Hamza Amjad

CS 474 Homework-3

netID: hamjad2

CS 474 Project 3 Documentation

Project Design (Java 13):

***I have used JAVAPOET library to generate the code for design Pattern. For example, classes, interfaces, and methods**

The Design Pattern Code Generator is build using 3 design patterns. The design patterns are Template, Façade, and Factory Method. The core functionality of the project revolves around the main classes. The Template abstract class, Façade design maker and the abstract factory class. The Template class has only abstract 1 method and a final method that calls the abstract method.

- The are other 8 classes of all the design patterns, and those 8 classes extends the Template abstract class.
- The Design_pattern_maker generates the design pattern for user using switch cases. The Method will get the input key from the use, it will match the key in switch case. It will get the key from Configs class and send to the PatternFactory Class.
- PatternFactory class is my Factory Method. It has one method that returns objects of the Design Patterns. For example, it will return the object of Abstrac_Factory_pattern class.
- Testing is used to test classes to make sure everything is working correct and the required generated files are correct.

There are other files that has been used in this project.

- There is myapp.conf and Configs.java files. Configs.java files gets the required keys for design patterns so PatternFactory class can use this key to return the objects of design patterns.
- Logback library has been used to get the logs from all around the project.

Deployment:

Dependencies used: JavaPoet, Slf4j, lockback and typesafe for Config

```
testCompile group: 'junit', name: 'junit', version: '4.12'
compile group: 'com.typesafe', name: 'config', version: '1.4.0'
compile 'com.squareup:javapoet:1.10.0'
compile group: 'org.slf4j', name: 'slf4j-api', version: '1.7.+'
compile group: 'ch.qos.logback', name: 'logback-classic', version: '1.0.13'
```

Name Clash Analyzer Methods:

Libraries used**: File, VirtualFile, LocalFileSystem, SourceVersion**

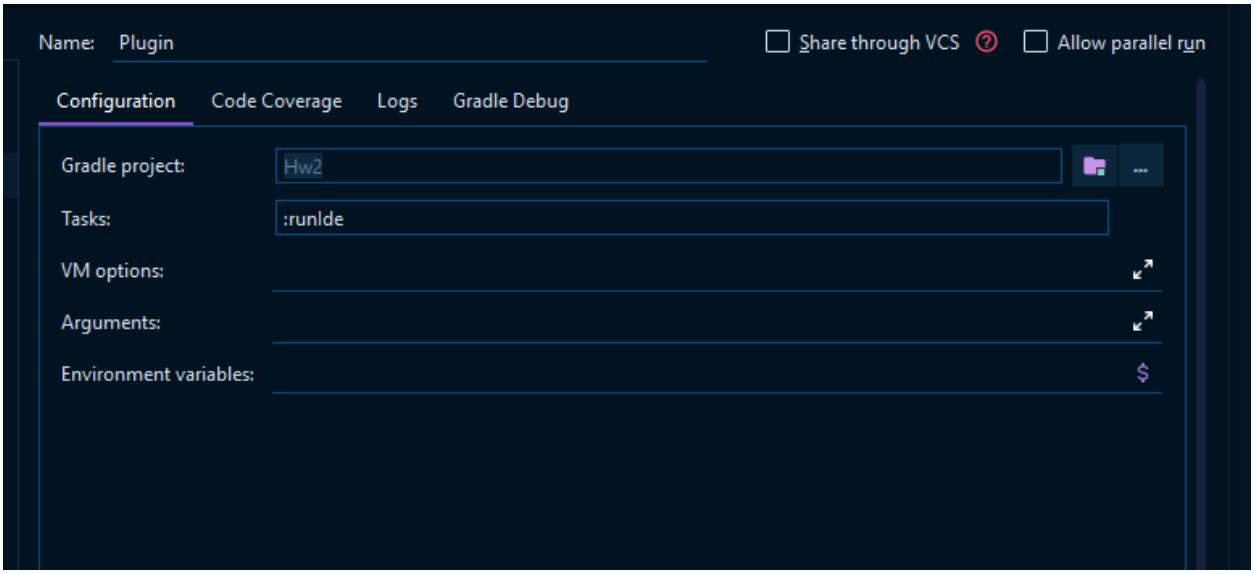
**In this project, The NCAM only checks if the name of classes already exist or not because this project only asks for class names as strings.

There are Name Clash Analyzer Methods in Configs class and they return Boolean. These Methods takes 2 parameters, user input and AnAction event. First, it refreshes the project's workspace then the NCAM gets the package path of a design pattern using Virtual File object and checks if the filename is already existing in the package by creating File object. If the filename exists, an Error will be generated.

Another Method is used to check the illegal name for classes which returns Boolean. Such as, abstract, interface, enum, int e.t.c all these names are illegal. If the user input matches these key, an Error will be generated. This method is being used since Homework1.

