Fragen, Antworten und Kommentare zur aktuellen Vorlesung

Bitte nehmen Sie an der anonymen Lehrevaluation unter https://forms.gle/jev7foCtLFXTYrq66 bis zum 5.6 teil. Die von mir kommentierten Ergebnisse stehen im letzten Fragen&Antworten-Dokument des Semesters.

Frage: Sollte Swing noch in neuen Projekten genutzt werden?

Antwort: Schwierige Frage. Generell wird Swing nicht weiterentwickelt, ist aber auf einem sehr stabilen Niveau, trotz einiger dann zu ignorierenden Exceptions bei komplexeren nebenläufigen Prozessen. Also Swing ist einsetzbar und recht mächtig. Als graphisch deutlich anspruchsvollere Alternative wurde JavaFX entwickelt, ist aber nicht mehr Teil der Java-Standardedition. Die Weiterentwicklung erfolgt unter OpenJFX und funktioniert ebenfalls problemlos. ZU beachten ist aber, dass OpenJFX nicht ganz einfach nachinstalliert werden kann, da es anders, als bei anderen Bibliotheken nicht ausreicht einfach weitere Jar-Dateien zur Verfügung zu stellen. Das ist auch der Grund, warum in der KleukerSEU eine Java-Version mit integrierter JavaFX (= OpenJFX)-Unterstützung. Java-Swing soll auch ab 2027 (ß) nicht mehr Teil der Standard-Version sein, wird dann aber ähnlich zu OpenJFX zur Verfügung stehen. Zusammengefasst, soll es eine besonders positiv auffallende Oberfläche sein, ist von Java Swing abzuraten, für technische Fenster zur Ein- und Ausgabe ist es aber problemlos weiterhin nutzbar. Es gibt einige „nette“ graphische Spiele, die in denen diese Te3chnologieren genutzt werden.

Nebenbei, das Spiel Minecraft verwendet die Lightweight Java Game Library (LWJGL) als Oberflächentechnologie. Diese Java-Bibliothek bietet eine plattformunabhängige Implementierung von OpenGL, OpenAL und OpenCL, was es ermöglicht, Grafiken, Sound und andere Funktionen über Java anzusteuern.

Frage: Ich hab das mit der Nutzung von GUI-Elementen nicht verstanden.

Antwort: Der Ansatz basiert auf dem Observer-Observable (oder Producer-Consumer)-Ansatz, der in OOAD genauer vorgestellt wird. In der Basisform hat jeder Knopf eine Liste von Objekten, die sich für Aktionen des Knopfes interessieren. Die Klassen dieser Objekte realisieren das Interface `ActionListener`. Jeder Knopf, genauer eine Verwaltungsklasse dahinter, hat damit eine Liste von Interessenten vom Typ `ActionListener`. Wird dann eine Aktion mit dem Knopf durchgeführt, wird jeder Interessent darüber informiert. Dazu wird die Methode `public void actionPerformed(ActionEvent e)` aufgerufen, die als Parameter ein Objekt übergibt, das Details über die Aktion, z. B. welche Maustaste wurde genutzt, enthält.

Der folgende Code erzeugt eine Minioberfläche mit einem Knopf und zeigt zwei von vielen Möglichkeiten Objekte zu erzeugen, die das Interface ActionListener realisieren.

```
package main;

import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
```

```java
public class Main {

  public static void main(String[] args) {
    JFrame frame = new JFrame("Titel");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(300, 300);
    frame.setLayout(new GridLayout());

    JButton button = new JButton("Klick");
    ActionListener listener = new ActionListener() {
      @Override
      public void actionPerformed(ActionEvent e) {
        System.out.println("Action1");
      }
    };
    button.addActionListener(listener); /*hier*/

    button.addActionListener(ev -> System.out.println("Action2")); /*hier*/

    frame.add(button);
    frame.setVisible(true);
  }
}
```

Hier melden sich zwei Interessenten (`/*hier*/`) bei dem Knopf an.

Die nachfolgende Abbildung zeigt die Oberfläche, nachdem dreimal der Knopf gedrückt wurde.