Ozan Isik
215001746
EECS 3311
Section A
lab 02
Kazi Mridul
kmridul@yorku.ca

Part 1: Instroduction.

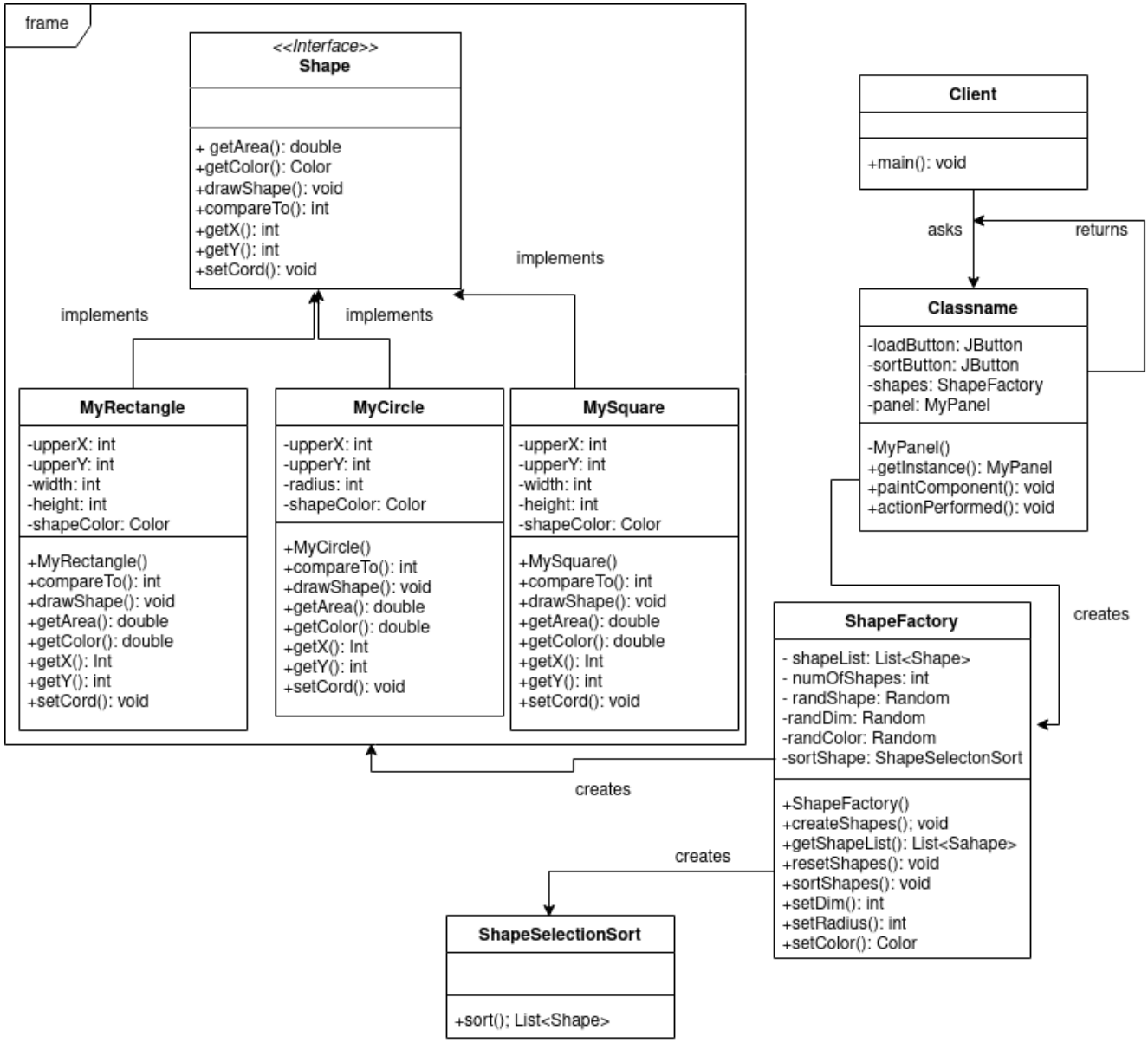
The project is a Swing Java program where we must have interface displaying various shapes with different surface areas. The purpose of this project is to be able to display shapes in the user interface while using a sorting algorithm to sort the shapes based on their surface area. As well ass, learning to use object oriented designs principle and design pattern to build a project.

Some challenges that are associated with the project is learning java swing as I have never used any sort of graphics with java before.

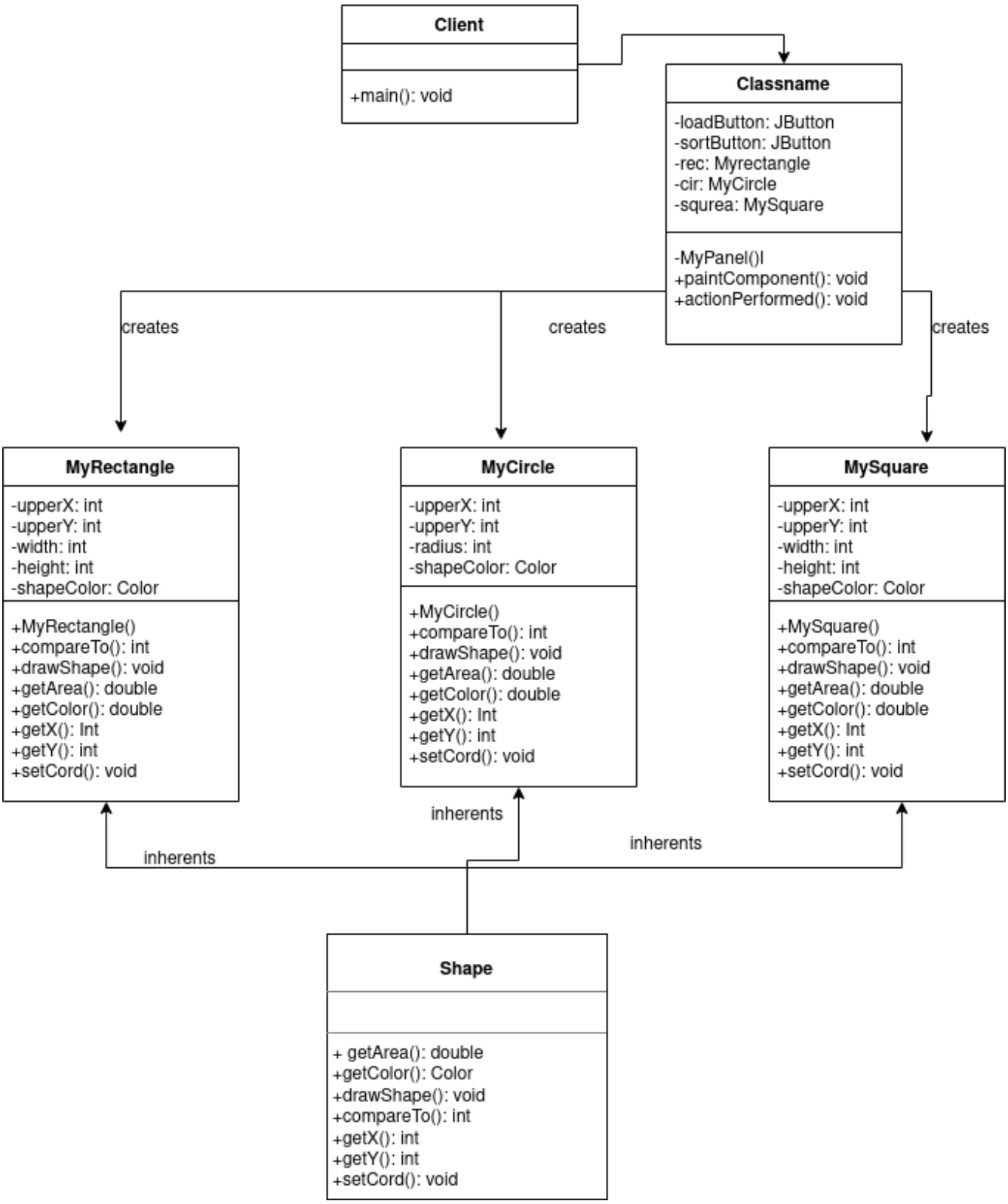
I will be using using object oriented programming to build this program. This will include classes and object to store the key characteristics of shapes and to properly organize my code. I will be using singleton and factory design principles.

