**NAME- SANIZ S MOMIN STUDENT NO- 217584749**

**EECS-3311 A LAB 3 TA NAME - KAZI MRIDUL**

**INTRODUCTION:**

The software project is about generating random six shapes and sorting them according to their surface area. The functionality to load the shapes randomly with random colors and sizes and sorting them is implemented through two buttons. The first button with the name **“Load Shapes”** has the functionality to load six shapes namely a Circle, Square, Rectangle, Oval, Semi-Circle, YRectangle (or vertical rectangle). The sizes of these shapes are random with a threshold set which is the range in which the size of a particular shapes can be maximum or minimum. The second button with the name **“Sort Shapes”** has the functionality to sort all of the six shapes according to the surface area they cover. A sorting method technique is used to sort these shapes, so it used Insertion Sort to sort them. The main goal of this software project is to use Object Oriented concepts and effective software design patterns to write code efficiently and follow the DRY principle which means (Don’t repeat yourself) by using design pattern such as Prototype pattern. Also, the sub-goal is to make best use of these design pattern and OOP concepts to generate six shapes of random sizes and sorting them based on their sizes. One challenge I faced was to change the positions of this shapes, insertion sort worked but to position these shapes I had to store the fixed position and call JPanel ‘s paint method to re-position these shapes when sorted. The design pattern I used is Prototype pattern where there is a generic abstract class Shape which contains fields, declared and defined methods. Classes such as Circle, Square, Rectangle and three other shapes are children of this Shape abstract class. Also, singleton design pattern is used to hide the instantiation part, so a static factory method is used for Instantiation of these specific shapes. Object oriented concepts such as Abstraction, polymorphism, encapsulation is also used. The report is going to be structured in a way in which it will have a flow which is similar to how the project started from scratch to its completion every design pattern, challenges, ideas will be explained.

**A picture containing text

Description automatically generated1st UML DIAGRAM**

**A screenshot of a computer

Description automatically generated with low confidence2nd UML DIAGRAM**

**Why First UML diagram is better than the Second UML diagram?**

I still believe that the first UML diagram is a better design because it is a mixture of factory pattern and prototype pattern. The good thing about it is extending the classes to the abstract shape class. Because of this combination not only I am able to hide the implementation of my logic that is I am fulfilling the role of factory design by hiding the object creation from the user. Using prototypal design pattern I am keeping performance in mind that is other shape classes don’t have code repetition that if I used the second UML diagram as my idea than I would have all the shapes that is circle, rectangle, square, oval, YRectangle, SemiCircle code repetition where each of this class would have same name of the fields and definition of methods in all the classes which to me is a lot of code repetition and it is against the DRY rule which is not to repeat yourself. Using the first UML Diagram pattern I avoid that code repetition and the code of my classes looks clean and readable.

**OO Design Principles Used:**

**Abstraction:** In this software project, some unnecessary details are hidden from other classes. I will give you examples where abstraction is used. Take a look at the shape class where random generation of RGB colors is hidden from its children such Circle, Rectangle, Square and three other shape classes. Another instance where abstraction will be seen is when using the ShapeFactory class to generate shapes. The way object is created is completely hidden what is done is a string is passed such as “Circle”, “Rectangle”, “Square” and object is returned.

**Encapsulation:** In this software project, you will see the concept of encapsulation used where some logical implementation required for some classes to give required functionality is set to private so that other classes cannot access it. For example in the UserInterface class LoadShapes() and CreatePanel() are set to private those methods are only needed for the proper implementation of the UserInterface class.

**Inheritance:** The concept of Inheritance is one of the most required principle used in this software project to satisfy the DRY rule which means Don’t repeat yourself. The shape class the parent class and its children Circle, Rectangle, Square, Oval, SemiCircle, YRectangle (vertical Rectangle) extends the Shape class this way our code repetition is reduced significantly. No child class even needed their own fields all they have is methods implemented which they inherited from Shape class.

**Polymorphism:** In this software project the concept of polymorphism is also used. You will see in the ShapeFactory class that the shapes are instantiated with the syntax Shape circle = shapeFactory.getShape(“Circle”) which upon decoding means Shape circle = new Circle(); Polymorphism means “many forms” therefore though the object created is of the Shape but the methods takes form according to the new SpecificShape() where (SpecificShape- Circle, Rectangle, Square) and whenever any method is invoked such as circle.calcArea() it uses the implementation of that specificShape class though the instant type is Shape this happens because of inheritance where it is considering the latest form of the method. Therefore if it is Shape rec = new Rectangle() and if rec.calcArea() is invoked this will call the latest implementation of calcArea() which is implemented in the Rectangle class

