Michael Victorino
216864621
EECS 3311 Section A
Display Shapes software project
Naeiji Alireza

Display Shapes Software Project Report

Part I: Introduction

Goals of Display Shapes

- Display a window application onto the screen that should be 600 in width and height.
- The graphic user interface should display two buttons at the top, “Load Shapes” and “Sort Shapes”
- After pressing the “Load Shapes” button, the application should display six shapes onto the screen on without superimposing over any other shapes
- After pressing the “Sort Shapes” button, the application should display the six shapes already present on the screen in the correct order of their surface area

Challenges of the Project

- How to display shapes on the screen
- How to make the buttons work to generate/sort the shapes
- Implementing the sorting algorithm for the shapes
- Connecting the panel and shapes through abstraction

Concepts Used

- Objected Oriented Design
    - Abstraction: Hides implementation from the user
    - Encapsulation: Hides information from the user
    - Inheritance: Class with the same properties from a base class
- Design patterns
    - Factory: Creates specific objects from user input and returns it
    - Singleton: Ensures single instance of an object

Structure of the Report

The structure for the rest of the report is as follows

- Part II(Design) Pages 2-4
- Part III(Implementation) Pages 5-7
- Part IV(Conclusion) Page 8

Michael Victorino
216864621
EECS 3311 Section A
Display Shapes software project
Naeiji Alireza

Part II: Design

Application Sequence

The “Display Shapes” application should follow a sequence of:

- Application opens and displays 600x600 window
- Two buttons display on top, one being “Load Shapes” and other “Sort Shapes”
- Pressing “Load Shapes” draws six different shapes of varying sizes and colours on to the application
- Pressing “Sort Shapes” sorts the shapes(if any) that are currently displayed in order of their surface area

This sequence is show through Figure 1

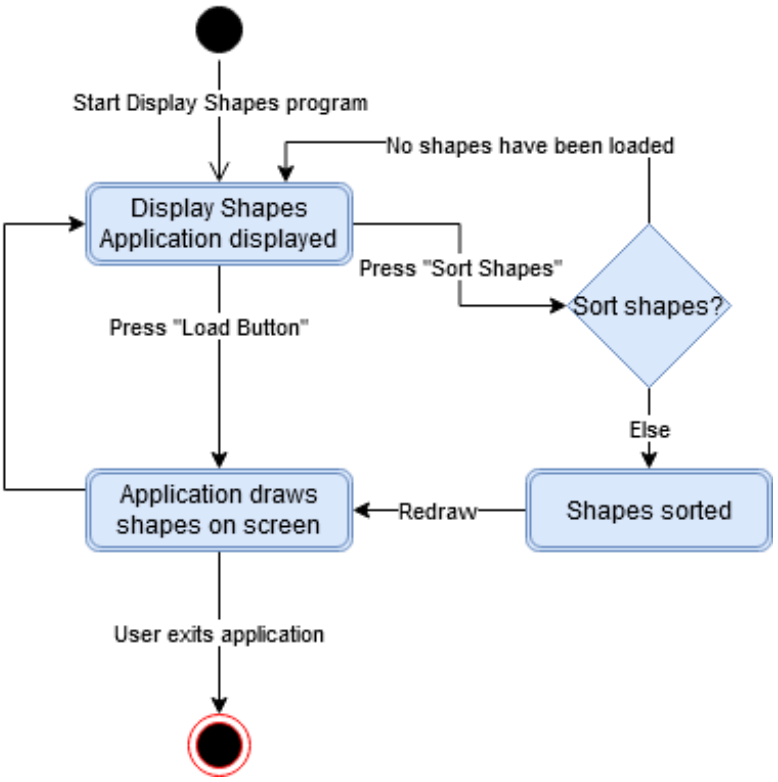
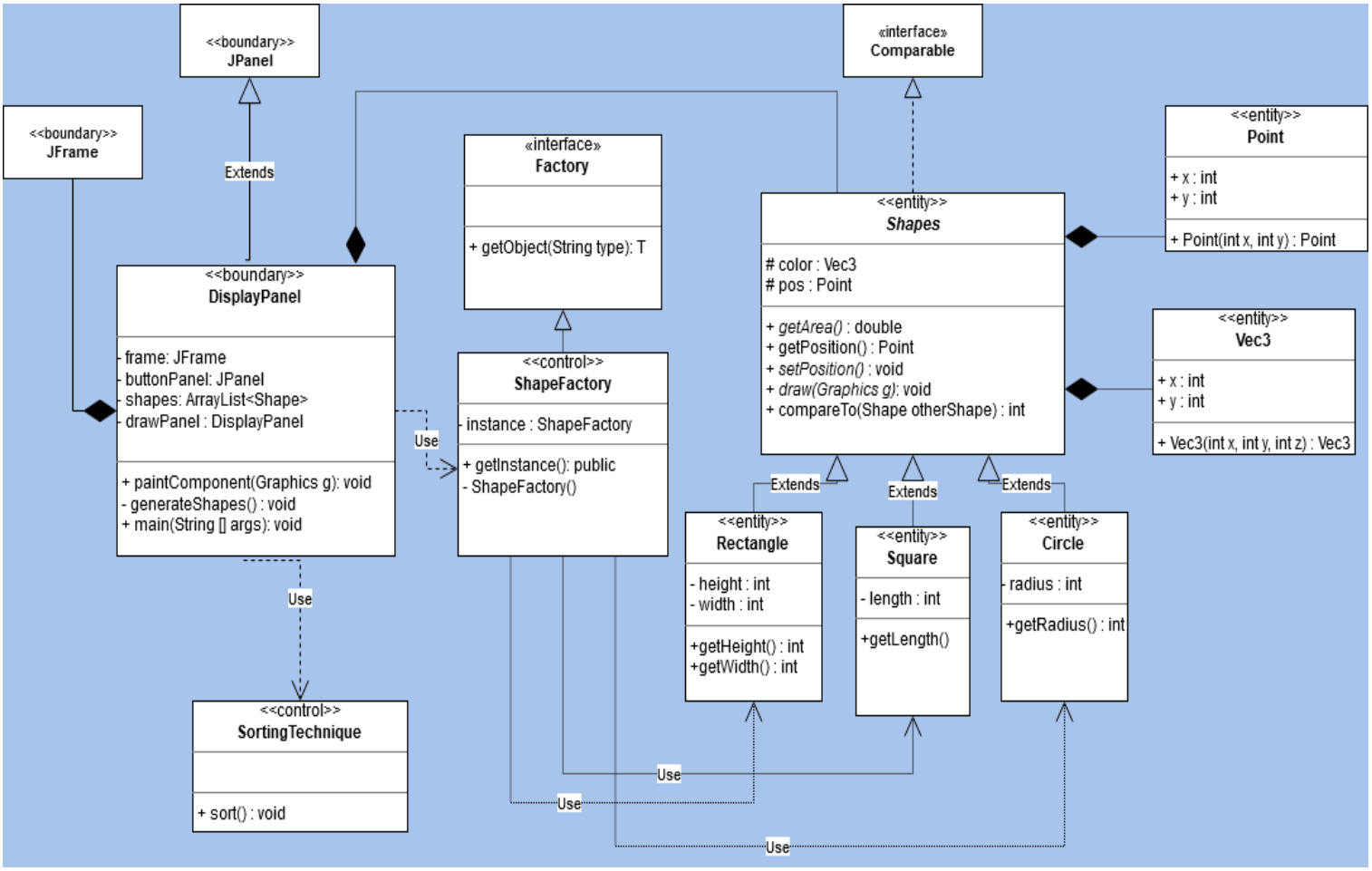


Figure 1: Expected sequence of “Display Shapes”

Michael Victorino
216864621
EECS 3311 Section A
Display Shapes software project
Naeiji Alireza

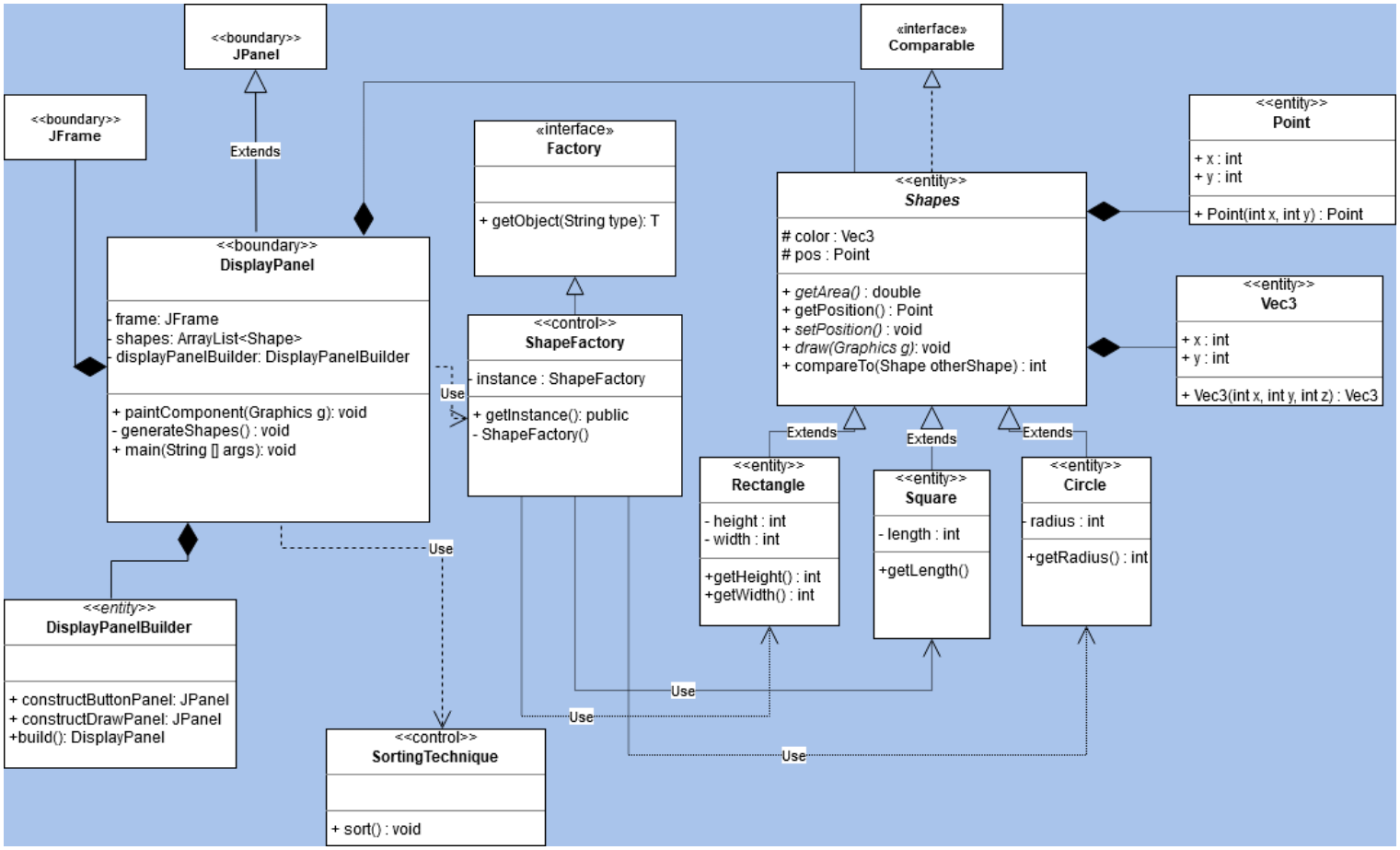
Design Diagram 1



- UML diagram of the design of Display Shapes:
  - Black lines with diamond shows composition
  - Hollow arrow shows inheritance from a parent class
  - Dotted lines or with “Use” subtitle are dependent on a class
- Object-oriented design:
  - Abstraction with abstract classes that provide the required functions and hides the implementation from the user(Factory interface, Shapes abstract class)
  - Encapsulation with classes hiding logic inside classes(users have no direct access to components inside Shape classes)
  - Polymorphism which allows a class to implement its own function from a base class(Rectangle,Square, ShapeFactory)
  - