Übungsblatt 05

3 Aufgaben, 20 Punkte

Objektorientierte Modellierung und Programmierung (inf031) Sommersemester 2020
Carl von Ossietzky Universität Oldenburg, Fakultät II, Department für Informatik

Dr. C. Schönberg

Ausgabe: 2020-05-19 14:00 **Abgabe:** 2020-05-26 12:00

Aufgabe 1: *Exceptions* **(6 Punkte)**

Ein Programmierer hat anderen Programmierern folgende Klasse `Util` mit nützlichen Methoden zur Verfügung gestellt:

```
1  class Util {
2
3    // liefert die kleinste Zahl des uebergebenen Arrays
4    public static int minimum(int[] values) {
5      int min = values[0];
6      for (int i = 1; i < values.length; i++) {
7        if (values[i] < min) {
8          min = values[i];
9        }
10     }
11     return min;
12   }
13
14   // konvertiert den uebergebenen String in einen int-Wert
15   public static int toInt(String str) {
16     int result = 0, factor = 1;
17     char ch = str.charAt(0);
18     switch (ch) {
19     case '-':
20       factor = -1;
21       break;
22     case '+':
23       factor = 1;
24       break;
25     default:
26       result = ch - '0';
27     }
28     for (int i = 1; i < str.length(); i++) {
29       ch = str.charAt(i);
30       int ziffer = ch - '0';
31       result = result * 10 + ziffer;
32     }
33     return factor * result;
34   }
35
36   // liefert die Potenz von zahl mit exp,
```

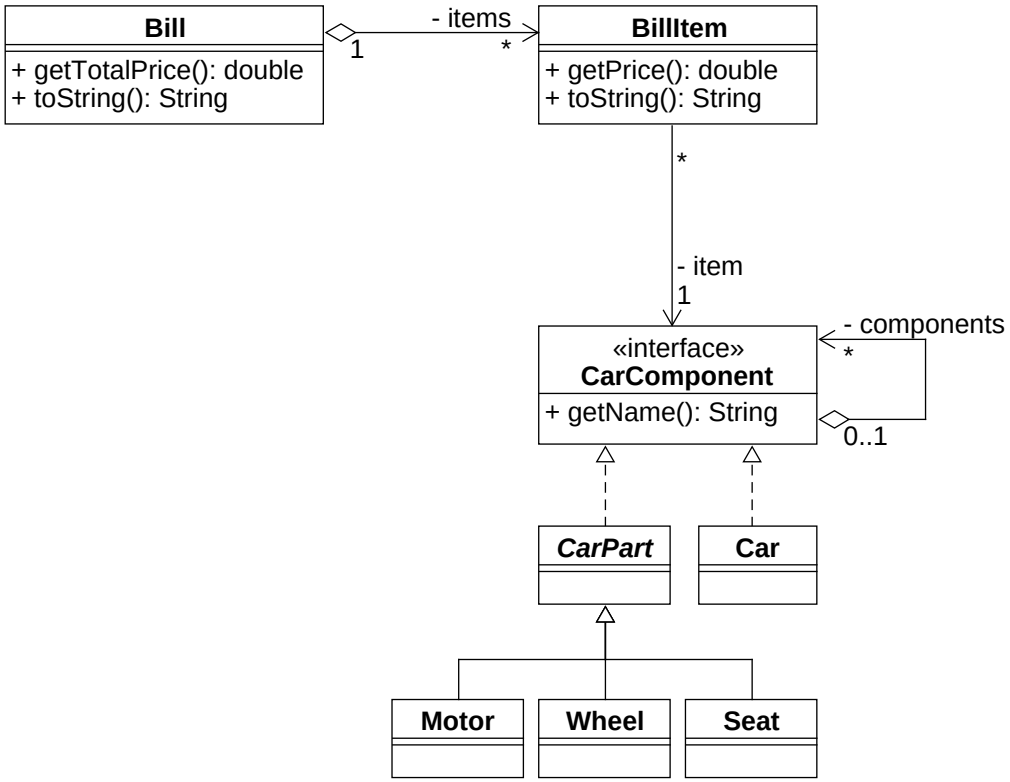
```
37    // also zahl ”hoch” exp (number to the power of exp)
38    public static long power(long number, int exp) {
39      if (exp == 0) {
40        return 1L;
41      }
42      return number * Util.power(number, exp - 1);
43    }
44  }
45
46  public class UtilTest {
47
48    // Testprogramm
49    public static void main(String[] args) {
50      String eingabe = IO.readString(”Zahl: ”);
51      int zahl = Util.toInt(eingabe);
52      System.out.println(zahl + ” hoch ” + zahl + ” = ”
53          + Util.power(zahl, zahl));
54      System.out.println(Util.minimum(new int[] { 1, 6, 4, 7, -3, 2 }));
55      System.out.println(Util.minimum(new int[0]));
56      System.out.println(Util.minimum(null));
57    }
58  }
```

Er vertraut dabei darauf, dass beim Aufruf der Funktionen semantisch gültige Parameterwerte übergeben werden.
Sie finden die Klasse `Util` prinzipiell sinnvoll, haben jedoch kein solches Vertrauen in Programmierer, die die Klasse nutzen wollen. Sie möchten sicherstellen, dass andere Programmierer über ungültige Parameterwerte informiert werden.

- Überlegen Sie bei den einzelnen Methoden, welche ungültigen Parameterwerte übergeben werden können.
- Definieren Sie hierfür adäquate Exception-Klassen.
- Schreiben Sie die Klasse `Util` so um, dass in den einzelnen Methoden die übergebenen Parameterwerte überprüft und gegebenenfalls entsprechende Exceptions geworfen werden. Versehen Sie die Signatur der Methoden mit den entsprechenden Exceptions. Es muss sichergestellt sein, dass bei der Ausführung der Methoden keine anderen als die in der Signatur deklarierten Exceptions geworfen werden.
- Passen Sie das Testprogramm entsprechend an. Exceptions sollen abgefangen und adäquate Fehlermeldungen ausgegeben werden.

Aufgabe 2: *Kapselung* **(6 Punkte)**

Gegeben sei folgendes rudimentäres Klassendiagramm zur Modellierung von Rechnungen eines Autoverkäufers:



Implementieren Sie das modellierte System in Java. Verwenden Sie dabei an *sinnvollen Stellen*

- mindestens eine *Inner Class*,
- mindestens eine *Static Class*,
- mindestens eine *Final Class* und
- mindestens ein *Package*.

Die `toString()`-Methoden von `BillItem` und `Bill` sollen jeweils den Namen (und die Namen aller Teil-Komponenten) und den Preis des Rechnungspostens bzw. alle Einzelposten und die Gesamtrechnungssumme ausgeben.

Schreiben Sie außerdem ein Testprogramm, das folgende Rechnung erzeugt und ausgibt:

```
Rolls Royce (Motor ): 100000.0
Seat: 2000.0
Wheel: 1000.0
Wheel: 1000.0
Wheel: 1000.0
Wheel: 1000.0
Total: 106000.0
```

Aufgabe 3: *Generics* **(8 Punkte)**

Gegeben Sei eine Schnittstelle, welche zwei Objekte vom generischen Typ T bezüglich ihres Alters vergleichen kann:

```
1 public interface Older<T> {
2   public boolean isOlder(T other);
3 }
```

Schreiben Sie eine Klasse `Person`, die diese Schnittstelle implementiert. Personen haben einen Namen und ein Alter, über das sie verglichen werden.
Schreiben Sie eine zweite Klasse `Group`, die eine Liste von Mitgliedern mit generischem Typ T (also eine `List<T>`) hat. Einer Gruppe kann man Mitglieder hinzufügen (**`public void`** `add(T member)`) und man kann das älteste Mitglied der Gruppe bestimmen (**`public`** `T getOldest()`).

Schreiben Sie die Klassen so, dass folgende Testklasse compiliert und das Ergebnis `Carl` ausgibt:

```
1 public class TestGroup {
2   public static void main(String[] args) {
3     Group<Person> group = new Group<>();
4     group.add(new Person("Alice", 25));
5     group.add(new Person("Bob", 23));
6     group.add(new Person("Carl", 26));
7     System.out.println(group.getOldest().getName());
8   }
9 }
```

Hinweis: Achten Sie darauf, dass Sie in der Klasse `Group` den generischen Typ T ggf. genauer qualifizieren (einschränken) müssen.