**Drawing Tool**

* **Requirement:**

You're given the task of writing a simple console version of a drawing program.

At this time, the functionality of the program is quite limited but this might change in the future.

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

* **Design:**

This is a console application to create Canvas and draw shapes on Canvas.

Below are the main classes used in the tool.

1. DrawingTool
2. Canvas
3. CanvasItem: abstract class inherited by below classes
4. Line
5. Rectangle
6. ConcreteShapeFactory: to get the shape based on user’s choice.
7. ColourfulConsole: A custom console to fill console with color.
8. ICommand:
9. CreateCommand
10. DrawCommand
11. BucketFillCommand

* **Design Patterns and Principles used:**

Factory Method:

In the tool depending on user’s input, different shapes like Line, rectangle has be drawn. Here we can use factory method to create instances depending on user’s input.

In case in future if we want to add new type of shape, no need to change client’s code.

Command Pattern:

It decouples the classes that invoke the operation from the object that knows how to execute the operation. Here all operations like create canvas, draw shapes and bucket fill will be invoked by commands.

Extensions to add a new command is easy and can be done without changing the existing code

Single Responsibility Principle:

Canvas is only responsible for its responsibilities like draw, fill, create, etc. It need not have to worry about which shape has to be added that is done by factory class here.

Also, Shape classes (Line, Rectangle) are responsible for validate coordinates required to draw that shape and drawing it.

Open-Close Principle:

As per Open-Close Principle, Canvas class and CanvasItem class is open for extension and closed for modification. All new shapes will be added by creating new class of that class.

* **Unit tests:**

All the unit tests success as well as failure are covered.