How to run?

- Import the project in IntelliJ using Gradle build **AND select IntelliJ Plugin Platform if Necessary** OR run through command line by writing "gradle runIde" without quotes**. (Gradle version is 5.2.1 for command line otherwise it may not work because of version compatibility)**
- Go to run and select edit configurations and use the following settings in IntelliJ

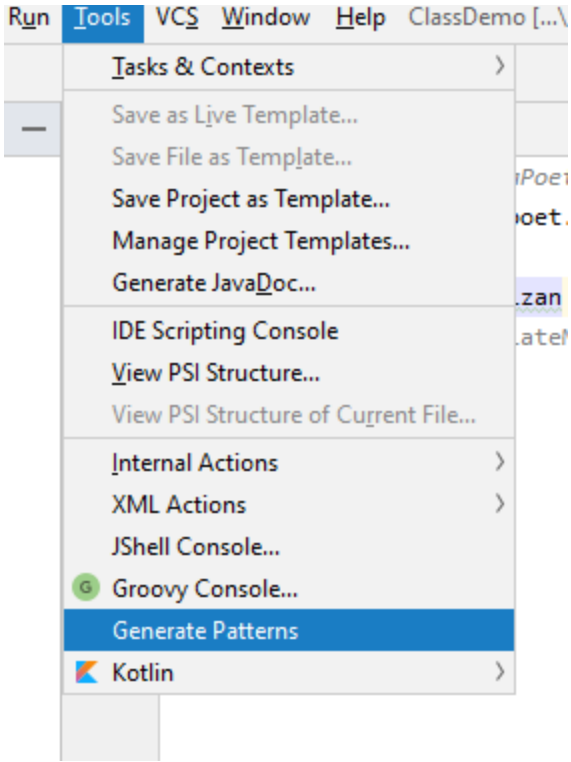


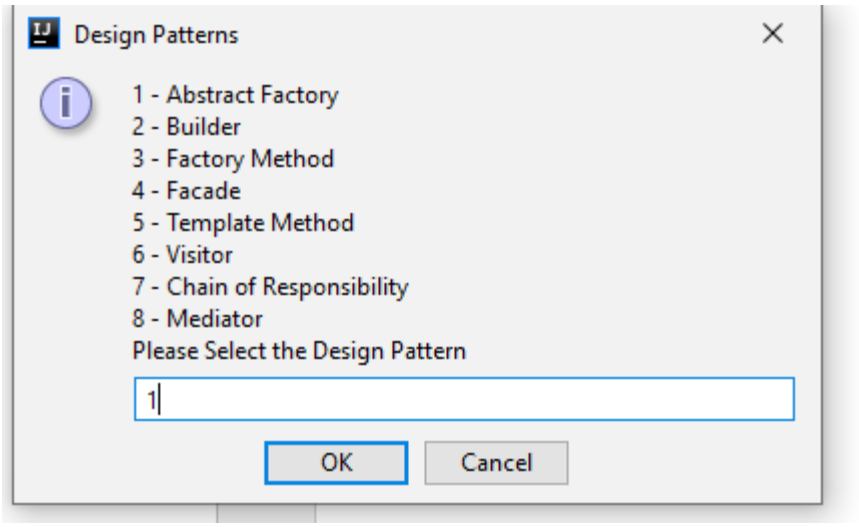
- The main is DaPaCoG class in which the first menu will be provided for the user to select.
- Every design pattern class is taking inputs based in its own design. For example, Abstract factory Design patter might take more inputs than Template Design Pattern and so on.

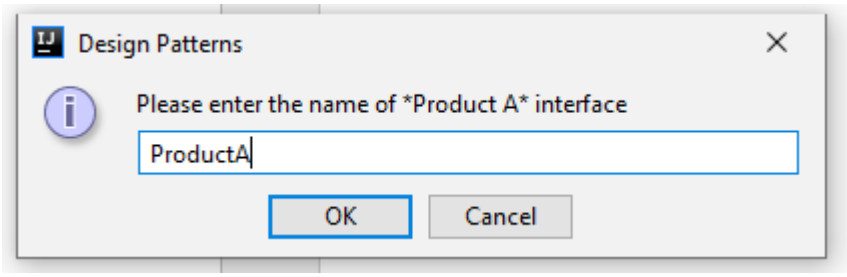
- You will only be able to generate pattern one at a time. You must run the project again to generate another design pattern.
- The inputs will be checked for every input. Make sure you give the legal class name. “Class Class”, “abstract”, “1234” etc, these types of inputs are not allowed for the class name.
- After giving the required inputs, the generated files will be available to compile in the Design pattern package. For example,

