Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. M. Wirsing, M. Hammer, A. Rauschmayer

SS 2006
Übungsblatt K

Informatik II

Dieses Blatt ist nicht abzugeben, es wird besprochen in der Zentralübung. Die Musterlösung wird vor der Klausur online gestellt.

Aufgabe K-1 **Adapter-Pattern** (`PrintStreamTreePrinter.java`)

Gegeben sei die Klasse `StringTree`, die Bäume von Zeichenketten verwaltet (herunterladbar von der Info-II-Homepage). Die Methode `printTo(TreePrinter)` setzt die Schnittstelle `TreePrinter` ein, um Bäume auszugeben:

```
package adapter;

/**
 * Internally keeps an indentation counter that is initially 0.
 * Every indentation level is printed as two spaces.
 */
public interface TreePrinter {
    /**
     * Write a string, consider the current indentation.
     *
     * @param str
     *            the string to be written
     */
    void println(String str);

    /** Increment the current indentation by one */
    void incIndent();

    /** Decrement the current indentation by one */
    void decIndent();
}
```

Schreiben Sie einen Adapter `PrintStreamTreePrinter`, so dass die `printTo`-Methode mit einem `java.io.PrintStream` (in diesem Fall `System.out`) wie folgt verwendet werden kann:

```
    public static void main(String[] args) {
        StringTree tree = new StringTree("Root")
            .addChild(new StringTree("A")
                    .addChild(new StringTree("A1"))
                    .addChild(new StringTree("A2")))
            .addChild(new StringTree("B"));
        tree.printTo(new PrintStreamTreePrinter(System.out));
    }
```

Das soll zu der untenstehenden Ausgabe führen:

```
Root
  A
    A1
    A2
  B
```

Aufgabe K-2 **Externe Hilfsmethoden** (`Util.java`)

Implementieren Sie eine Klasse `Util`, mit folgenden statischen Hilfsmethoden:

- `public static String removePrefixSafely(String str, String prefix)`
  Wenn `str` das Präfix `prefix` hat, soll dieses in der Rückgabe entfernt werden, andernfalls wird `str` unverändert zurückgegeben.
- `public static String join(Collection<?> elements, String separator)`
  Dies Methode wandelt `elements` in Zeichenketten um und hängt diese aneinander, wobei zwischen jeweils zwei Elementen der Trenner `separator` steht. Beispiel: `join(Arrays.asList(``a'', ``b'', ``c''), ``-'')` ist die Zeichenkette ``a-b-c''.
- `escapeNull`: hat zwei Argument `value` und `defaultValue`. Wenn `value` nicht `null` ist, soll `value` zurückgegeben werden, andernfalls der Default-Wert. Verwenden Sie bei der Signatur dieser Methode einen Typparameter.
- `equals`: Vergleiche zwei Argumente `left` und `right`, so dass der Vergleich auch funktioniert, wenn eines der Argumente `null` ist. Gehen Sie davon aus, dass beide Argumente den gleichen Typ haben und drücken Sie die Signatur mit einem Typparameter aus.

Mit der (herunterladbaren) Klasse `UtilTest` können Sie Ihre Implementierung auf Korrektheit überprüfen.

Aufgabe K-3 **OCL** (`ocl.txt, Taxometer.java`)

Ein Taxometer (Gerät zur Berechnung des Fahrtpreises in Taxis) soll in Java implementiert werden. Dabei soll eine Klasse `Taxometer` mit OCL-Spezifikationen versehen werden.

Ein Taxometer speichert die bisherigen Kosten der Fahrt in einem Attribut `charge : double`. Eine Variable `running : boolean` speichert, ob das Taxometer eingeschaltet ist. Es gibt vier Methoden:

- `switchOn() : void` - schaltet das Taxometer ein. Dabei soll `charge` auf 5.0 gesetzt werden.
- `switchOff() : void` - schaltet das Taxometer aus. `charge` wird dabei nicht verändert.
- `tick() : void` - wird aufgerufen, um `charge` um 0.5 zu erhöhen.
- `getCost(): double` - gibt den Stand von `charge` zurück.

a) Spezifizieren Sie in OCL folgende Eigenschaften:

- `switchOn()` darf nur aufgerufen werden, wenn `running` auf `false` steht. Nach der Beendigung der Methode soll `running` auf `true` stehen.
- Für `switchOff()` verhält es sich gerade andersrum: Beim Aufruf muss `running` auf `true` stehen, danach auf `false`.
- Nach der Terminierung von `switchOn()` steht `charge` auf 5.0.
- `tick()` darf nur aufgerufen werden, wenn `running` auf `true` steht. Nach der Terminierung soll `charge` um 0.5 erhöht worden sein.
- `getCost()` darf stets aufgerufen werden, und gibt den Wert von `charge` zurück.

b) Implementieren Sie die Klasse `Taxometer` in Java. Prüfen Sie dabei Vor- und Nachbedingung, wie in der Vorlesung angegeben. Definieren Sie eine geeignete Exception, die bei Verletzungen der Vorbedingung geworfen werden soll.