Exercises 1: Tool installation and JML Basics

TECHNISCHE UNIVERSITÄT DARMSTADT

The solutions to the exercises will be discussed on Thursday, 23rd April.

Problem 1 Tool installation: JML Editing, KeY4Eclipse Starter and more

To install the necessary tools, please follow the instructions below:

- Install Eclipse Luna SR2 (4.4.2) available at

  http://www.eclipse.org/downloads/

  Either of the variants *Eclipse IDE for Java Developers* or *Eclipse Modeling Tools* can be used.

- Next we need to install a few additional extensions:
  - Find the menu entry *Install New Software* in menu *Help* (exact location may be OS specific)
  - In the upcoming dialog enter in text field labeled *Work with:* the URL

    http://www.key-project.org/download/fsav/luna/

    and press the *Enter* key.
  - After a short moment the table below the text field should display five entries: Debugging, Editing, Utilities, Verification and Visualization. Either select all of them or at least *Editing > JML Editing Feature* and *KeY 4 Eclipse Starter Feature*. The feature

    JML Editing offers syntax checking and highlighting of JML expressions along with keyword completion and more. An overview can be found at: http://www.key-project.org/eclipse/JMLEditing/index.html

    KeY 4 Eclipse Starter allows to start the verification system KeY from within Eclipse, which we will use later to verify our specified programs.

Problem 2 Importing the Eclipse projects for the Exercises

When downloading this exercise the archive contained a directory called `eclipse`. Import the contained project into your Eclipse installation. Use the Import-Wizard at *File > Import ...* and select *General > Existing Projects into Workspace*. Follow the instructions of the wizard.

Problem 3 Identifying Specification Cases

How many specification cases for `System.arraycopy` do you count when reading its API documentation at

http://tinyurl.com/system-arraycopy (link redirects to Oracle Java 8 API documentation[1])

- Write a short unit test for some (your choice) of the exceptional cases (those that throw an exception).

- Write a normal behavior JML method specifications for the normal terminating cases incl. assignable clauses. As a simplification assume that the signature of `System.arraycopy` is `System.arraycopy(int[], int, int[], int, int)`. This avoids technical complications like the need to exclude `ArrayStoreExceptions` in the precondition and lots of casts in the ensures clauses.
  The imported Eclipse project from Problem 2 contains a class `System` in package `problem3` with the method declaration of `arraycopy`. You can attach your JML specification there.

[1] long link: http://docs.oracle.com/javase/8/docs/api/java/lang/System.html#arraycopy-java.lang.Object-int-java.lang.Object-int-int-

Problem 4 JML specification for `HealthTracker.addCategory(Category)`

The Eclipse project from Problem 2 contains the class `Healthtracker` in package `problem4` containing the source code with the JML specifications for `HealthTracker.addCategory(Category)`.

Clean up the specification by e.g.,

- using class level specifications
- removal of redundant specifications
- usage of queries

Attention: For this exercise you need some material from the upcoming lecture on Monday, 20th April.

Problem 5 Specification vs. Implementation

The Eclipse project from Problem 2 contains the file `ProblemCollection.java` in package `problem5` with some JML specified classes and methods. Which of the methods adhere to their specification? Assume JML's visible state semantics and not the one of JML*. Justify your answer.