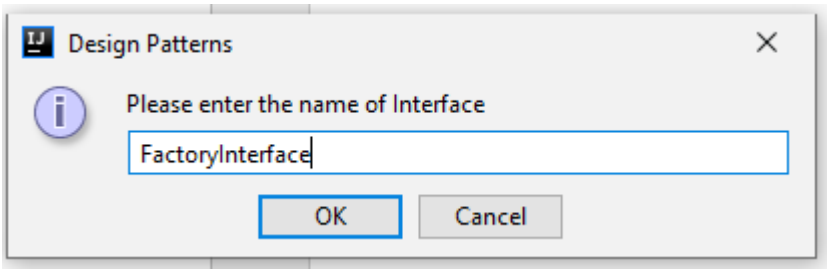
Sample input for Abstract Factory

Go to “ToolsMenu” in opened project in newly opened IDE and Select the last option “Generate Patterns” to start generating the patterns. If the patterns don’t show up, wait for 2 mins or another solution is to refresh the IDE by minimizing it.



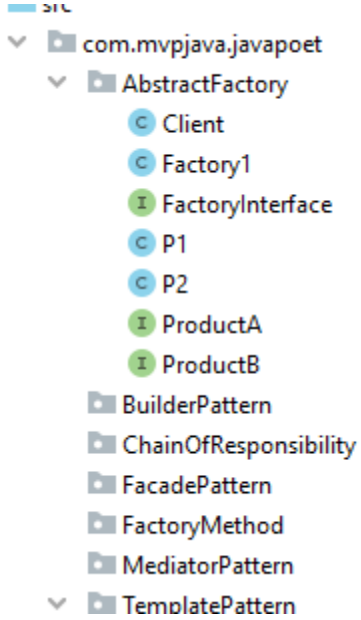






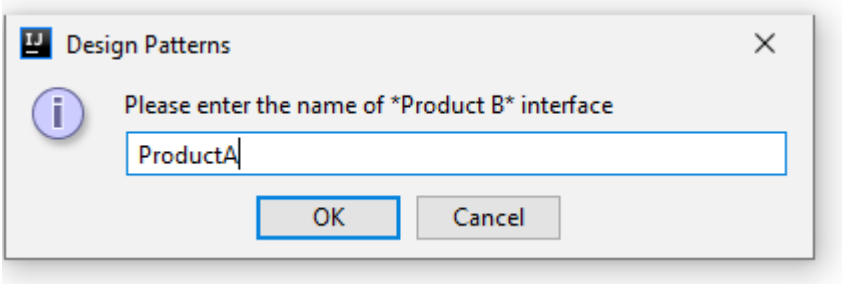
And so on ……

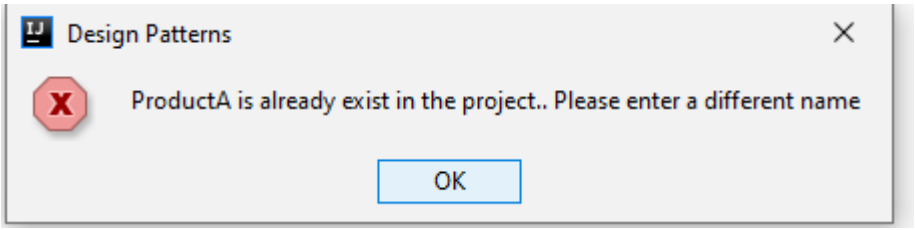
The Files will be available to compile in the following package:

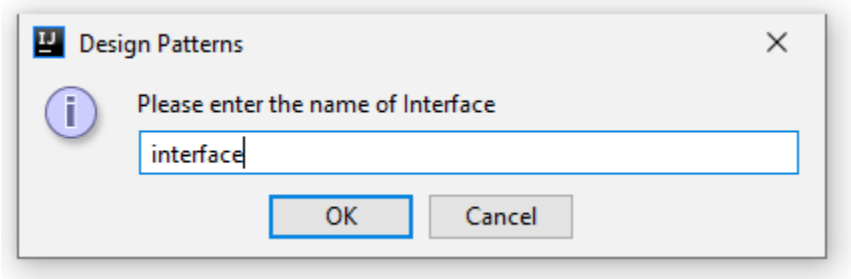


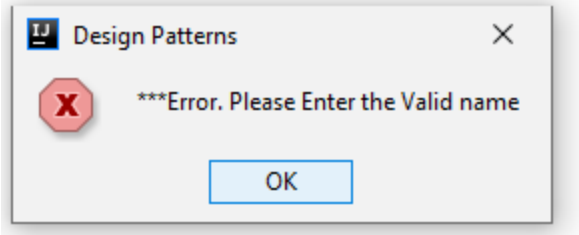
*The sample also shows the illegal name example. So, make sure to give the legal class names.

Sample Input for Name Clash Checking









Settings

To run tests, simply type “gradle test” in the command line.

Or

Use the following arguments and setting in IntelliJ IDE