# General Preparation for 2IP15

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

`www.win.tue.nl/~wstomv/edu/2ip15`

1. Make sure you have registered for 2IP15 at `oase.tue.nl`. Once you have done that, you will also be registered for 2IP15 at `peach.win.tue.nl`.
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 2IP15.
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 2IP15*
   - *Checklist for larger OO programs*
   - *Notation*
   - *Specification*
5. Make sure you have installed *Java 7 JDK*, *NetBeans 7.2*, and optionally *DrJava*.

   Set the indent level of your editor to 4, so that TAB characters are expanded.
6. Download `PrePostDoclet.jar`, and make sure you know how to configure a *NetBeans* project to use it. This makes `javadoc` recognize some additional tags used in 2IP15.

   You can also configure *DrJava* to use it.
7. Install the *Checkstyle* plugin for *NetBeans*, and configure it to use the 2IP15 *Checkstyle* configuration file. This file may be updated in the future.

# Assignments

1. (*Secure Key Collection*) 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 divide it 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 peach$^3$, submit the following files with your code filling the holes:

- `KeyCollectionDecomposed.java`
- `KeyCollectionDecomposedTest.java`

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

2. (*Binary Puzzle Checker*) Details to be supplied.

## 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 peach$^3$ 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.