Fragen, Antworten, Kommentare zur aktuellen Vorlesung

Die Online-Befragung zur genutzten alternativen Veranstaltungsform und zur Lehrevaluation ist online. Bitte ausfüllen: https://forms.gle/MLCEGPeBxFZMGbcj9. Sie werden eventuell aufgefordert sich bei Google anzumelden, das ist nur notwendig, wenn Sie in der Bearbeitung eine Pause machen wollen und das Teilergebnis zwischenspeichern wollen. Die Befragung endet am 22.12., die Ergebnisse stehen in einem nachfolgenden Fragen&Antworten-Dokument auf der Webseite der Veranstaltung.

Frage: Kann man eigentlich Spiele mit dem Interaktionsbrett schreiben?

Antwort: Generell ja, da Sie jetzt die Tastatur- und Maussteuerung kennen sollten. Bei komplexeren Spielen kann die Problematik bestehen, dass durch das notwendige Abwischen des Interaktionsbretts das Bild anfängt zu flackern. Zwei Minispiele, die ursprünglich mit dem Interaktionsbrett entstanden sind, sind online und sollten von Ihnen nachprogrammiert werden können.

Frage: Bei mir scheitert ein Test und die Ausgabe ist „expected true but was false“. mein Ergebnis ist aber true, was stimmt das nicht?

Antwort: Sie haben recht, die Ausgabe ist irritierend. Der Test findet einen Fehler, allerdings wäre die korrekte Ausgabe „expected false but was true“. Der Hintergrund ist, dass die Methode assertTrue(x) genutzt wird und wenn der Wert von x nicht true ist erzeugt JUnit automatisch die Ausgabe „expected true but was false“. Einige meiner Tests prüfen aber assertTrue(!x), also Negation, da macht die Ausgabe keinen Sinn. Die Fehler sollten aber bei der ergänzenden Ausgabe brauchbar erklärt sein. Der Hintergrund ist, dass ich möglichst wenig Methoden nutzen wollte und so auf assertFalse verzichtet habe. Das kleine Beispiel zeigt die Problematik, die folgende Klasse soll eine Methode haben, die false als Ergebnis liefert.

```
public class MachFalsch {
  public boolean falsch() {
    return true;  // falsche Methode
  }
}
```

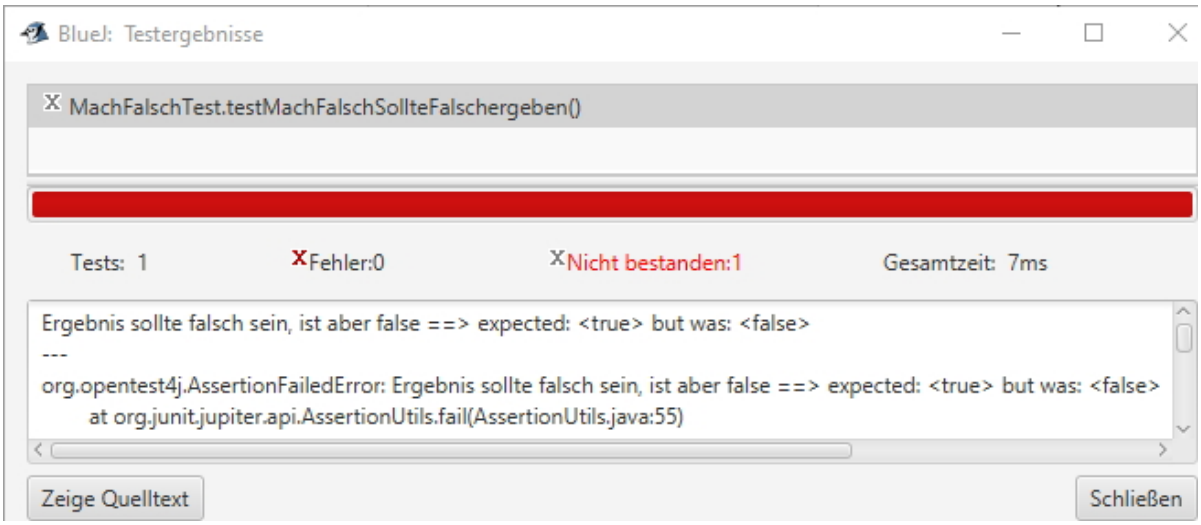
Die zugehörige Testklasse sieht wie folgt aus:

```
import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.Test;

public class MachFalschTest {

  @Test
  public void testMachFalschSollteFalschergeben() {
    MachFalsch zuTesten = new MachFalsch();
    Assertions.assertTrue(!zuTesten.falsch()  // Negation beachten
      , "Ergebnis sollte falsch sein, ist aber " + !zuTesten.falsch());
  }
}
```

Die Ausgabe zeigt die skizzierte Problematik, die Ausgabe nach „==>“ stammt von JUnit.



