JOHANN WOLFGANG GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN
FB Informatik und Mathematik
Prof. R. Brause
www.cs.uni-frankfurt.de/~ppi

Programmierpraktikum Grundlagen der Programmierung WS07/08

1. Übungsblatt - Abgabe: 24.10.2007

Lernziele

Objektorientierung, Eclipse, Dokumentation, Schnittstellen, Fehlerbehandlung

Zielsetzung

In diesem Aufgabenblatt werden Sie einen Einheitenumrechner programmieren, den man über die Konsole startet. Der Einheitenumrechner ist dabei modular aufgebaut, so dass sich die Art der Einheiten, mit dem er rechnet (Entfernung, Währung, Temperatur, etc.) leicht verändern lassen. Dieses Blatt ist in Einzelarbeit zu lösen, damit jeder Teilnehmer einen Einstieg in das Praktikum erhält.

Aufgabe 1.1

a) Schreiben Sie eine Klasse `Währungsrechner`, welche das Interface `Umrechner` implementiert und mindestens *Euro* und *Dollar* sowie zwei weitere Währungen ineinander umrechnet. Nachdem Sie diese Klasse geschrieben haben, kopieren Sie sie und machen aus der Kopie eine zweite Klasse `Entfernungsrechner` mit verschiedenen Entfernungseinheiten (z.B. *Zentimeter*, *Meter*, *Meilen*). Auch diese Klasse muss mindestens vier verschiedene Einheiten anbieten.
Die Klassen muss das folgende Interface `Umrechner` implementieren. Die Erklärungen der einzelnen Funktionen steht in den JavaDoc-Kommentaren des Interfaces.

Interface `Umrechner`:

```
package blatt1;
import java.util.Vector;

/**
* Interface für den Umrechner. Ermöglicht der GUI Zugriff auf die Klasse.
* @author Markus Palcer
* @version 1.0
*/
public interface Umrechner
{
        /**
        * Gibt einen Vektor mit allen verfügbaren Einheiten zurück
        * @return Alle verfügbaren Einheiten
        */
        public Vector<String> getTypes();
        /**
        * Startet die Berechnung
        * @param value Der Wert, der umgerechnet werden soll
        * @param src Die Einheit, in der dieser Wert vorliegt
        * @param tgt Die Einheit, in die umgewandelt werden soll
        * @return Den umgerechneten Wert
        * @throws UnknownUnitException Eine angegebene Einheit ist nicht verfügbar
        */
        public double umrechnen(double value,String src, String tgt)
        throws UnknownUnitException;
}
```

b) Sollte eine Einheit von einer Klasse angefragt werden, welche sie nicht unterstützt, ist eine Ausnahme vom folgenden Typ `UnknownUnitException` zu werfen.

Exception `UnknownUnitException`:

```
package blatt1;

/**
* Diese Exception wird geworfen, wenn eine Einheit dem Umrechner unbekannt ist.
* @author Markus Palcer
* @version 1.0
*/

@SuppressWarnings(“serial“)
public class UnknownUnitException extends Exception
{
        private String unit=;
        /**
        * Erstellt die Klasse
        * @param unitName Der Name der unbekannten Einheit
        */
        public UnknownUnitException(String unitName) { unit=unitName; }

        /**
        * Gibt den Namen der Einheit zurück, die der Umrechner nicht kannte.
        * @return Ein dem Umrechner unbekannter Einheitenname
        */
        public String getUnit() { return unit; }
}
```

Aufgabe 1.2

a) Programmieren sie eine Applikation in Java, welche Kommandozeilenparameter einliest und diese der Instanz einer der beiden Klassen aus Aufgabe 1.1 übergeben kann. Die Eingabe des Nutzers hat über die Kommandozeilenparameter zu erfolgen.

b) Macht der Nutzer eine Fehleingabe, so soll er auf die richtige Syntax und die möglichen Einheiten hingewiesen werden.

c) Die Instanz Ihrer Klasse aus Aufgabe 1.1 muss in einer als `Umrechner` deklarierten Variable gespeichert werden.

Aufgabe 1.3

Überlegen Sie sich diverse Testfälle für jede Klasse sowie das fertige Programm gemäß der Anforderungen aus Aufgabe 1.1 und 1.2 und testen Sie nach diesen Kriterien. Benutzen Sie auch systematisch fehlerhafte Werte, um die Fehlerbehandlung und ihre Grenzen zu testen. Notieren Sie diese Grenzen.

Aufgabe 1.4

a) Schreiben Sie die Spezifikation (Pflichtenheft) für das Programm. Die Spezifikation muss die Anforderungen an das Programm in Ihren eigenen Worten enthalten. Sie muss außerdem die Grenzen des Programms enthalten, damit von vornherein feststeht, was von ihrem Programm nicht erwartet werden kann.

b) Schreiben Sie eine Entwicklungsdokumentation zu Ihrem Programm. In der Entwicklungsdokumentation muss ersichtlich sein, wieso Sie sich für einen bestimmten Weg der Implementation entschieden haben. Die Entwicklungsdokumentation beinhaltet auch das Dokumentieren jeder einzelnen Funktion/Klasse mithilfe von JavaDoc-Kommentare. Kommentieren Sie Ihren Quellcode so, dass er auch für andere verständlich ist. Dies ist später wichtig, da Ihre Gruppenmitglieder Ihren Quellcode verstehen müssen.

c) Schreiben sie eine Testdokumentation über alle durchgeführten Tests. Die Testdokumentation hat die verwendeten Testfälle und die Testergebnisse zu enthalten. Listen Sie außerdem die durch die Tests bekannten Grenzen ihres Programms auf.

Tips

- Die Namen der unterstützten Einheiten können Sie über die Funktion `getTypes()` des Interfaces erfragen, um sie anzuzeigen.