Events / Ereignisse

Events hat oft mit grafischen Oberflächen z. B. Anklicken eines Buttons oder das Schließen eines Fensters zu tun. Dies ist aber nicht immer der Fall. Ein Event kann auch die Änderung des Preises einer Aktie sein.

Folgende Objekte sind nötig bei der Behandlung von Events:

1. Event/Ereignis:

   Das ist etwas, was passiert kann, wenn ein Programm läuft wie z. B.

   - Anwender lässt sich ein Menü anzeigen.
   - Ein Fenster wird minimiert.
   - Ein Button wird betätigt.

2. Event-Source/Ereignis-Quelle:

   Das ist das Objekt, das das Ereignis absendet.

   - Anwender klickt auf den Fenster-Minimieren-Button, dann ist der betätigte Button die Ereignis-Quelle.

3. Event-Listener/Ereignis-Hörer

   Das ist ein Objekt, welches wartet und aufpasst, ob ein Ereignis stattfindet. Merkt der Ereignis-Hörer, dass z. B. der Mimimieren-Button gedrückt wurde, dann löst dieser die gewünschte Antwort aus, also wird das Fenster minimiert.

4. Reaktion

   Das ist, was passieren wird, wenn das Ereignis/Event stattfindet.

   - Minimieren des Fensters
   - Ein Bild zeigen
   - Ein Lied abspielen

Wir schauen uns zuerst die Events bei der grafischen Programmierung.

1. Beispiel:

Gehe wir zu:
https://studyflix.de/informatik/zeichnen-in-java-521

Im Eclipse ein Projekt Events eröffnen. Die Beispiele aus 3) und 4) in Paket fensterevent implementieren.

Dort schauen wir die folgenden Videos:
1) Intro grafischen Programmierung
2) GUI-Elemente-Grundlagen
3) GUI-Elemente-Fenster und Frames (Das Beispiel implementieren!)
4) GUI-Layout (Aussage bei 3:40 beachten! Beispiel implementieren!)
5) Buttons- und Action-Events

Zu 4) die Klasse GUI wird nicht funktionieren, wie sie sollte. Nach dem fünften Video kann die GUI-Klasse mit den entsprechenden Anweisungen ergänzt werden, damit die Buttons die gewünschten Events auslösen. Machen Sie sich Notizen!

Zu 5) die Klasse ActionListener1 auch implementieren.

Ergänze die Klasse TestFrame, sodass das Fenster zu geht, wenn x angeklickt wird.

Video 6) ist zurzeit nicht relevant.

In der Klasse WindowListener kommentiere die Methode actionPerformed() aus. Schreibe eine zweite actionPerformed()-Methode, sodass der Text, der bei "DB-Suche"eingegeben wurde, nach dem Drücken der Button „Senden" auf der Konsole ausgegeben wird.

2. Beispiel:
Implementieren Sie das folgende Event-Beispiel:
Jede Klasse oder Interface hat eine eigene Datei.

https://stackoverflow.com/questions/6270132/create-a-custom-event-in-java

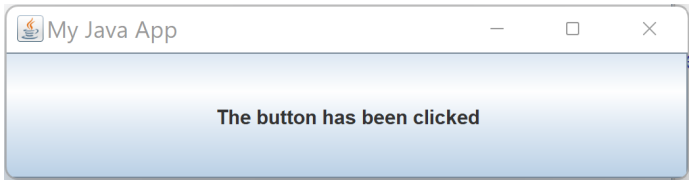
Was ist das Event?
Welches Objekt ist die Event-Source?
Welches Objekt ist der Event-Listener?
Welche ist die Reaktion?
Welche Rolle spielt das Interface HelloListener?

Erstellen Sie andere Initiater-Objekt und implementieren Sie die Events sayGoodbye und sayGoodMorning. Die Responder-Objekte sollen dem sprechend antworten.

3. Beispiel:
Wir wollen die Events üben und schauen uns die folgenden Videos und implementieren die Beispiele und testen die Klassen. Danach können wir selber Events auslösen und werden die Beispiele ergänzen.
1)
https://www.youtube.com/watch?v=JPTZATLXwbc

2)
https://www.youtube.com/watch?v=cyZzPo0ssp8

Wenn das Programm läuft, dann können wir das Programm so ändern: wenn auf den Button „Click Me“ geklickt wird, soll das folgende Fenster eröffnet werden.



Das Fenster soll sich schließen lassen.

4. Beispiel
Erzeugen wir ein Paket pricechanged und schreiben wir in diesem Paket die folgenden Klassen:

Schreibe die Klasse Stock. Die Instanzvariablen symbol und price werden über den Konstruktor die Werte zugewiesen. Für die Variable price werden auch getter- und setter-Methode geschrieben.

Testen Sie die Klasse Stock.

Wir wollen erreichen, dass wir benachrichtigt werden, wenn der Preis der Aktie sich ändert. Dabei soll die Klasse Stock nur minimal verändert werden.

Implementiere das folgende Interface:

```
public interface PriceListener extends EventListener{

    void priceChanged(PriceChangedEvent e);

}
```

Das ist das Objekt, das auf das Event aufpasst. Hier wird festgelegt, worauf der Listener hören soll, nicht aber, was er machen soll. Hier auf die Änderung des Preises lauschen.

