General Preparation for 2IPC0

Links, downloads, and other details can be found via the course web page:

`http://canvas.tue.nl/courses/473`

1. Make sure you have registered for 2IPC0 at `oase.tue.nl`. Once you have done that, you will also be registered for 2IPC0 in Canvas.
2. Obtain a copy of the book *Programming in the Large with Design Patterns*, available as inexpensive print book or ebook, e.g. via `amazon.de`. This serves as the reference text for design patterns in 2IPC0. There is also a free PDF on Canvas.
3. Obtain a (free digital) copy of the book *Introduction to Programming Using Java*. You can use it to brush up your Java knowledge.
4. Download and read the following handouts:
   - *Course Outline*
   - *Coding Standard for 2IPC0*
   - *Checklist for larger OO programs* (not everything is relevant from the beginning)
   - *Momotor Feedback*
   - *Configuration Management*
   - *Notation*
   - *Specification*
5. Make sure you have installed *Java 8 JDK* (should be installed on your laptop by default).
6. Install *NetBeans 8.2*, available from `https://netbeans.org/downloads/`. The Java SE edition suffices.

   Configure the NetBeans editor to *not* "Indent Case Statements in Switch":
   - Preferences > Editor > Formatting
   - Select Language: Java
   - Uncheck Indent Case Statements in Switch
7. Install *Mercurial*, see `https://www.mercurial-scm.org/`
8. (*Optional*) Install *DrJava*, a Java IDE with Interactions, and used for the 2IP90 course. Set the indent level of your editor to 4 (Edit > Preferences > Miscellaneous), so that TAB characters are expanded.

9. Install a local copy of the *Java Platform Standard Edition 8 Documentation*. See `http://wiki.netbeans.org/FaqJavaDoc`.

10. Download `PrePostDoclet.jar` from Canvas, and store it in an easily accessible place.

    For each project, configure its properties:

    - Project Properties > Build > Documenting
    - Check @author and @version
    - Fill in Additional Javadoc Options:
      `-docletpath <dp> -doclet nl.tue.doclets.PrePostDoclet`
      where `<dp>` is the absolute path to the jar-file (including `PrePostDoclet.jar`). If that path contains spaces, then put quotes around the path.

    This makes `javadoc` recognize some additional tags used in 2IPC0.

    (N.B. We try to include `PrePostDoclet.jar` in NetBeans projects that we provide.)

    You can also configure *DrJava* to use it.

Assignments

1. (*Mercurial Demo*, Compulsory) For the description, see Canvas.

2. (*Candy TDD with Mercurial*, Compulsory) For the description, see Canvas.

3. (*Secure Key Collection*, Compulsory) Given are a description, contract, and implementation of a method with signature

   ```
   public boolean isSecure(int[][][] keys)
   ```

   that determines whether a collection of lock keys is secure. This method contains a defect and is too complex. Improve the implementation by applying functional decomposition to spread it out over at least three methods, with reduced control nesting.

   The top-level method may contain the outer two nested loops, which should be turned into *for-each* loops. Also provide (possibly informal) contracts as javadoc for the auxiliary methods. These methods *need not* be robust.

   Provide some test cases for the auxiliary methods of your decomposition.

   Given are the files:

   - `AbstractKeyCollection.java`
   - `KeyCollectionMonolithic.java` (with defect)
   - `KeyCollectionDecomposed.java` (with holes to fill)

- `AbstractKeyCollectionTest.java`
- `AbstractKeyCollectionTestCases.java`
- `KeyCollectionMonolithicTest.java`
- `KeyCollectionDecomposedTest.java` (with holes to fill)

In Canvas, you can find a description of the intended workflow, using Mercurial. Submit the following files with your code filling the holes:

- `KeyCollectionDecomposed.java`
- `KeyCollectionDecomposedTest.java`
- `SecureKeyCollection.zip` (zip archive of project directory, including a Mercurial repository with at least three commits)

Do not include **`package`** statements.

General Rules

1. Submit text documents as PDF `*.pdf` or plain ASCII `*.txt` (no `*.doc`).
2. In each submitted file containing your work, write:
   - full author name,
   - id number,
   - date of latest change.

   In a program text you, obviously, do this in a comment.

   In text files (including program code), write author information above the *cut line*, being the first line containing the *cut mark* `--8<--`. This enables Momotor to provide an anonymized view during peer reviews.
3. Make sure you work neatly; pay attention to layout, spelling, and grammar.
4. You are responsible for checking your work before submitting it.
5. Submit only your own work. All assignments are to be made individually.

