Einführung in die Programmierung für NF

Übung 11

15.01.2014

Inhalt

- Korrektur Blatt 10
  - JList mit ListModel bzw. DefaultListModel
  - ActionListener und InputDialoge
  - UML

- Praktische Anwendung Observer-Pattern
- Programmierung von Interfaces
- Kurze Einführung in Exceptions

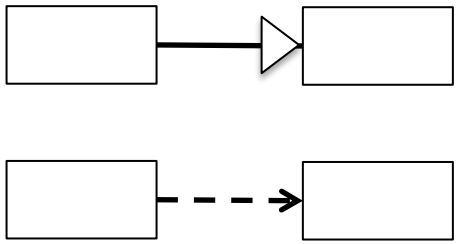
Korrektur Blatt 10

- Schwierigkeiten mit ListModel
- Einer JList wird bei Erzeugung ein ListModel (z.B. ein DefaultListModel) übergeben
- Änderungen an der Liste passieren auf bzw. mit dem ListModel

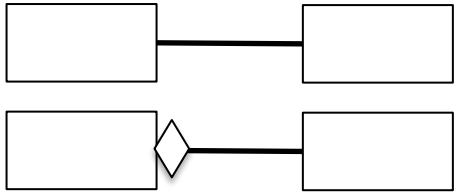
- Detaillierte Infos im empfohlenen Video: http://youtu.be/DarkE6QvBGQ

Korrektur Blatt 10

- UML
  - Vererbung von Klassen
  - Implementierung von Interfaces



- Beziehungen
  - Assoziation
  - Aggregation



Observer Pattern

- Observer „überwachen“ ein Observable
- Das Observer-Interface implementiert eine `update()`-Methode im Observer
- Diese Methode wird immer ausgeführt, wenn die Observer benachrichtigt werden:

```
setChanged();
notifyObservers();
```

- Dies geschieht im Observable

Observer Pattern

- Gemeinsame Aufgabe:
  - Implementierung von Funktionalität in der Banksimulation
  - Bei Klick auf einen Button wird nicht direkt ein Listen-Eintrag erstellt, sondern die entsprechende Methode des Models über den Controller aufgerufen
  - Die update-Methode aktualisiert alle Listen mit den entsprechenden Daten aus dem Model

Interfaces

- Interface bedeutet Schnittstelle
- Interfaces beinhalten abstrakte Klassendeklarationen, z.B. Methodenrümpfe ohne Inhalt
- Wenn eine Klasse ein Interface implementiert, müssen diese Funktionen vorhanden sein
- Sinn: Andere Klassen können statt der Klasse selbst das Interface verwenden

Interfaces

- Beispiel:

```
public interface Interface {

    void methode1(int parameter);

    int getAnInteger();

}
```

```
public class Klasse implements Interface {

    @Override
    public void methode1(int parameter) {
        // TODO Auto-generated method stub

    }

    @Override
    public int getAnInteger() {
        // TODO Auto-generated method stub
        return 0;
    }

}
```

Interfaces

- Falls es für eine Anwendung verschiedene Models gibt (z.B. ein neues Model mit einem besseren Algorithmus), müssen andere Klassen nicht verändert werden, wenn sie die Klasse über das Interface ansprechen, da sichergestellt ist dass alle bekannten Methoden auch weiterhin vorhanden sind

Exceptions

- Exceptions dienen der Fehlererkennung und Behandlung
- Während der Laufzeit eines Programms können in bestimmten Fällen Fehler passieren
- Wenn diese Fehler nicht behandelt werden, stürzt das Programm ab
- Java wirft, wenn ein solcher Fehler auftritt, eine Exception

Exceptions

- Eine der bekanntesten Exceptions:

  die NullPointerException

- Die NullPointerException wird immer dann geworfen, wenn etwas verwendet werden soll, dass es nicht gibt, also das null ist

Exceptions

- Beispiel:

```
public class NullPointerTest {

    private static Objekt Objekt;

    public static void main(String[] args) {
        System.out.println(Objekt.toString());
    }
}
```

Das Objekt wurde an der Stelle, an der es ausgegeben werden soll, noch nicht erzeugt, das Programm bricht ab

```
Exception in thread "main" java.lang.NullPointerException
        at NullPointerTest.main(NullPointerTest.java:9)
```

Exceptions

- Wenn wir vermuten, dass hier eine NullPointerException auftreten könnte, können wir diese abfangen:

```
public class NullPointerTest {

    private static Objekt Objekt;

    public static void main(String[] args) {
        try {
            System.out.println(Objekt.toString());
        } catch (NullPointerException e) {
            System.out.println("gibts net!");
        }
    }
}
```

Exceptions

- Es gibt viele Exceptions, besonders bei Netzwerkverbindungen oder wenn Dinge ineinander umgewandelt werden sollen
- Viele solche Dinge müssen mit einem try-catch-Block versehen werden
- Die Exception wird im Catch-Block mit übergeben und kann verwendet werden, z.B. zum Auslesen einer Fehlermeldung

Exceptions

- Außerdem können eigene Exceptions wie andere Klassen auch erstellt werden
- Sie erben dann von der Bibliotheksklasse Exception

```
class MyException extends Exception
{
    public MyException(){
        // Aufruf von Exception mit Fehlertext
        super("Ein Fehler ist aufgetreten");
    }
}
```

Fragen zum Übungsblatt?