**Part 1: Introduction**

This software project is about designing a system to solve a problem using design patterns and OOD principles. Its goals are to generate and display different shapes of different sizes and color and sort them and display the sorted shapes on an interface. The concepts used include object-oriented design (OOD), its principles and design patterns. OOD is a programming paradigm supported by several object-oriented programming languages such as Java and C++. Main principles of OOD include abstraction, encapsulation, polymorphism, and inheritance. All of these principles are used in the implementation of software design. The design pattern used in this software project is singleton where the class will have a maximum of one instance which offers a global access point and allows users to see the instance. This report will be structured in 4 parts, the first of which is this one, known as introduction. It will be followed by design, implementation, and conclusion as its remaining components.

**Part 2: Design**

For the project, I created a UML diagram for the design process. The screenshot of the diagram is attached below:

Diagram

Description automatically generated

The diagram includes:

* The circle class, which represents a circle.
* The rectangle class, which represents a rectangle.
* The square class, which represents a square. The square is the same as the rectangle, but the height and width are the same.
* The generic shape class, which defines a general shape.
* The sorting technique class, which used a selection sort algorithm to sort shapes based on their surface areas.
* The shape factory class, which generates a number of different shapes with various sizes and colors.
* My panel class, to display the shapes on an interface.

Diagram, schematic

Description automatically generatedDesign principles used in this UML diagram are inheritance and aggregation. The classes for circle, square, and rectangle inherit the generic shape class. The classes for sorting, generating shapes, and displaying shapes have a 1-1 aggregation relationship. The shape class has a many-1 aggregation relationship with sorting, generating shapes, and displaying shapes classes. I also created a second UML diagram for a design alternative. A screenshot of the second UML diagram is attached below:

This diagram does not yield a better design than my first UML diagram because it does not contain the sorting class and does the sorting in the panel class which is specified for displaying the shapes, hence my first UML diagram is a better diagram, and I will be implementing that with my java code.

**Part 3: Implementation**

The sorting technique I used to sort the shapes was selection sort. The way it works is it finds the smallest element in a list and moves it to the front. Now the first element is sorted, and the rest of the list is unsorted. Then we take the unsorted list and look for the smallest element and move it to the second position and so the first two elements are now sorted. We repeat this process until all the elements in the list are sorted and so we get a sorted list. In my implementation, the sortShapes method iterates through the shapes in the list, sets the first element’s surface area as the smallest and then iterates through the rest of the list and if it finds an elements with a smaller surface area it shifts that to the front. This process is repeated through the rest of the list, and at the end we get a sorted list and that is returned.

For my system, I followed my first UML diagram as it was the better design that the other. The first thing I create is the shape class, it contains all the variables common between all shapes like x and y coordinates, width, and color. It also contains abstract methods such as surface and draw, which are to be implemented by each of the shape classes. It also contains the compare method which compares two shapes and determines which one of the two has a smaller surface area. Additionally, the shape class contains the getters and setters for the common variables. These include getters and setters for color, x and y coordinates, and width. Then I create individual classes for rectangle, square, and circle. I implement the abstract methods of the shape class like surface and draw. For the rectangle class I include the height parameter unique to rectangle, as its height is not the same as its width. Subsequently, a getter and setter for height is also created in the rectangle class. The circle and square classes only have the implementation of the abstract methods surface and draw. I then create a factory class that creates a number of random shape objects with random colors and stores them in a list. Then, the sorting class is made which sorts the list of shapes generated in the factory class using a selection sort algorithm. The last class is the panel class which creates and implements the buttons for creating and sorting shapes and displays the unsorted and sorted list of shapes accordingly.

The tool I used for the implementation of my design is an IDE called eclipse. I created a package and had all my classes under it to ensure they are at one place and eclipse allows me to keep them at one place using a package which is a good feature. Below is a screenshot of execution of the code:

Chart, waterfall chart

Description automatically generated

**Part 4: Conclusions**

The thing that went well in my project were that I was able to create a good UML diagram using my design and then I was able to convert the diagram well in code using an IDE called eclipse. The things that didn’t work well was that I was not able to display the sorting algorithm on my panel and had difficulty doing that even though the algorithm itself was correct and did what it was supposed to do. The things I have learned in this software project are how to use a good UML tool like disgrams.net and how to create different classes for one system and have them interact in a way that is appropriate for the problem that needs to be solved by the system. My top three recommendations to ease the completion of the software project are to provide some sort of source code that will make it easier for students to learn as they will have something to add to and they wont have to make it from scratch. I would also suggest starting this project from lab 1 and divide it into parts that can be completed at earlier stages. This would give students the exposure from the beginning, and they will get a better understanding of the concepts involved. Lastly, I would suggest defining and explain a overall structure of what should be done at each stage, that will give students the chance to retrace their steps and figure out what’s wrong if they are lost or stuck at any given stage.