M. Roggenbach, CS-135 – Lab Class 7

- **To be solved in groups of two.**
- **To be submitted to blackboard by Monday, 1.4., 11am.**
- **Submission: Every student individually, one pdf file.**
- **Please mark clearly with whom you are working together: there should be 2 student numbers on top of your submissions.**
- **No other formats than .pdf will be accepted!!!**

This lab is about producing code which complies with coding conventions. Here, you will use the tool Checkstyle to help you achieve this.

Note: For this lab class there is no need for you to understand how the Sphinx or badFormatting programs work. They work correctly, but are very badly formatted, and break numerous code conventions.

□ Task 7.1

Sphinx formatting – realising the concepts from the lecture

- Download, compile, and run the Java program `Sphinx.java` available at `http://www.cs.swan.ac.uk/~csmarkus/Tools/`

  To this end, start the Java compiler and the Java virtual machine from the command line – see the Computer Instructions at the end of this lab sheet.

  The program reads two integer values and returns an integer.

- The program is currently unreadable for regular humans, however perfectly readable for the computer: You have compiled and ran it!

  Format the program according to the Java formatting rules presented in the lecture.

Submission: The formatted Java program. (Note: there is no need for commenting!)

□ Task 7.2

Code formatting with Checkstyle – looking at the constructs that Checkstyle offers

- Download, compile, and run the Java `badFormatting.java` program available at `http://www.cs.swan.ac.uk/~csmarkus/Tools/`

  To this end, use the Eclipse IDE, i.e., you create a new project in Eclipse and import the program badFormatting to it and run it in Eclipse – see the Computer Instructions at the end of this lab sheet.

- Download the Checkstyle configuration file MarkusChecks.xml available at `http://www.cs.swan.ac.uk/~csmarkus/Tools/`.

- Activate the Checkstyle plugin on the project by:
  - Right click your project in the left hand panel (Package Explorer), click Properties.

  - Select Checkstyle, turn on Checkstyle active for this project.
  - Select the tab Local Check Configurations, click New
  - Add an External Configuration File, with the name MarkusChecks, and make the location point to your locally downloaded file MarkusChecks.xml.

    Note: your path must start with `P:`. This can be achieved by browsing via "Computer" and then "P drive" (`P:`).
  - Select the Main tab again, Change drop-down box to use the MarkusChecks configuration.
  - Click OK Now when you save, compile, etc. Checkstyle will run and check your code.
- Re-format `badFormatting.java` so that it complies to the rules sets on formatting (implemented in the configuration file MarkusChecks.xml). When complete Checkstyle should not produce any warning messages.

Submission: Screenshot(s) demonstrating that there are no Checkstyle warnings anymore in `badFormatting.java`. (Note: there is no need to comment the code.)

Computer Instructions

1 Getting a terminal

Under the "Specialist Apps", open the folder "College of Science". Within this folder, open the folder "Computer Science". There, you find various version of the "Java Development Kit", which provides a terminal configured for Java. For our purposes, any of them will do.

2 Changing to the right directory

Typing `p:` changes the drive to the one with your home directory.

The **Change Directory - Select a Folder (and drive) command:**

```
Syntax
      CD [/D] [drive:][path]
      CD [..]

Key
   /D : change the current DRIVE in addition to changing folder.
Examples

   To change to the Desktop.
   p:\> cd Desktop

   To change to the parent directory.
   p:\Desktop> cd ..
```

3 The Java commands

These commands can be typed in the command line of the terminal:

1. `javac <filename.java>` compiles a program.
2. `java <filename>` executes the .class file.
3. `javadoc -d <directory-name> -version -author <filenname.java>` produces the documentation.

4 Recommended editor

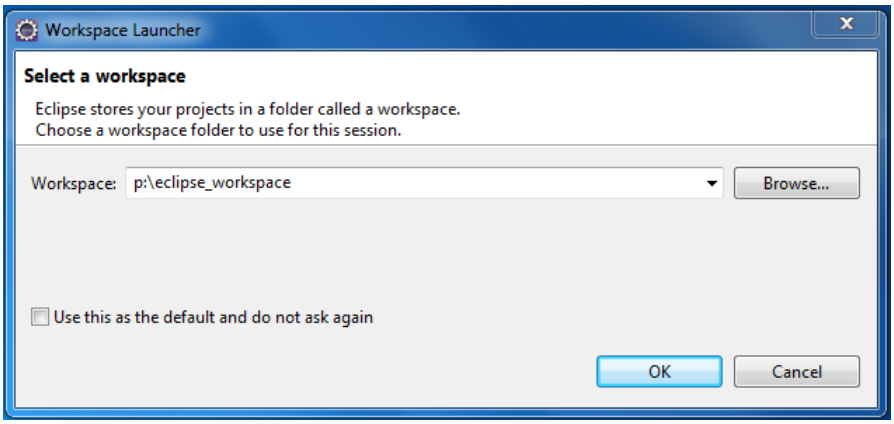
Under the "Specialist Apps", open the folder "College of Science". Within this folder, open the folder "Computer Science". There, you find the program "Notepad++", which provides a reasonable environment to edit Java programs.

5 Making a screen-shot

Click on 'Start', type 'Snipping Tool' in the search field, press 'enter'. Use the tool.

6 Eclipse

Under the "Specialist Apps", open the folder "College of Science". Within this folder, open the folder "Computer Science". There, you find the program "Eclipse". When you start Eclipse you might be asked for the workspace path. This path should be set as follows:



6.1 Making a new project

1. Click `File` → `New` → `Project` → `Java Project`.
2. Typing a good project name i.e. `Sphinx`.
3. Click `Finish`.

6.2 Importing a file into a project

1. Expand your project, say `Sphinx` in the left hand panel (Package Explorer),
2. Right click the `src` folder, click import.
3. Select `File System` under `General`, click `Next`.
4. Locate the directory containing the `Sphinx.java` file, click `OK`.
5. Check the file, e.g. `Sphinx.java`, in the right hand list, Click `Finish`.

6.3 Running a program

You run a program, e.g., `Sphinx.java`, by clicking the play icon. This may bring up a wizard where you need to select to run a `Java Application`. You may need to show the `Console` view by clicking `Window` → `Show View` → `Console`.

6.4 Activating JUnit4 for a project

1. Right-click on your project and select `Properties`.
2. Click on `Java Build Path`.
3. Select `Libraries`
4. Select `Add Library`.
5. Select `Junit`.
6. Click on next, select the Junit Version `JUnit 4`.
7. Click `Finish.`
8. Click `OK.`