Übungen zu „Einführung in Software Engineering“ (WS 15/16)
Prof. Dr. Barbara Paech, Marcus Seiler
http://se.ifi.uni-heidelberg.de

Arbeitsblatt 1 (13.10.2014)
Eclipse

In dieser Übung lernen Sie die Grundfunktionen der Entwicklungsumgebung Eclipse kennen:

- ✓ Lernen Sie das Anlegen von Projekten und das Wechseln von Perspektiven kennen.
- ✓ Lernen Sie das Anlegen von Paketen und Klassen kennen.
- ✓ Lernen Sie das Ausführen von Java-Anwendungen kennen.
- ✓ Lernen Sie die Verwendung des Eclipse Debuggers kennen

Aufgabe 1: Eclipse starten, Projekt anlegen, Perspektive wechseln

1. **Eclipse starten**
   - Starten Sie die Entwicklungsumgebung Eclipse über das Terminal mit folgendem Befehl: `eclipse&` (gilt für den CIP-Pool in INF350)
   - Alternativ können Sie Eclipse z.B. unter Windows über die Anwendung `eclipse.exe` im Eclipse-Ordner starten.
   - Wählen Sie nach dem Start den Ort Ihres Workspaces aus. In einem Eclipse-Workspace werden alle Ihre erstellten Projekte und Dateien gespeichert.

2. **Projekte und Perspektiven**
   - Erstellen Sie ein erstes Projekt vom Typ „Project“.
     - Menu → File → Project ...
     - Wählen Sie im „New Project“ Wizard unter „General“ den Typ „Project“ aus.
     - Geben Sie dem Projekt einen Namen, z.B. `foo`
   - Wechseln Sie in die „Resource“ Perspektive.
     - Menu → Window → Open Perspective → Other ...
     - Wählen Sie „Resource“ als Perspektive aus.
   - Leere Datei anlegen
     - Rechtsklick auf Project → New → File
     - Geben Sie der File einen Namen, z.B. `foo.txt`
     - Schreiben Sie einige Zeichen in die Datei und speichern Sie die Änderung.
   - Erstes Projekt löschen
     - Rechtsklick auf Project → Delete
     - Wählen Sie im Dialog „Delete Resources“ die Option „Delete project contents on disk“ aus.

3. **Java Projekt anlegen**
   - Erstellen Sie ein zweites Projekt vom Typ „Java Project“.
     - Menu → File → Project ...
     - Wählen Sie im „New Project“ Wizard unter „Java“ den Typ „Java Project“ aus.
     - Geben Sie dem Projekt einen Namen, z.B. `FooProject`.
     - Wählen Sie die Option „Create seperate folders for sources and class files“.
     - Klicken Sie „Next“ und bestätigen Sie mit „Finish“.

4. **Perspektiven wechseln**
   - Wechseln in „Java (default)“ Perspektive
     - Beim Anlegen eines Java Projects (siehe 3. Aufgabe oben) erscheint ein Dialog, der Sie auffordert, in die geeignete Perspektive zu wechseln.
     - Alternativ können Sie über folgenden Befehl die Perspektive „Java (default)“ öffnen:
       - Menu → Window → Open Perspective → Other ...
       - Wählen Sie „Java (default)“ als Perspektive aus.
   - Wechseln in „Java Browsing“ Perspektive und zurück.
     - Mit der Perspective „Java Browsing“ können Sie sich die in einem Projekt enthaltenen Packages, Types und Members anzeigen lassen.
     - Wechseln Sie in die „Java Browsing“ Perspektive
       - Menu → Window → Open Perspective → Other ...
       - Wählen Sie „Java Browsing“ als Perspektive aus
     - Machen Sie sich mit dieser Perspektive vertraut.
     - Wechseln Sie am Ende zurück zur „Java (default)“ Perspektive.

Aufgabe 2: Paket anlegen, Refactoring, Klasse anlegen

1. **Paket anlegen**
   - Erstellen Sie in Ihrem Java Projekt aus Aufgabe 1 ein Package.
     - Rechtsklick auf Projekt → New → Package
     - Geben Sie dem Package den Namen `de.unihd.isw.foo`
     - Das Package ist danach leer (erkennbar am weißen Icon).

2. **Refactoring**
   - Benennen Sie das angelegtes Paket um
     - Rechtsklick auf Package → Refactor → Rename
     - Geben Sie einen neuen Namen ein, z.B. `de.unihd.isw.helloworld`

3. **Neue Klasse „HelloWorld“ anlegen in Paket**
   - Erstellen Sie eine neue Klasse mit Hilfe des Class Wizard.
     - Rechtsklick auf Package → New → Class
     - Geben Sie der neuen Klassen den Namen `HelloWorld`
     - Wählen Sie unter „Which method stubs would you like to create?“ als zusätzliche Option „public static void main(String[] args). Dadurch wird gleich der Methodenrumpf mitgeneriert, der die Java-Klasse ausführbar macht.
   - Verwenden Sie den Editor um HelloWorld zu programmieren.
     - Fügen Sie in der main-Methode folgende Zeile Java-Code ein:
       - `System.out.println("Hello world!");`

4. **Applikation starten**
   - Starten Sie HelloWorld.
     - Rechtsklick → Run As → Java Application
     - Hinweis: Nur Java-Klassen, die eine main-Methode haben, lassen sich über „Run As“ ausführen!
     - Es sollte sich eine neue View „Console“ öffnen und den Text „`Hello world!`“ anzeigen.