The structure of the report will begin with designing of the project and having a clear understanding of the goal. Then putting the design into action while implementing.

Part 2: Design of the solution



In our UML diagram, we are using 2 design patters. Singleton and Factory. The client object creates only one instance of object MyPanal which will only ever return one unique reference to guarantee that there will be no other instance of the object. MyPanel description is that, it is the user interface. The characteristics of this user interface include the Jbuttons and Jframe. These are the characteristics that describes a user interface or as we are going to call it, MyPanel. The next design patterns we will be using is factory design. MyPanel will create a shapeFactory object that will act as the factory object to instantiate every other shapes that we need. The ShapeFactory class is described by the list of shapes it can create. But to store different shape objects in the same list, we must have a generic way to group all the shape object. This is where we will be using an interface called Shape. Shape interface will include all the characteristics of all the shapes. This will include how to draw them selves and how to compare them selves to each other as well as the way to calculate their own areas. MyRectangle, MyCircle, and MySquare classes will all implement the interface Shape as they all share key characteristics. The ShapeFactoryClass will also create ShapeSelectionSort to be able to sort the shapes in the list.


Client
+main(): void
Classname
-loadButton: JButton
-sortButton: JButton
-rec: Myrectangle
-cir: MyCircle
-squrea: MySquare
-MyPanel()l
+paintComponent(): void
+actionPerformed(): void
creates
creates
creates
MyRectangle
-upperX: int
-upperY: int
-width: int
-height: int
-shapeColor: Color
+MyRectangle()
+compareTo(): int
+drawShape(): void
+getArea(): double
+getColor(): double
+getX(): Int
+getY(): int
+setCord(): void
MyCircle
-upperX: int
-upperY: int
-radius: int
-shapeColor: Color
+MyCircle()
+compareTo(): int
+drawShape(): void
+getArea(): double
+getColor(): double
+getX(): Int
+getY(): int
+setCord(): void
MySquare
-upperX: int
-upperY: int
-width: int
-height: int
-shapeColor: Color
+MySquare()
+compareTo(): int
+drawShape(): void
+getArea(): double
+getColor(): double
+getX(): Int
+getY(): int
+setCord(): void
inherents
inherents
inherents
Shape
+ getArea(): double
+getColor(): Color
+drawShape(): void
+compareTo(): int
+getX(): int
+getY(): int
+setCord(): void

This second UML design is taking a different approach. This second Design yields a better design because it is much simpler to follow and build. For a small project as this, We would not need a very complex design. A simple inheritance would allow as to build the program is fewer steps.

