Assignment 3 (9 marks)

Paint Application

Objectives

- Design an object-oriented model for geometric shapes.
- Draw UML diagrams that represents your analysis and design.
- Apply the OOP concepts of inheritance and polymorphism to your design.
- Apply different design patterns to your model.
- Create an advanced GUI with 2D Graphics capabilities.
- Get to know XML and JSON formats.

[30%] Part 1: Geometric Shapes Data Model

Description

Geometric shapes belong to different groups (ex: Elliptical Shapes, Polygons, Sectors…etc.); members of these different groups are related to each other in the sense that they share common properties. To be able to implement an efficient and object-oriented drawing application, it is essential to design a model that takes these relations into consideration.

Tasks

1. Design an object-oriented model that covers the following geometric shapes: **Line** Segment, **Circle**, **Ellipse**, **Triangle**, **Rectangle** and **Square**.
2. Draw a Use Case Diagram that shows the analysis of your model.
3. Draw a UML Class diagram that represents your model, showing all the classes, attributes and methods.
4. Provide at least 3 Sequence diagrams which will show three different scenarios: drawing a Circle, editing an Ellipse, save some shapes to a JSON file.
5. Start working on the report.
6. Apply the concepts of inheritance and polymorphism to your design.

Integration

You should organize your code following the **MVC** AP under **three** main packages with name:
paint.model
paint.controller
paint.view

You should implement the following **two** interfaces by two different classes:

```
package paint.model;

public interface Shape{
	public void setPosition(java.awt.Point position);
	public java.awt.Point getPosition();

	/* update shape specific properties (e.g., radius) */
	public void setProperties(java.util.Map<String, Double> properties);
	public java.util.Map<String, Double> getProperties();

	public void setColor(java.awt.Color color);
	public java.awt.Color getColor();

	public void setFillColor(java.awt.Color color);
	public java.awt.Color getFillColor();

	/* redraw the shape on the canvas,
	   for swing, you will cast canvas to java.awt.Graphics */
	public void draw(Object canvas);

	/* create a deep clone of the shape */
	public Object clone() throws CloneNotSupportedException;
}
```

```
package paint.controller;

public interface DrawingEngine {
	/* redraw all shapes on the canvas */
	public void refresh(Object canvas);

	public void addShape(Shape shape);
	public void removeShape(Shape shape);
	public void updateShape(Shape oldShape, Shape newShape);
```

```
        /* return the created shapes objects */
        public Shape[] getShapes();

        /* limited to 20 steps. You consider these actions in
         * undo & redo: addShape, removeShape, updateShape */
        public void undo();
        public void redo();

        /* use the file extension to determine the type,
         * or throw runtime exception when unexpected extension */
        public void save(String path);
        public void load(String path);

        // *************** bonus functions ****************
        /* return the classes (types) of supported shapes already exist and the
         * ones that can be dynamically loaded at runtime (see Part 4) */
        public java.util.List<Class<? extends Shape>> getSupportedShapes();

        /* add to the supported shapes the new shape class (see Part 4) */
        public void installPluginShape(String jarPath);
}
```

[50%] Part 2: Drawing and Painting Application

Description

Drawing and painting applications are very popular and have a huge user base; they generally offer a big number of features that includes but is not limited to: Drawing, Coloring, and Resizing. They also include several built in, and possibly extensible set of geometric shapes, and classically, they allow the user to undo or redo any instructions to make the application more usable.

Tasks

1. Implement your design from part 1 using Java. (check "**Singleton DP** and **MVC**") (optional: "Observer DP").
2. Design and implement a GUI that allows the following functionalities for the user on all the shapes defined in part 1: **Draw**, **Color**, **Resize**, **Move**, **Copy**, and **Delete**. (check "**Factory DP**") (optional: "State DP and Prototype DP").