Wenn Sie das zu sehr irritiert, ersetzen Sie „assertTrue(!“ durch „assertFalse(“, dann stimmt auch die generierte Ausgabe.

Frage: gibt es eine Möglichkeit systematisch zu analysieren, auf welche Nullwerte geachtet werden muss?

Antwort: Grundsätzlich sich ist anzunehmen, dass alle übergebenen Objekte, also nicht elementare Datentypen, null sein können. Dies kann ebenfalls auf alle Objekte zutreffen die in Objektvariablen stehen. Daraus kann eine Boolesche Matrix gebaut und dann für einzelne Fälle, meist Gruppen von fällen entschieden werden.

Beispiel: Klasse Messreihe;

```
public class Messreihe {
  private String messort;
  private ArrayList<Integer> messwerte;

  public boolean gleicheWerte(Messreihe m) {
        // hier soll in diesem Fall fuer das Ergebnis null
        // wie eine leere Liste behandelt werden
```

In der Methode können this.messwerte, this.messort, , other, other.messwerte und other.messort null sein. Da der messort in der Aufgabe nicht interessiert, reduziert es sich auf folgende Fälle:

this.messwerte	other	other.messwerte	Beispielergebnis
null	null	null	true
~~null~~	~~null~~	~~not null~~	ist nicht möglich, wenn other null ist
null	not null	null	true
null	not null	not.null	Ergebnis abhängig von other.messreihe.size()
not null	null	null	Ergebnis abhängig von this.messreihe.size()
~~not null~~	~~null~~	~~not null~~	ist nicht möglich, wenn other null ist
not null	not null	null	Ergebnis abhängig von this.messreihe.size()
not null	not null	not null	Prüfalgorithmus ausführen

eine von vielen Umsetzungsmöglichkeiten ist:

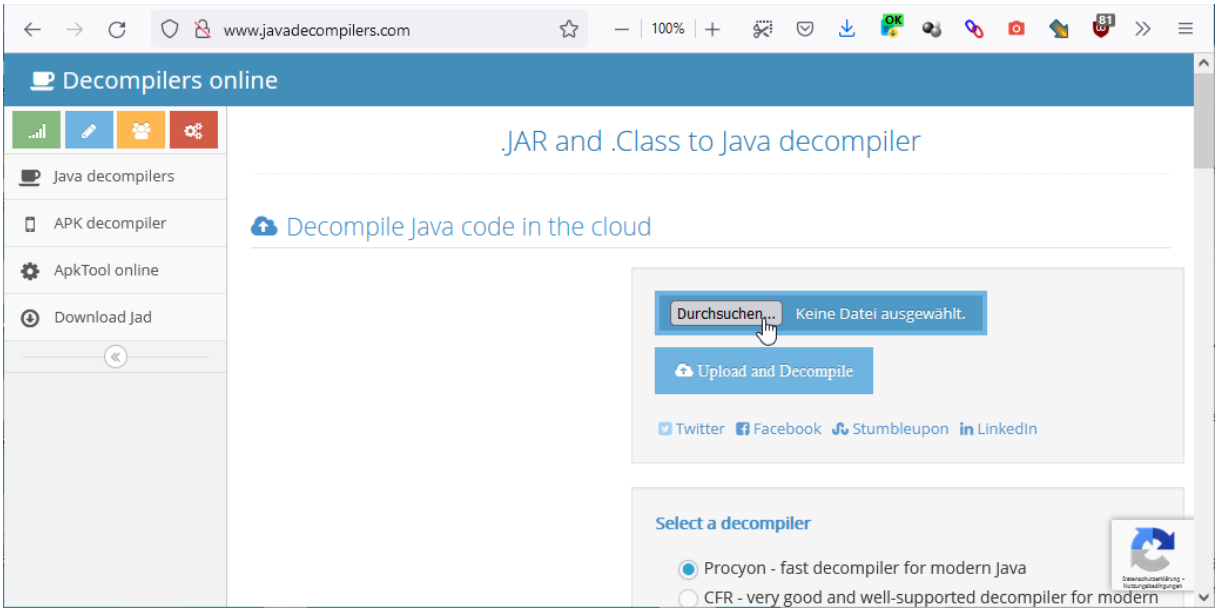
```
if (this.messwerte == null || this.messwerte.isEmpty()) {
  return other == null || other.getMesswerte() == null
      || other.getMesswerte().isEmpty();
}
if (other == null || other.getMesswerte() == null) {
  return false;
}
// “eigentliches” Programm
```

Es gibt auch Ansätze durch die umgebende Programmierung zu verhindern, dass null-Werte vorkommen können. Dazu muss aber auf einfache set-Methoden verzichtet werden, was das Testen erschwert.

