert;

      boolean groesser_als(Klasse1 x) {
          return (wert > x.wert);
      }
  }
  ```

- ```
  public class Klasse3 {
      int wert;

      boolean groesser_als(int wert) {
          return (wert > wert);
      }
  }
  ```

- ```
  public class Klasse2 {
      int wert;

      boolean groesser_als(Klasse2 x) {
          return (this.wert > x.wert);
      }
  }
  ```

- ```
  public class Klasse4 {
      int wert;

      boolean groesser_als(int wert) {
          return (this.wert > wert);
      }
  }
  ```

Entscheiden und begründen Sie, ob die folgenden Aussagen wahr oder falsch sind.

0. Seien `a` und `b` Zeiger auf Instanzen von `Klasse1`. Der Ausdruck „`a.groesser_als(b)`“ ist genau dann `true`, wenn die Variable `wert` in `a` größer als die Variable `wert` in `b` ist.
1. Seien `a` und `b` Zeiger auf Instanzen von `Klasse2`. Der Ausdruck „`a.groesser_als(b)`“ ist genau dann `true`, wenn die Variable `wert` in `a` größer als die Variable `wert` in `b` ist.
2. Sei `a` ein Zeiger auf eine Instanz von `Klasse3`. Der Ausdruck „`a.groesser_als(4)`“ ist genau dann `true`, wenn die Variable `wert` in `a` größer als 4 ist.
3. Sei `a` ein Zeiger auf eine Instanz von `Klasse4`. Der Ausdruck „`a.groesser_als(4)`“ ist genau dann `true`, wenn die Variable `wert` in `a` größer als 4 ist.
4. Es ist fehlerfrei möglich, `Klasse1` zu erweitern, indem man die Methode `groesser_als()` aus `Klasse4` unverändert am Ende von `Klasse1` einfügt.

Aufgabe 07-1 (Methoden)
Betrachten Sie folgendes lückenhafte Listing.

```
1  public class Array {
2      private int[] arr;
3
4      public void print(String name) {
5          System.out.print(name + ":");
6          for(int i = 0; i < arr.length; i++){
7              System.out.print(" " + arr[i]);
8          }
9          System.out.print('\n');
10     }
11
12     public Array(int[] arr) { ... }
13     public Array erzeuge_kopie() { ... }
14     public void nulle_vielfache_von(int a) { ... }
15     public int summe() { ... }
16 }
```

Vervollständigen Sie die Methodenrümpfe so, daß sie folgende Bedingungen erfüllen:

- Der Konstruktor `Array(int[] arr)` soll eine Kopie des ihm übergebenen Arrays `arr` erzeugen und diese in der Instanzvariable `arr` speichern. Achten Sie darauf, daß Sie nicht nur den Zeiger kopieren, sondern für die Kopie ein neues Objekt erzeugen.
- Die Methode `erzeuge_kopie()` soll eine Kopie der aufrufenden Instanz erzeugen. Insbesondere soll, analog zum Konstruktor, in der Kopie eine Kopie des Arrays `arr` vorliegen (und nicht nur der Zeiger kopiert werden).
- Die Methode `nulle_vielfache_von(int a)` soll alle Einträge von `arr`, die im Wert Vielfache von `a` sind, auf 0 setzen.
- Die Methode `summe()` soll die Summe der Werte von `arr` zurückgeben.

Testen Sie Ihr Programm mit folgender `Main`-Klasse und prüfen Sie die Ausgabe.

```
1  public class Main {
2      public static void main(String[] args) {
3          int[] a = {
4              1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
5          };
6          Array b = new Array(a), c;
7
8          b.print("b");
9
10         c = b.erzeuge_kopie();
11         c.print("c");
12
13         b.nulle_vielfache_von(3);
14         b.print("b (nach Entfernung)");
15         c.print("c (nach Entfernung)");
16
17         System.out.println("Summe von b: " + b.summe());
18         System.out.println("Summe von c: " + c.summe());
19     }
20 }
```

Präsenzaufgabe 07-2 (Vererbung und `instanceof`)

Betrachten Sie folgende Listings.

```
1  public class Produkt {
2      private static int zaehler = 0;
3      private int artnr;
4
5      Produkt() {
6          artnr = zaehler++;
7      }
8      void info() {
9          System.out.println("Artikel mit " + "Artikelnummer " + artnr);
10     }
11 }
```

```
1  public class Buch extends Produkt {
2      private int isbn;
3
4      Buch(int isbn) {
5          this.isbn = isbn;
6      }
7      void info() {
8          super.info();
9          System.out.println("\tBuch mit ISBN " + isbn);
10     }
11 }
```

```
1  public class Softcover extends Buch {
2      public Softcover(int isbn) {
3          super(isbn);
4      }
5      public void info() {
6          super.info();
7          System.out.println("\tTyp: Softcover");
8      }
9  }
```

```
1  public class Hardcover extends Buch {
2      public final double einbanddicke; /* mm */
3
4      Hardcover(int isbn, double einbanddicke) {
5          super(isbn);
6          this.einbanddicke = einbanddicke;
7      }
8      public void info() {
9          super.info();
10         System.out.println("\tTyp: Hardcover");
11     }
12 }
```

Die Klasse `Buch` ist eine Subklasse von `Produkt` und die Klassen `Softcover` und `Hardcover` sind Subklassen von `Buch`. Betrachten Sie folgendes lückenhafte Listing.

```
1  class Main {
2      public static void main(String[] args) {
3          Produkt[] p = {
4              new Buch(938145623),
5              new Softcover(924266821),
6              new Hardcover(232812134, 2),
7              new Softcover(283848228),
8              new Produkt(),
9              new Hardcover(237277173, 4),
10             new Buch(492581833),
11             new Hardcover(882491984, 5),
12         };
13
14         for (int i = 0; i < p.length; i++) {
15             p[i].info();
16         }
17
18         ...
19     }
20 }
```

Füllen Sie die Lücke so aus, daß das Programm neben der Auflistung aller Produkte im Sortiment `p` auch noch jeweils eine Auflistung aller Bücher, aller Softcover und aller Hardcover macht. Geben Sie bei den Hardcovern zusätzlich noch die Variable `einbanddicke` (in mm) aus.
Hinweis: Nutzen Sie den `instanceof`-Operator zur dynamischen Typprüfung und einen Cast.

Präsenzaufgabe 07-3 (Fakultät)

Schreiben Sie ein Programm, welches für eine gegebene nichtnegative ganze Zahl n ihre Fakultät $n!$ berechnet. Beachten Sie, daß $0! = 1! = 1$ gilt.
Versehen Sie Ihr Programm mit einer sinnvollen Kommandozeilenausgabe und implementieren Sie es sowohl iterativ als auch rekursiv.

Präsenzaufgabe 07-4 (Fibonacci)

Die Fibonaccifolge a_n für $n \in \mathbb{N}$ ist definiert als $a_0 := 1$, $a_1 := 1$ und $a_n := a_{n-2} + a_{n-1}$ für $n > 1$. Schreiben Sie ein Programm, welches die $n$te Fibonaccizahl a_n berechnet.
Versehen Sie Ihr Programm mit einer sinnvollen Kommandozeilenausgabe und implementieren Sie es sowohl iterativ als auch rekursiv.