3. Implement your application such that it would allow the user to undo or redo any action performed. (check "**Command DP** and **Memento DP**").
4. You should implement a user-friendly drawing way for your Paint App. You should draw by mouse dragging.
5. [**Bonus**] Allow the user to group select objects, then apply a resize, move, or delete operation on them.

[20%] Part 3: Save and load

Description

One of the main features in any paint application is saving user's drawings in a file and loading it later to modify it.

Tasks

1. Provide an option in GUI to save the drawing in XML and JSON file (You should implement both). (check: "**Strategy DP**").
2. Provide an option to load previously saved drawings and modify the shapes.
3. User must choose where to save the file.

[Bonus] Part 4: Dynamic Application Extensions and plug-ins

Description

The concept of dynamic class loading is widely spread in computer applications; it is an option that allows the user to extend application features at runtime. This can be easily done by the dynamic class loading capabilities that OOP languages offer.

Tasks

1. Create a new Shape class like "round rectangle" and compiler it to generate a jar file, the class should be in the path *paint.plugin*.
2. Provide an option in the GUI of your application that allows for selecting the jar file.
3. On selecting and loading the file, the isolated shape should be appended to the available list of shapes that can be drawn in the application. (check "**Dynamic Linkage DP**").

Notes

- You can check lab 3 "**simple paint**" for a suggested design and a quick start into this project 😊
- It is better to separate the GUI buttons actions into a different class.
- It would be better to code to interface than to code to a concrete class.
- You should maintain a clean code for your program, like: variables, classes, functions naming, code indentation, function length, …etc.
    - You can check this book for more details Clean Code – Robert C. Martin
- Be creative! The required features are only the beginning of what you can do, add more features or spice up the required ones, bonus marks will be given to those with eye-catching extra features and user-friendly interfaces.
- It is better to use a git version control service like bitbucket to collaborate with your teammates. However, your repository must be private so no one can copy from it, which will consider an act of cheating for both parties!
- Start to work early, so you can ask us early, and you will finish the assignment in time.
- Again, google is your best friend 😉 the debugger is your second-best friend XD

Report

It should contain the following:

- **Description**: You should briefly describe the application you implemented.
- **Features**: Write down the features of the application.
- **Design Overview**: Make a small overview on your design and the overall structure with minimal details.
- **Assumptions**: Tell us if you had taken any assumption during the analysis, design or the implementation phase. necessary to be clarified.
- **Teamwork**: You should state the detailed division of labor among the team members.
- **Data Structure**: Write down the data structure you have used in the program, like Array List, Linked List, ...etc.
- **Description of the important functions/modules**: Describe the important functions you implemented. Also, write down the classes divisions if you made additional packages.
- **UML Diagrams**: Use Case, Class Diagram, and 3 Sequence Diagrams.

- **User Manual**: You should make a user manual to teach the user how to use the application.
- **Sample runs**: You should provide some screenshots showing the program while running through various cases.
- **References**: Any copied material from anywhere should be referenced. Any discovered unreferenced copied material will result in sever reduction in your grade.

Deliverables

- You should work in groups of four.
- You should develop this assignment in Java.
- You should implement the given interfaces **without changing anything in them**.
- You should implement all the **five** design patterns listed in this document in **bold** plus the MVC architectural pattern.
- You are free to use any GUI framework you like.
- You should submit this assignment online on Schoology, you should include the following into **one compressed file**, then rename it with you **id1_id2_id3_id4**:
  - Your **code** folder.
  - A **self-executable jar file**: The program should be executable by simply double clicking the icon, given that you have a running JRE.
  - A **report** in pdf format.
  - [**Bonus**] A plugin jar file that has a .class file, it should be used to test your implementation in part 4.
- A discussion date will be determined later.
- Your program **MUST NOT** crash under any circumstances, even against malicious users!
- Late submission is not permitted.
- [**Cheating Policy**] Delivering a copy will be severely penalized for both parties, so delivering nothing is so much better than delivering a copy.

Good luck 😊