Exercises
Design Patterns: Decorator

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 Decorator 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/decorator
```

3. Enter the created directory and run the tests to check that everything is ok.

```
mvn test
```

4. Enter the created folder and generate the eclipse project

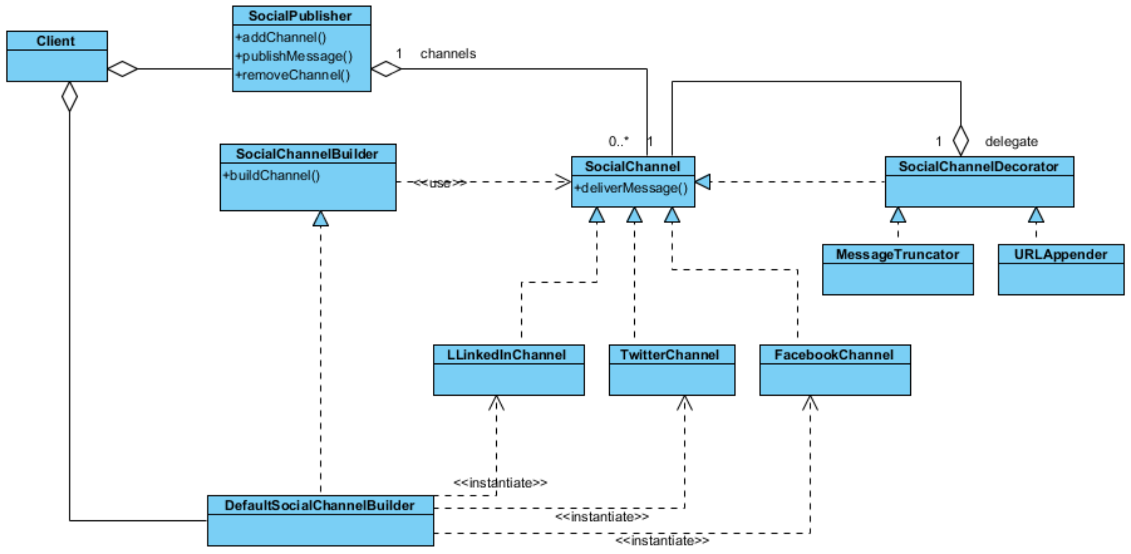
```
mvn eclipse:eclipse
```

5. Import project into eclipse

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

1.2 Add new decorator

The program that you will extend is using the Decorator pattern. The design is the following:

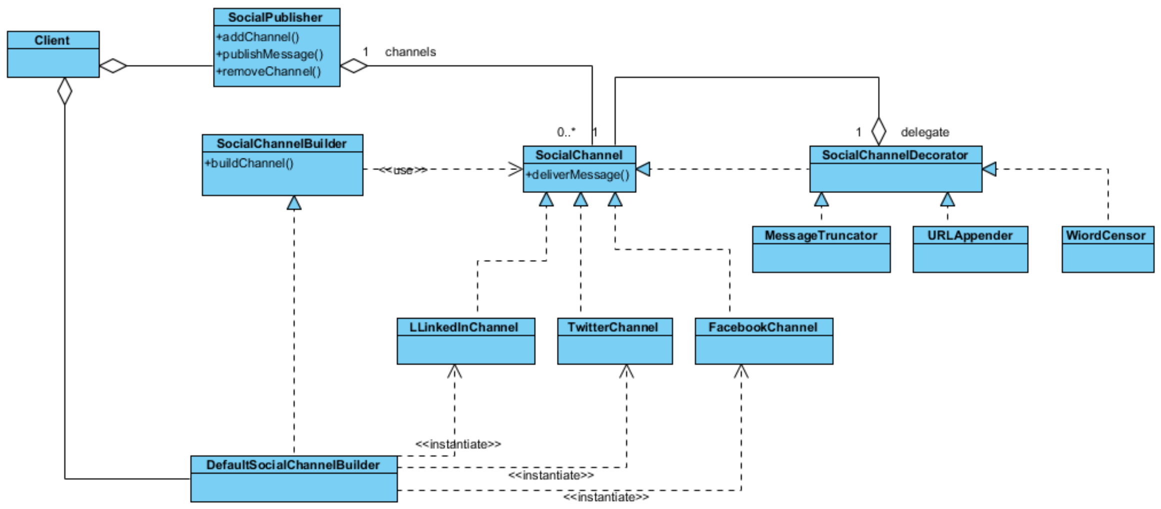


You task is to add a new decorator that will censor certain words. For example:

If the message is "Microsoft Windows is great!!" and the word "Microsoft" is censored, then the delivered message should be "### Windows is great!"

The words that will be censored are of your choice.

The final design should be something like:



You should not change any existing code, unless you have strong reasons for it.

***Complete the following tasks.*↓**

1. Open the project “Decorator” in eclipse
2. Analyze all tests and the implementations and discuss it
3. Create the Test for the new Decorator
    a. In `${project.dir}/src/test/java,` under package `eu.jpereira.trainings.designpatterns.structural.decorator.channel.decorator,` create the test class named `WordCensorTest`
    b. Implement the test that verifies that messages sent to a channel decorated with `WordCensor` are censored correctly. See how the test `MessageTruncatorTest` was developed and steal some ideas from there.
4. Implement the `WordCensor` decorator.
    a. Create the new class under `${project.dir}/src/main/java` in package `eu.jpereira.trainings.designpatterns.structural.decorator.channel.decorator,` next to the rest of decorators
    b. See other Decorator to steal ideas
5. Implement a test to chain your decorator with other decorators
    a. Create a new test class, named `ChainCensorDecoratorTest`

   b. Implement a test that chains you decorator with the other two decorators. See example in `SocialChannelDecortatorIntegrationTest`
6. Generate cobertura report to check that you have an acceptable test coverage percentage for your decorator

✓*You're done.*

1.3 Add new channel (Optional)

Try to add a new channel that will deliver messages to Google Plus. Add the new channel without touching existing code. Try to understand how the current program follows the Open Closed Principle.