Coding Challenge Solution for Drawing Application

# Code / Project Structure

The newly created project is a maven java project. The project information is available in **pom.xml** file in the root directory of the project.

Just like any typical maven project:

* The main source codes are available in **src/main/java** directory
* The test source codes are available in **src/test/java** directory

# Dependencies

1. JDK 8
2. Maven
3. spring-context version 4.3.12.RELEASE (and its relevant sub dependencies)
4. commons-lang3 version 3.6
5. junit version 4.12
6. hamcrest-library version 1.3
7. mockito-all version 1.10.19
8. spring-test version v 4.3.12.RELEASE

# How to Build and Run the Application

As mention in Dependencies section above, JDK 8 and Maven is required to build and run this application.

To build this application, please navigate to the root folder where the application is extracted and then run:

**mvn clean package**

Maven will automatically download the dependencies (number 3 to 8 from the section above) from Maven repositories (if they don’t exist) for this build purpose.

The build may fail if the dependencies do not exist in your organisation’s maven repository or if you do not have access to the Maven central repository. This depends on your Maven’s setting (M2\_HOME/conf/settings.xml).

Upon successful build, you would find 2 jar files below in the target directory:

* drawing-1.0.0-SNAPSHOT.jar
* drawing-1.0.0-SNAPSHOT-jar-with-dependencies.jar

To run this application, please navigate into the target folder and run the following:

**java -cp drawing-1.0.0-SNAPSHOT-jar-with-dependencies.jar com.cs.drawing.DrawingApplication**

# Application Behaviours

This application behaviours follow the requirements given, which is a simple console version of a drawing program.

In a nutshell, the program should work as follows:

1. Create a new canvas
2. Start drawing on the canvas by issuing various commands
3. Quit

The table below represents the commands available in this application:

|  |  |  |
| --- | --- | --- |
| **Command** | **Description** | **Type** |
| C (case insensitive) | Creates Canvas | Core Feature |
| L (case insensitive) | Creates Line | Core Feature |
| R (case insensitive) | Creates Rectangle | Core Feature |
| B (case insensitive) | Creates Bucket Fill | Core Feature |
| Q (case insensitive) | Quits the application | Core Feature |
| H (case insensitive) | Prints the Help / list of commands | Additional Feature |
| P (case insensitive) | re-Prints / re-render the image from the memory | Additional Feature |

### Important Notes:

* Line, Rectangle and Bucket Fill can only be created after a Canvas has been created.
* Creating a new Canvas when there is an existing Canvas will overwrite the old Canvas. The object/s (e.g. Line, Rectangle or Bucket Fill) on the old Canvas will also disappear.
* Drawing a Bucket Fill on an existing object/s will overwrite the colour on the existing object/s with the colour specified for the Bucket Fill.

# Technical Information

This application is using Spring IoC. The main class to start this application is com.cs.drawing.DrawingApplication.

When main() method in DrawingApplication class is called, it loads Spring context by using AnnotationConfigApplicationContext class.

This context loader uses com.cs.drawing.config.SpringConfig configuration and initialises spring beans by using ComponentScan specified in this class.

When Spring beans are all ready, DrawingApplication.start() is being called to start listening for commands from user.

# Code Coverage Test

Code coverage test plugin cobertura is configured in Maven configuration file pom.xml.

To generate code coverage test, please navigate to the root folder where the application is extracted and then run:

**mvn cobertura:cobertura**

The generated report can then be found in target/site/cobertura/index.html file

For your convenience, the code coverage test report as per last build is included attached.