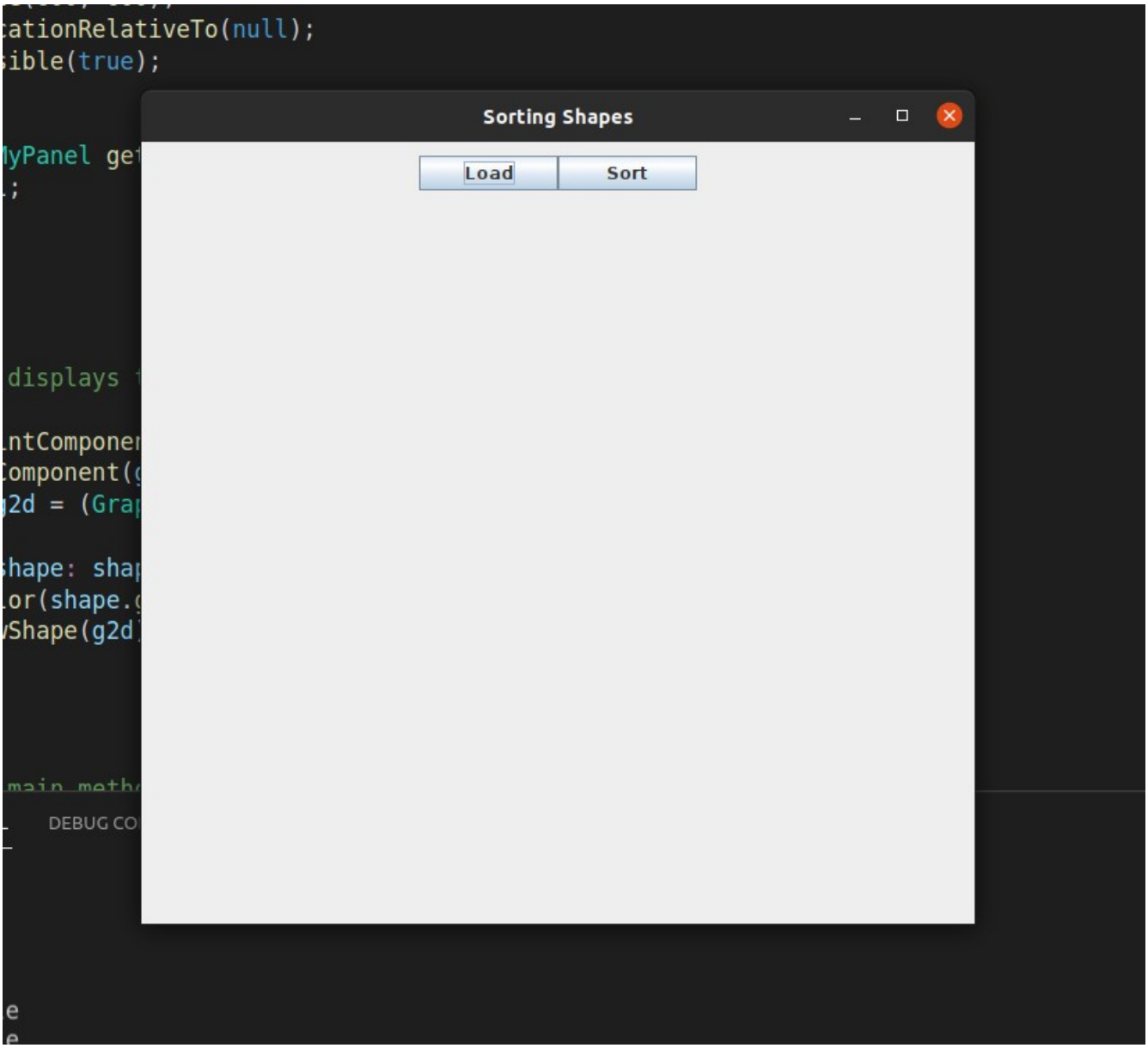
Part 3: Implementation of the solution.

The algorithm technique use was selection sort. Selection sort is a simple comparison algorithm implemented with a nested for loop. We are constantly holding two values from the list. This is done by holding the smallest value from the list and comparing each element in the list to find the next smallest value. When the next smallest value is found, we perform a swap.

