Alexandria University
Faculty of Engineering
Computer and Systems Engineering Dept.
Second Year
Fall 2016

CS221: Object Oriented Programming
Programming Assignment 2
Due: Thursday, November 3rd 2016

Vector Based Drawing Application

Objectives

Upon completion of this assignment, you will be able to:

- Design an object oriented model for geometric shapes
- Draw a UML class diagram that represents your model
- Apply the OOP concepts of inheritance and polymorphism to your design
- Create an advanced GUI with 2D Graphics capabilities
- Enable dynamic extensions to your applications

Part 1: Geometric Shapes Data Model

Description:

Geometric shapes belong to different groups (ex: Elliptical Shapes, Polygons, …etc). Members of these different groups are related to each other in the sense that they share common properties. In order 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 UML Class diagram that represents your model, showing all the classes, attributes and methods.
3- Apply the concepts of inheritance and polymorphism to your design.

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 a number of built in, and possibly extensible set of geometric shapes, and classically, they allow the user to undo or redo any instructions so as to make the application more usable

Tasks:

1- Implement your design from part 1 in Java.
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, and Delete.
3- Implement your application such that it would allow the user to undo or redo any action performed. (Optional hint: Check the "Command Design Pattern")
4- The cursor should be used to select the location of a shape while drawing it, or moving it to another location, for more accurate control on the shape parameters (ex: size), dialog boxes could be used, or you are free to implement it in a more user friendly way of your choice.

Part 3: 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 Java offers.

Tasks:

1- Pick one of the Shape Classes listed in part 1, and compile it as a class library.
2- Provide an option in the GUI of your application that allows for selecting the class library file.
3- On selecting and loading the file, the isolated shape should be appended to the available list of shapes in the application.

Part 5: Move, Delete and Resize

Description:

A very important feature to add is editing the painting, either by moving, deleting or resizing any object.

Tasks:

1- Allow the user to select an object by clicking on it, and then start to resize, move or delete.
2- When an object is selected, its appearance should change such that the user understands that it is selected. This can be done by a dotted rectangle that shows up around it or by showing boxes for resizing around it.
3- While resizing, show small boxes that the user can drag in order to resize the shape.
4- [**bonus**] Allow the user to group objects to resize, move or delete together as one object

Part 4: Save and load

Description:

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

Tasks:

5- Provide an option in GUI to save the drawing in XML and JSON file (You should implement both).
6- Provide an option to load previously saved drawings and modify the shapes.
7- User must choose where to save the file.

Deliverables

- You have to work in groups of exactly two. No teams of one will be accepted.
- For exceptional reasons, groups can be of a three members. However, they must support Java Network Launching Protocol (JNLP) in their deliverables
- The implementation for the given interfaces.
- Develop this assignment in Java.
- A self-executable Jar: The program should be executable by simply double clicking the icon provided that you have a running JRE.
- You should deliver your source code using your git repository.
- You can use external libraries for GUI or for XML/JSON parsing and formatting, or you can use JDK classes, or write your own ones.
- You should deliver a report that contains the required UML diagram, describes your design thoroughly, and contains snapshots of your GUI and a user guide that explains how to use your application. Any design decisions that you have made should be listed clearly.
- 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.
- Delivering a copy will be **severely** penalized for both parties, so delivering nothing is much better than delivering a copy.