Führen wir die nötigen Importe durch. Hier erbt das PriceListener Interface vom EventListener-Interface.

Jetzt benötigen wir eine Klasse in der die Events gehandelt werden, also die Reaktion auf das Event festgelegt wird. Implementieren wir die folgende Klasse.

```
import java.util.EventObject;

public class PriceChangedEvent extends EventObject implements PriceListener {

     private double oldPrice;
     private double newPrice;

     public PriceChangedEvent(Stock stock, double newPrice){
          super(stock);
          this.oldPrice = stock.getPreis();
          this.newPrice = newPrice;
          }

     @Override
     public void priceChanged(PriceChangedEvent e) {

          //TODO
          // Das weiter gegebene Objekt e ist ein Objekt dieser Klasse, kennt also die Instanzvariablen, z. B.
          // e.oldPrice usw.
          //Zu implementieren wären:
          //- ist der Aktienpreis ungleich, dann "Price has changed!" ausgeben
          //- ist der Aktienpreis gestiegen, dann alter Preis + " Price has gone up " + neuer Preis ausgeben
          //- ist der Aktienpreis gefallen, dann alter Preis + " Price has fallen " + neuer Preis ausgeben
     }

}
```

Die Klasse EventObject verlangt, dass die Kindsklasse einen Konstruktor mit einem Objekt haben musst. Hier ein Objekt der Klasse Stock. Mit der zweiten Variable teilen wir den neuen Preis zu dem Event Listener mit.

In der Methode princeChanged wird die Reaktion auf das Event festgelegt.

Events wie PriceChangedEvent sind Java Objekte, die Infos über Benutzeraktionen verarbeiten. Hier melden die EventListener die Events, die stattfanden, an. Hier wird festgelegt, was gemacht wird, wenn ein Event passiert ist.

Hier wird auf die Preisänderungen reagiert.

Nun müssen wir nur noch den Listener bei der Event-Quelle anmelden, also bei der Klasse Stock. Wir geben diese Anweisung bei den Variablendeklarationen an.

```
private PriceListener priceListener = new PriceChangedE-
vent(this, 0);
```

priceListener ist eine Referenz auf ein Objekt der Klasse PriceChangedEvent, weil der Konstruktor ein Objekt verlangt, geben wir das Objekt der Klasse-Stock, das wir gerade erzeugen, und einen neuen Preis von 0 weiter. Es geht darum das priceListener auf ein Objekt der Klasse PriceChangedEvent zeigt.

Den Listener haben somit wir in der Klasse Stock angemeldet. Nur die Leitung fehlt noch!

Um die Leitung zu dem Event herzustellen, geben wir die folgende Anweisung in der setPrice()-Methode, als erste Anweigung, hier wird nun mal der Preis verändert:

```
priceListener.priceChanged(new PriceChangedEvent(this,
value));
```

Die Methode priceChanged() verlangt ein Objekt der Klasse PriceChangedEvent und der Konstruktor diser Klasse verlang ein Objekt der Klasse Stock -this- und den neuen Aktienwert. Das Objekt der Klasse PriceChangedEvent wird erstellt und existiert nur so lange wie nötig.

Geben wir in der Testklasse in der Methode main() diese Anweisungen:

```
Stock stock = new Stock("AEG", 15.15);

System.out.println(stock.symbol + " hat den Preis von " + stock.getPrice()+ "€");

stock.setPrice(15.78);
stock.setPrice(15.78);
stock.setPrice(14.78);
stock.setPrice(20.78);
```

Ist das Programm fehlerfrei, dann wird die Ausgabe sein:

```
AEG hat den Preis von 15.15€
Price has changed!
15.15 Price has gone up! 15.78
Price has changed!
15.78 Price has fallen! 14.78
Price has changed!
14.78 Price has gone up! 20.78
```

Ändere diese Anweisung aus der Klasse Stock,
`private PriceListener priceListener = new PriceChangedEvent(this, 0);`
in `private PriceListener priceListener;`
und test das Programm. Was stellen wir fest?

Jetzt soll das folgende Event implementiert werden:
Steigt der Aktienpreis um mehr als 10%, dann soll die Warnung `"Preissteigerung um mehr als 10%!"` ausgegeben werden. Name des Events `priceChangedMorethanTenPercent`

Ergänze weitere Events zur Eventquelle Stock!

Übung 1) Implementieren Sie das Event, in dem Darth-Vader zu Luke Skywalker gesteht, dass er, Darth Vader, Lukes Vater ist. Nicht vergessen, das Luke laut Nein geschrien hat, als er dies hörte.

Übung 2) Implementiern Sie einen Guthabentaschenrechner, wenn das Event „Division“ ausgelöst wird, dann soll das Programm mit der Reaktion „Achtung nicht duch Null dividieren“ auslösen. Ist das Ergebnis größer als 2000, dann wird auf dieses Event mit „Sie haben nicht soviel Geld im Konto!“ reagiert.

Zum Vertiefen:

http://dev.cs.ovgu.de/java/Books/javainsel3/javainsel_150000.htm

Nur 15.1 lesen.

Übung 3) Implementiere das Event „Maximale Geschwindigkeit erreicht“ bei der Ventilator-Aufgabe.