**Documentation**

In this second homework, I tried to create an object-oriented design and implementation of a program that runs within an Integrated Development Environments (IDEs) to generate the code for a selected design pattern and adds this generated code to an open Java project. In that, this homework is based on homework one and uses it as a module to provide needed functionality, i.e., a call to a method within this module to generate code for a design pattern returns an object of some type that contains the generated code.

As one of the requirements of this homework is to create a plugin which will activate a dockable windowpane within IntelliJ, I used Tool Windows to achieve this requirement. The IntelliJ Platform provides the com.intellij.toolWindow extension point that you can use to create and configure your custom tool windows. In my plugin project, I created a Java class MyToolWindowFactory that implements the ToolWindowFactory interface. A screenshot of a cell phone

Description automatically generated

MyToolWindowFactory then uses the class called MyToolWindow to set up the user interface content for the tool window. MyToolWindow has all the necessary java swing elements and their action listeners for my tool window design. It also has an instance of the DePaCoG class that I created in homework one which is responsible for generating the desired design patterns.

A screenshot of a cell phone

Description automatically generated

To get the necessary inputs form the users, MyToolWindow uses UserInputDialogWrapper class to prompt a dialog. In order to achieve this behavior UserInputDialogWrapper extends DialogWrapper class. The DialogWrapper is the base class which is supposed to be used for all modal dialogs (and some non-modal dialogs) shown in *IntelliJ Platform* plugins. A screenshot of a computer

Description automatically generated