JC2002 Java Programming - Practical 4 (Day 5)

The goal of this practical is to get more experience with defining methods and classes in Java, as well as using Maven and JUnit for automated testing of your Java programs.

1. Create a file ***'Calculator.java'*** which will define the **Calculator** class. Within this class, define three public methods: **add()**, **subtract()**, and **multiply()**. Each of these methods should take two integer parameters and return the result of the corresponding arithmetic operation as an integer. Remember to add comments to your code.

2. Create file ***'RunApp.java'*** which will contain a class **RunApp** with **main()** method. Use this method to create an instance of the calculator. Use the Scanner object to receive an input from the console so the user can perform the calculations. The flow should be as follows:

   - User enters the first number (followed by enter).
   - User enters operator '+', '-', or '*' (followed by enter) to decide which method of the calculator object will be called. Hint: you can use **scanner.next().charAt(0);** to read the first character of the user input from the keyboard into a *char* variable (**scanner** is an instance of **Scanner** for system input).
   - User enters the second number (followed by enter).
   - The program returns the result of the selected operation.

   Note the program should run indefinitely (i.e., the user can perform multiple calculations) until 'q' or 'Q' (as "quit") is typed as operator.

3. Turn the application into a Maven project. You can create a dummy Maven project, as instructed in the lecture slides. If you have not Maven installed already, install it as instructed in the lecture slides.

   To create a project, enter command **'mvn archetype:generate'**. After some processing, Maven will ask questions. For the first two questions, you can use the defaults (just press enter). For **groupId**, you can enter *'jc2002'*, for the **artifactId**, you can enter *'calculator'*, and for the package name, you can enter *'mypkg'* (actually, you can choose the package name yourself, but in this practical, we assume that the package name is *'mypkg'*).

   If the project is created successfully, new directory ***'calculator'*** with several subdirectories have been created. Copy the files ***'Calculator.java'*** and ***'RunApp.java'*** to the directory ***'calculator/src/main/java/mypkg'*** (replace *'mypkg'* with the actual package name if you named your package differently). You can delete file ***'App.java'*** in the same directory. Make sure you have package definition **package mypkg;** in ***'Calculator.java'*** and ***'RunApp.java'***.

   Move to directory ***'calculator'*** and open file ***'pox.xml'*** there. Edit section 'maven-jar-plugin' as follows:

```
<plugin>
 <artifactId>maven-jar-plugin</artifactId>
 <version>3.0.2</version>
 <configuration>
  <archive>
   <manifest>
    <addDefaultImplementationEntries>true</addDefaultImplementationEntries>
```

```
      <addDefaultSpecificationEntries>true</addDefaultSpecificationEntries>
      <addClasspath>true</addClasspath>
      <mainClass>mypkg.RunApp</mainClass>
    </manifest>
   </archive>
  </configuration>
 </plugin>
```

Then, use command **'mvn package'** to generate a JAR package combining classes **Calculator** and **RunApp**. Try to run the resulting JAR file in directory ***'target'*** by using command **'java -jar target/calculator-1.0-SNAPSHOT.jar'** (make sure this is the name of the JAR file).

4. Modify the default test file ***'AppTest.java'*** in directory ***'src/test/java/myorg'*** to create some tests for the methods **add()**, **subtract()**, and **multiply()** of class **Calculator**. Make at least two tests for each method. Include negative and large values, and other unusual inputs. Enter command **'mvn clean test'** to compile the code and run the tests.

   When all the tests are passed, modify the methods **add()**, **subtract()**, and **multiply()** to give intentionally wrong results. Run the tests again to ensure that the tests fail.