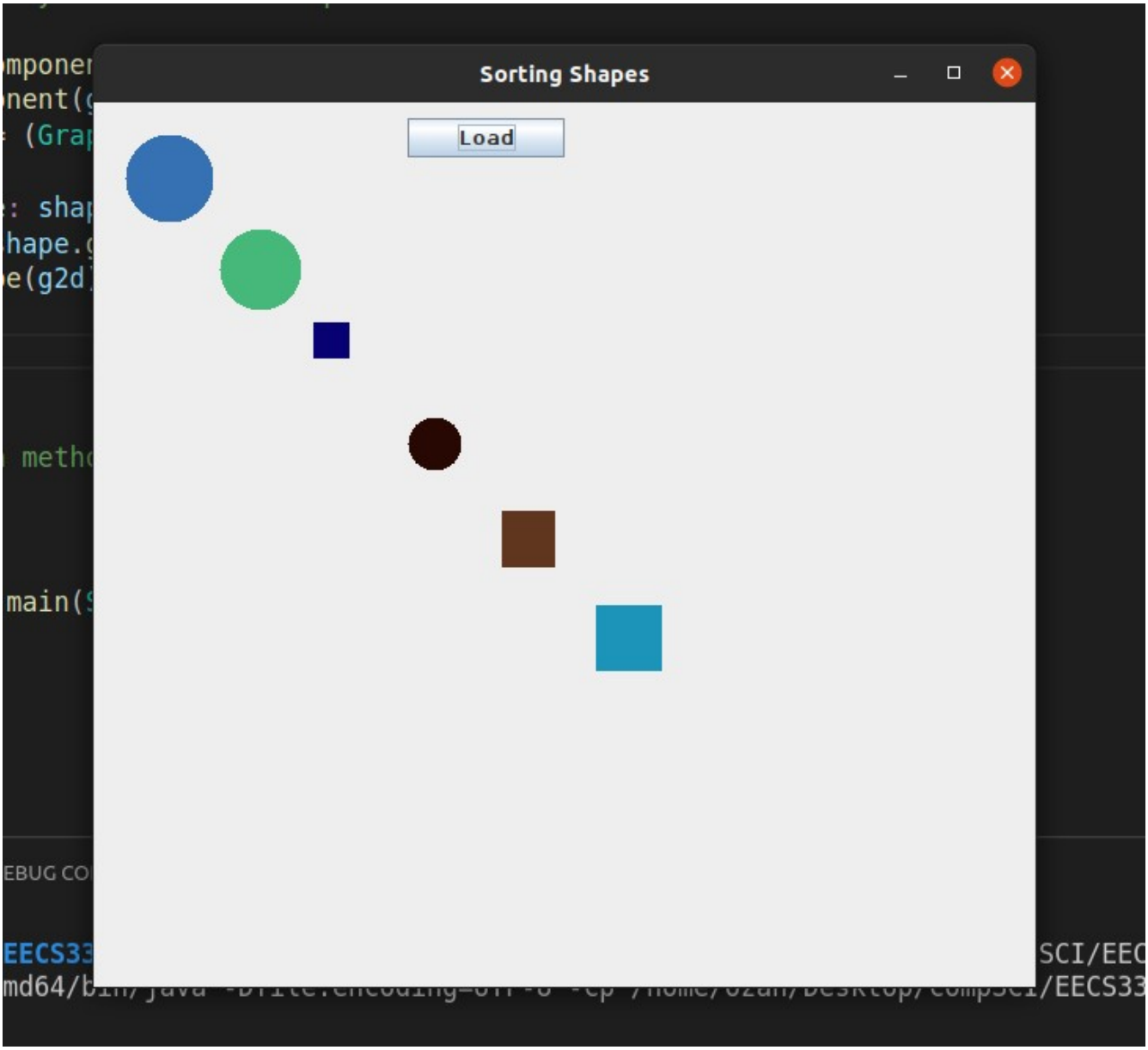
for the implementation, I have used the first UML class diagram with the singleton and factory design patterns. The program starts by creating single instance of MyPanal object which than creates ShapeFactory. From ShapeFactory we instantiate 6 shape object and store them in a single ArrayList. The ability to store different object in a single list is done with a Shape interface. This interface is used to group up common object which are the shapes classes like MyRectangle, MyCircle, and MySquare.

The tools that have been used is the VsCode IDE with the 11.0.11 jdk version.

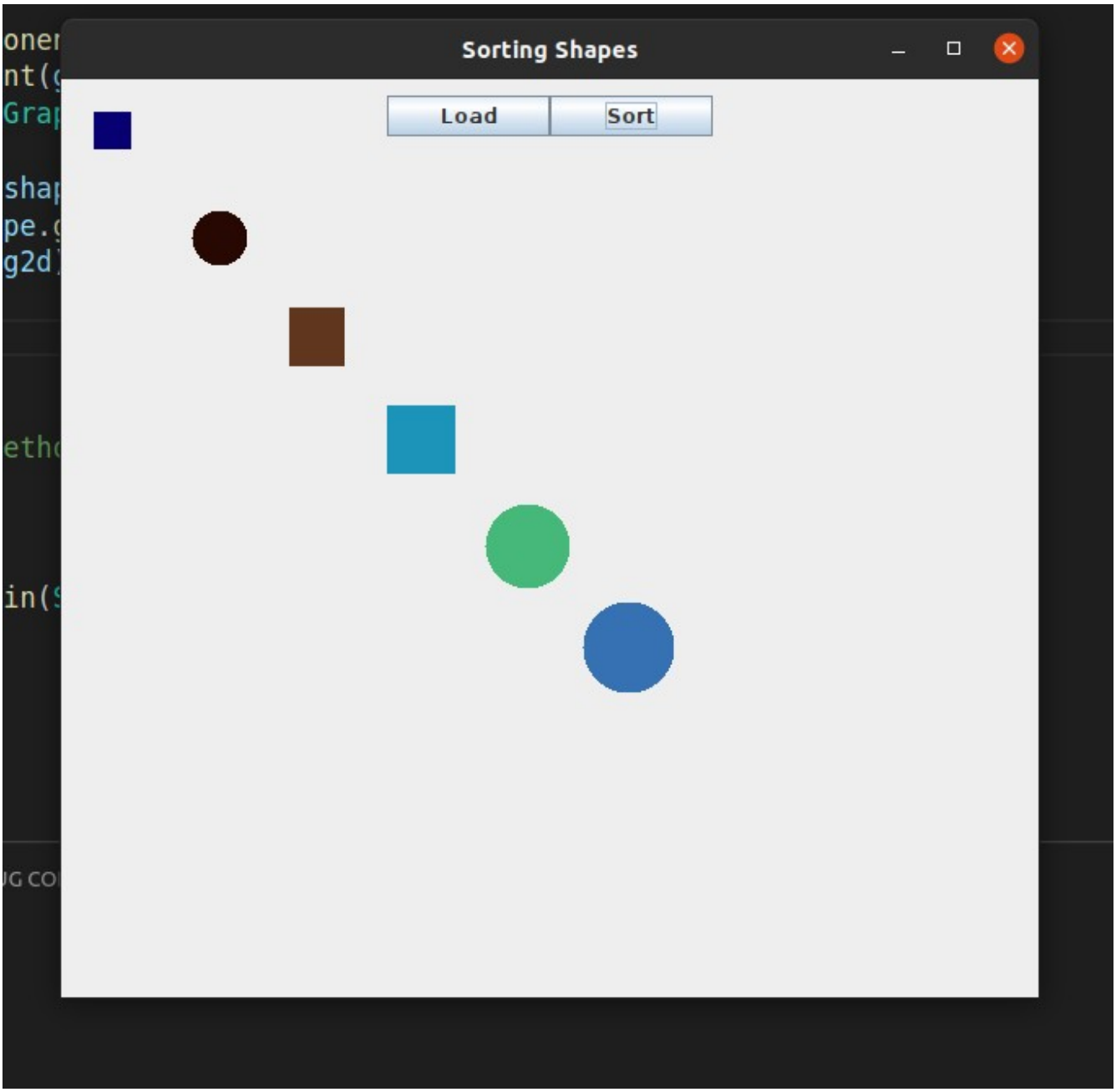
Before execution:



Load button function:



Sort Button function:



Part 4: Conclusion.

Having the function of the interface and object work together went very well. While following the 2 design patters the code was very organized and easy to follow.

what went wrong with the project was working with swing. There was alot of debugging done when implementing the buttons.

I have now learned to properly approach a new project. After having a clear design of the project, the implementation becomes much more easier.

To ease:

1) try to figure out which design pattern would suit the project most.
2) create the UML class diagrams first before implementing.
3) use programming practices that best organize your code. Organized code speeds up implementation.