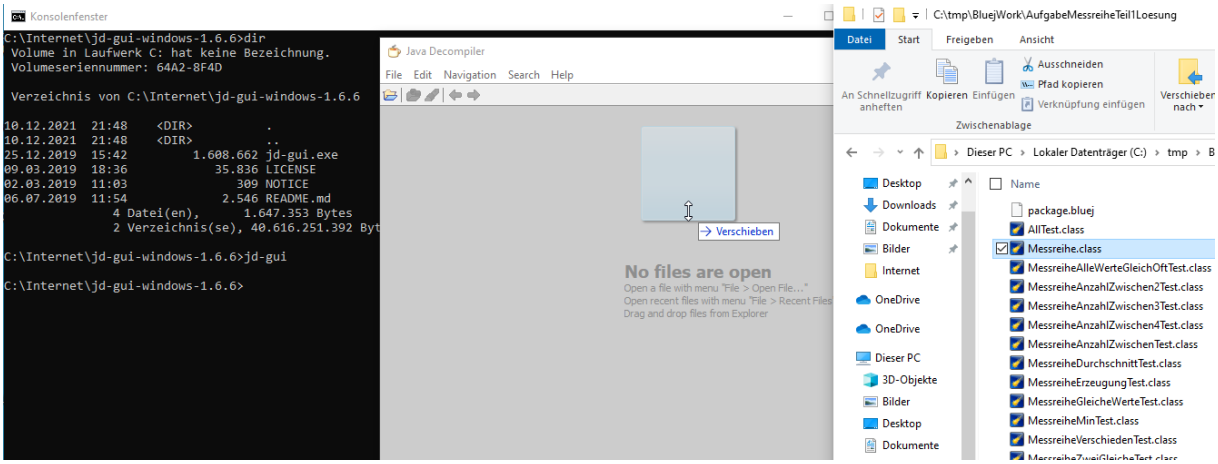
Frage: BlueJ ist abgestürzt und anscheinend ist meine Java-Datei weg, was kann ich machen?

Zunächst einmal die Reste des Projekts sichern und einfach eine Kopie davon erstellen, um die später beschriebenen Analysen machen zu können. Weiterhin sei angemerkt, dass ich zwar Abstürze kenne, BlueJ aber den aktuellen Code immer auf der Festplatte hält, so dass verschwundener Code sehr sehr selten auftreten sollte und Abstürze von BlueJ übersteht. Schauen Sie, ob sich im Ordner noch eine „.java“-Datei befindet und kopieren Sie diese in ein neues BlueJ-Projekt.

Java-Klassen werden generell in Bye-Code übersetzt/compiliert. Dieser Byte-Code kann auf allen unterstützen Betriebssystemen laufen und steht in einer Datei mit Endung „.class“. Da wir keine besonderen Tricks anwenden, kann dieser Byte-Code bis zu einem gewissen Grad zurück in Java-Code übersetzt werden, so dass der Einsatz eines sogenannten Decompilers sinnvoll sein kann. Die einfachste Möglichkeit bietet http://www.javadecompilers.com/ mit der Online verschiedene Decomplier ausprobiert werden können. Nach einem Klick auf „Durchsuchen…“ wird die .class-Datei ausgewählt und auf „Upload and Decompile“ geklickt.



Wenn Sie lieber offline arbeiten: Laden Sie sich jd-gui-windows-1.6.6.zip von http://java-decompiler.github.io/ herunter und packen Sie es in einem Ordner XY aus. Nutzen Sie von der kleukersSEU StartKonsole.bat, um ein Konsolen-Fenster mit installiertem Java zu erhalten. Steuern Sie in den Ordner XY und rufen Sie „jd-gui“ auf. Ziehen Sie dann die .class-Datei in das sich öffnende Fenster. Mit etwas Glück, ist der Code zu sehen.



Generell sind Datenverluste immer mit einzukalkulieren. Um diese Gefahr zu minimieren sind Backups zu machen. Bei einer lokalen Entwicklung in BlueJ kann in BlueJ einfach eine Kopie des Projekts erstellt werden. Genauso ist dies einfach mit dem Projektordner und Crtl+C und Ctrl-V in Windows möglich. Nebenbei angemerkt: Bei formalen Abgaben (Hausarbeiten, Projektberichte, Bachelorarbeiten) ist nebenbei ein Rechner- oder Festplattenausfall kein Grund für eine verspätete Abgabe. Es wird davon ausgegangen, dass Sie innerhalb von maximal 24 Stunden wieder arbeitsfähig sind und diesen Zeitpuffer eingeplant haben. Bei Praktikumsaufgaben kann man das etwas lockerer sehen.

In der nächsten vorlesungsfreien Zeit sollen Sie sich mit dem Thema Git und auch GitLab bzw. GitHub beschäftigen. Dabei handelt es sich um ein Versionsverwaltungssystem, das auch die Zusammenarbeit mehrerer Entwickler ermöglicht.

Nebenbei, wenn Sie es schaffen eine Absturzsituation zu wiederholen, bin ich sehr interessiert daran Berichte mit Informationen über die verwendete Installation zu erhalten.