5. **Applikation debuggen**
   - Erstellen Sie mehrere Zeilen zur Ausgabe (System.out.println) oder Variablendefinitionen (z.B. `int x = 42`) oder Methoden in der HelloWorld-Klasse, je nach Ihren Kenntnissen.
   - Setzen Sie einen Breakpoint.
     - Doppelklicken Sie in der linken Scrollbar vor eine Zeile, z.B. `int x = 42`. Es sollte ein blauer Punkt erscheinen.
     - Durch Setzen eines Breakpoints wird beim Ausführen im Debug-Modus die Java-Anwendung angehalten und Sie können sie „debuggen“.
   - Starten Sie HelloWorld im Debug-Modus.
     - Rechtsklick → Debug As → Java Application
     - Es sollte ein Dialog erscheinen, der Sie auffordert, in die Debug Perspektive zu wechseln.
     - Mit Step Into (F5) oder Step Over (F6) können Sie im Debug-Modus navigieren, z.B. durch wiederholtes Drücken von Step Over (F6) können Sie den Aufruf unterschiedlicher Java-Methoden verfolgen.

Aufgabe 3: Anwendung „Quadrat“

Gegeben sei folgendes Java-Programm „Quadrat“:

```
1  package de.unihd.isw.quadrat;
2
3  public class Quadrat {
4
5        static int quadrat(int n) {
6              return n * n;
7        }
8
9        static void ausgabe(int n){
10             String s;
11             int i;
12
13             for (i = 1; i <= n; i=i+1){
14                   s = "Quadrat(" + i + ") = " + quadrat(i);
15                   System.out.println(s);
16             }
17       }
18
19       public static void main( String args[])
20       {
21             ausgabe(4);
22       }
23 }
24
```

1. Programmieren Sie das Programm in Eclipse.
2. Führen Sie das Programm in Eclipse aus.
3. Debuggen Sie das Programm in Eclipse.
4. Verändern Sie das gegebene Programm nach Belieben und führen Sie es erneut aus.

Aufgabe 4: Anwendung „Quadrat“ debuggen

1. Debugger Starten
Debuggen dient der detaillierten Analyse des Laufzeitverhaltens von Programmen. Den Debugger für Java Programme in Eclipse starten sie durch:

1. Rechtsklick im Java Editor der Datei *Quadrat.java*
2. Wählen sie dann im Kontextmenu *„Debug as Java Application“*

2. Breakpoints Festlegen
Damit Sie eine Anwendung debuggen können müssen Sie zunächst Haltepunkte (engl. Breakpoints) für den Programmfluss festlegen. Einen Breakpoint legen sie fest in dem Sie:

1. Am linken Rand des Java Editors der Datei *Quadrat.java* Doppelklicken
2. Es erscheint ein blau ausgefüllter Kreis welcher die Stelle des Breakpoints anzeigt

3. Debug Perspektive
Legen Sie nun für das Programm *Quadrat.java* in Zeile 13 und 15 jeweils eine Breakpoint fest.
Starten sie das Programm über das Kontextmenu mit *„Debug as Java Application“*
Es erscheint eine Abfrage in welcher Sie gefragt werden ob die *„Debug perspective“* geöffnet werden bestätigen Sie diese mit ja
Die *„Debug perspective“* wird geöffnet in welcher Ihnen folgende Ansichten (Views zur Verfügung stehen)

1. Debug – zum Steuern des Programmablaufs wären einer Debug Sitzung
2. Variables – zeigt die aktuellen Werte von Variablen; Breakpoints – zeigt eine Übersicht der Eclipse bekannten Breakpoints
3. Java Editor – Zeigt den aktuell Ausgeführten Java Code einschließlich einer Hervorhebung der Stelle an welcher sich der Programmfluss gerade befindet
4. Console – Standard Ein- und Ausgabe des Java Programms

4. Debugging durchführen und steuern
Das Steuern des Programmflusses ist über die *Debug View* möglich in der Toolbar der View oben rechts haben sie unterschiedliche Möglichkeiten um den Programmfluss gezielt zu steuern. Nachfolgende die wichtigsten dieser Möglichkeiten:

1. Resume (F8) – Setzt das Programm fort bis der nächste Breakpoint oder das Programm ende erreicht wird
2. Suspend/ Terminate – Pausieren/ Beenden der aktuellen Programm Instanz
3. Step-Into (F5) – Methodenaufruf nachverfolgen
4. Step-over (F6) – Methodenaufruf oder Zuweisung überspringen
5. Step-return (F7) – eine Schritt zurück im Programmablauf

Nach dem Setzten der Breakpoints in Zeile 13 und 15 von *Quadrat.java* und dem Starten der Anwendung im Debug Mode können Sie im Java Editor Fenster die Hervorhebung von Zeile 13 für den aktuellen Stand des Programmablaufes sehen. Nutzen Sie die Funktion *„Resume“* um den Programmablauf zum Breakpoint in Zeile 15 weiter laufen zu lassen. Betrachten sie dabei die Veränderungen in der *Varibales View*. Testen sie nach Beendigung und dem erneuten Start des Programms im Debug Modus auch die anderen Funktionen des Debuggers zum Steuern des Programmflusses.