TUT-Anleitung: Eclipse-Repository für die Korrektur

Allgemeines

Es gibt einen (virtuellen) Rechner *193.196.54.237* ~~wilma.informatik.hs-mannheim.de~~, im Folgenden *wilma* genannt, mit einem Git-Repository pro Studenten-Zweier-/Dreiergruppe und einem gemeinsamen Repository für die Tutoren (Lösungen!). ~~Der Rechner ist nur im Hochschulnetz oder mittels VPN erreichbar.~~

Die Studenten bekommen eine Anleitung, wie sie ein Repository mit einem existierenden PR1-Projekt in Eclipse einbinden.
Sie können das Repo zum Dateiaustausch nutzen (Gruppenpartner, zuhause, Laptop etc.). Ein Push (nicht nur Commit) schiebt die Daten auf *wilma*; darüber würde dann der Austausch erfolgen.
Nach jedem Push bekommen alle Gruppenmitglieder eine E-Mail (*Git-Post-Commit-Hook* und *Jenkins*) und eine Nachricht in ihren Discord-Text-Channel:

- Lassen sich die Studi-Dateien übersetzen?
- Laufen initiale JUnit-Tests?
  Diese initialen Tests entsprechen etwa den gedruckten Beispielen auf den Übungsblättern.

~~Einmal pro Woche (mittwochs 10:00 Uhr) verschickt der *Jenkins* die dann aktuelle Version eines jeden Repositories auf *wilma* per E-Mail an denjenigen Tutor, der in dieser Woche die Gruppe betreut.~~

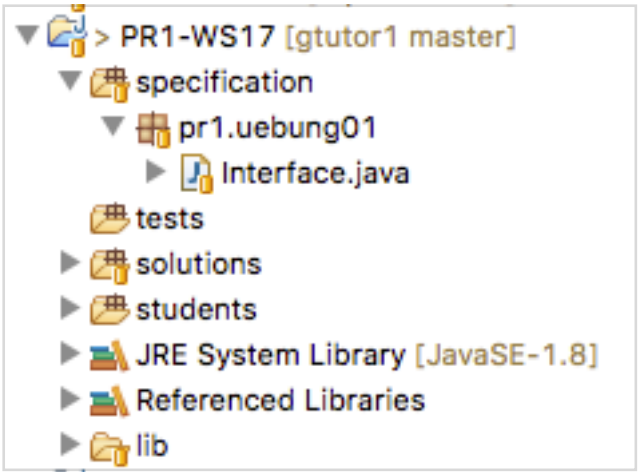
Alle Tutoren teilen sich ein Repository (Commits bitte **nur durch mich**!), in dem es die jeweils aktuellen Musterlösungen zusammen mit JUnit-Tests für die Lösungen der Studenten gibt. Jeder Tutor bekommt eine E-Mail mit der Repository-URL, seinem Account-Namen nachdem er ein Passwort dafür gewählt hat.

Einbinden in Eclipse

- Repository-View öffnen
- Clone Repository:
  - die Daten aus der E-Mail eingeben
  - Eclipse-Projekt gleich mit erstellen lassen

~~Vorschlag für die Korrektur der abgegebenen Übungen jede Woche~~

- Update des Tutoren-Repositories



- Der *source folder* "specifications" enthält
  - eventuelle Vorgaben (Interfaces, Teil-Implementierungen, Hilfsklassen) für die verschiedenen Aufgaben der Übungsblätter
  - JUnit-Testklassen "…1stTest", die dafür gedacht sind, die geforderten Methoden mit den einfachsten Vorgaben zu überprüfen; diese Testklassen sind für die Studenten auch im Wiki verfügbar
- Der *source folder* "tests" enthält "richtige" JUnit-Testklassen, welche die Lösungen der Studenten sehr intensiv überprüfen. Problem:
  - Die Erfahrung zeigt, dass es fast nie korrekte studentische Lösungen gibt.
  - Da die Tests nur auf Symptome prüfen, führt ein einzelner Fehler oft zu vielen fehlschlagenden JUnit-Tests!
- Der *source folder* "solutions" enthält eine Musterlösung für die verschiedenen Aufgaben der Übungsblätter.

ab hier im WS 2020 nicht relevant!

- Der *source folder* "students" ist dafür vorgesehen, die Lösungen der Studenten, die per E-Mail an uns Tutoren verschickt werden, aufzunehmen, solange wir die Tests laufen lassen, um die Abgabe zu bewerten.
  Dazu sollte Folgendes passieren:
  - Bei Bedarf Musterlösung ansehen.
  - Den *source folder* "solutions" aus dem *build path* entfernen (sonst gibt es doppelte Klassen im Projekt).
  - Die Lösung der Gruppen in das passende Package im *source folder* "students" einfügen.
    (Die "abgegebenen" Lösungen der Gruppen wurden per E-Mail an den zugeordneten Tutor verschickt.)
  - Tests für das entsprechende Package im *source folder* "test" laufen lassen, um nicht alle eventuellen Fehler im Listing suchen zu müssen.
- Listings ansehen:

- Ist das Programm verständlich? Gibt es sinnvolle Kommentare, keine unsinnigen Kommentare, sprechende Namen etc.?
    - Indikator: Findet man Stellen, an denen man aufgrund der JUnit-Tests Fehler vermutet, einfach oder muss man lange suchen?
- Werden nur die erlaubten Konstrukte verwendet (z.B Schleifen, die in der Vorlesung behandelt wurden; keine Bibliotheken, welche die Lösung unverhältnismäßig vereinfachen etc.)?
    - Nicht-behandelte Programmkonstrukte sollten *alle Gruppenpartner im Testat gut erklären* können (z.B. Schleifen in andere Schleifen umwandeln); können sie das, gibt es keinen Punktabzug.
    - Die verwendeten Teile unerlaubter Bibliotheken (z.B. Verwendung von Integer.parseInt, wenn eigentlich selbst eine Umwandlungsmethode geschrieben werden soll) müssten ebenfalls durch eigenen Code ersetzt werden können; das ist aber oft im Rahmen der Übung/des Testats nicht möglich! ➜ individuelle Punkte-Entscheidung treffen
- Sind Java-Konventionen (z.B. Groß-/Kleinschreibung, Formatierung) eingehalten?
- **Der (gründliche) Review von Listings ist essenziell! Die JUnit-Tests sollen dabei lediglich entlasten und ersetzen nicht den manuellen Review!**

Bitte merken Sie sich eventuelle Verbesserungsvorschläge für den Ablauf, Probleme etc. und sagen Sie mir Bescheid!

Stand 10/20: Das Projekt im Tutor-Repository ist mit Eclipse 2020-09 erstellt ~~(keine neuere Version im bwLehrpool)~~.

#PR1