Exercises

Design Patterns: Singleton

Java SE and Java EE patterns and best practices

João Miguel Pereira – http://jpereira.eu

0 Prerequisites, assumptions and notes

- Have Maven 2 installed in your computer
- Have Eclipse installed in your computer (Recommended: Indigo Version)
- I’m assuming you’re running the exercises in Ubuntu
- It’s recommended that you place all Design Pattern exercises under a common directory. For example: `${user.home}/javatraining/designpatterns`

  During the exercises I will refer this directory as `${designpatterns.exercises.folder}`
- In every exercise I will refer the directory where you are working as `${project.dir}`.

1 Quick Start Exercise

You will put your hands on a small program and apply the Singleton Design Pattern.

1.1 Checkout code and create eclipse project

In this step you will checkout the code to `${project.dir}`.

Complete the following tasks. ↓

1. Go to the `${project.dir}` directory

   ```
   cd ${project.dir}
   ```

2. Checkout the code from code.google.com

   ```
   svn checkout
   http://javatrainings.googlecode.com/svn/designpat
   terns/trunk/singleton
   ```

3. Enter the created directory and run the tests to check that everything is ok. NOTE: You will se a failing test. Don’t worry, you will fix it soon.

   ```
   mvn test
   ```

4. Enter the created folder and generate the eclipse project

   ```
   mvn eclipse:eclipse
   ```

5. Import project into eclipse

✓ *you're done! You have now the project ready to refactor.*

1.2 Implements a singleton.

You will need to convert the class `ReportBuilder` to a singleton and experiment the two types of concurrency strategies for a lazy singleton.

Complete the following tasks. ↓

1. Open the `ReportBuilder`, analyze it and discuss it.
2. Change the class to make it singleton.
3. Open `ReportBuilderTest`, analyze it and discuss it.
4. Run the test `testMultiThreading` and see the output. Try to interpret it by analyzing also the class `ReportBuilder`
5. Fix the failure of `testMultiThreading`. Experiment to measure the time using a synchronized method and a double-check locking

✓ *You're done.*