**Sorting Technique Used:**

In this software project, the sorting technique that is used is Insertion Sort. Insertion sort is implemented by creating a class InsertionSort which takes parameter such as localeX array which keeps record of the X-axis positions, localeY which keeps record of the Y-axis positions. An ArrayList of shape which contains all the six shapes. An array of colors which keeps record of colors of sorted shapes. The insertion sort is conducted using the calcArea() method in each of the shapes. So each shape that is Circle, Rectangle, Square, SemiCircle, YRectangle, Oval has a method calcArea() which is inherited from the shape class. So each of this shape has a area Formula which is used in the calcArea() method to calculate area of that particular shape. The area is calculated with the help of the getWidth() and getHeight() method in the shape class which these six shapes inherit. Now Insertion sort has two loops in which the first loop iterates the whole array and second loop iterates in the opposite direction such that while iterating in the oppositing direction if there exist a shape whose area is larger than its next shape than their positions in the Shape ArrayList is changed. Because of aliasing I can directly manipulate the Shape ArrayList in the userInterface class by passing it as parameter in the InsertionSort class and by not creating a shallow or deep copy this allows me to sort the ArrayList of shapes. In this project the InsertionSort is applied through the applied() method defined in the InsertionSort class.

**Implementation of Classes:**

* Firstly there is this abstract class which is named as Shape. This class contains fields like localeX – point on X-axis , localeY – point on Y-axis, width – width of the shape, height – height of the shape. RGBGenerator – A class that generates random RGB values, Color – to set the color value. It contains some implemented and unimplemented methods which are to be used by its children. Getters such as getWidth(), getHeight(), getLocaleX(), getLocaleY() and other utility methods such as calcArea(), draw( Graphics, Color) – this method is the backbone for each of its children because it is this method due to which each shape gets its color and outline without this you cannot see the shapes in this project.
* I also Created a shapeFactory class which uses the Prototype software design pattern to hide the implementation of how these shapes are instantiated and created. So this class basically contains only one method which is getShape(String) and it takes a string value of each shape and using an if-else if statement it determines which shape is requested and which shape object must be instantiated and returned and making sure the implementation of instantiating is hidden. The random sizes of the shapes are determined through this method using a custom RandomNumGenerator class which returns random number between a specific range as per the demand. The instantiation of these shape objects are hidden too by using the static factory method and the design pattern that is used is Singleton design pattern where the constructors of the six shapes are private and the instances of the object are returned through the static getInstance() method defined in each class.
* These six shape classes which are namely Circle, Rectangle, SemiCircle, Oval, YRectangle has only 3 methods implemented mainly calcArea(), draw(Graphics, Color), static getInstance() and these classes have their private constructor defined.
* In this software project I have implemented first class Diagram design pattern. The Prototype design pattern enables instantiating these shapes without using the new keyword in the main UserInterface. Instead it is made more simple, readable and easy to understand for the coder. All it takes is to instantiate this shapeFactory class which uses Prototype design pattern and it just takes a string as parameter to instantiate any of the six shapes which matches the String that is passed for example if string “circle” is passed what you will get in return is a shape object which is circle. Also the instantiation of these shapes is hidden one level down for the getShape(String) Method. Singleton design pattern is used to hide the instantiation of these shapes so each of this six shape classes have a static factory method which return the instance of the class for example Circle class , to get its instance you have to write Circle.getInstance() and it is impossible to get the instance like new Cirlce() because the constructors of Circle as well as other shapes are private.
* In this project I have used eclipse as a tool and the version of JDK I used is 14, another amazing tool I used to was WindowBuilder which makes your life really easy where creating a GUI.

Chart

Description automatically generated with low confidence

Chart

Description automatically generated

Random sizes and random color shapes are generated each time when the load button is pressed and on pressing the sort button it sorts the shapes based on their surface area.

**Conclusion:**

* The good thing that went well in this project is implementing the design pattern and planning of the software design before writing the code. I feel more experienced after creating this project. I learned by making plan on a piece of paper firstly and than I implemented gradually.
* One of the challenge or a thing that went wrong in the software project was putting the shapes in the sorted position. My arraylist of shapes was sorted properly by using insertion sort but the main issue I faced was displaying it on the jpanel I was not able to do it initially but than I figured out a technique to display them in the sorted order.
* The things that I learned through the journey of creating this software project is initial planning, if it’s a project which requires lot of demand to be fulfilled. I used divde and rule principle where I breakdown the demands of this project into subpart and tried to find what should be the role of that subpart in the project how many classes will be required and is there a way to implement code reusability by using OOP Principles.
* The top three recommendation I believe necessary for this software project is:-
* Proper Planning – without initial planning and starting the project thinking that you will be able to find ways while making the project is a bad practice. Initial thoughts and ideas on the project building are a must and later while implementing you can make changes as and when needed.
* OO Concepts Implementation – concept such as abstraction, polymorphism, inheritance and encapsulation should be used in the project using these concepts you will automatically write code efficiently without much repetition.
* Software design patterns – hiding your logical implementation and writing clean code is considered a professional quality in a programmer. Implementing software techniques such as Prototype pattern and singleton pattern can lead to writing code which hides your logic and shows more readable code in simple English which simplifies things significantly when you have 1000 of lines of code.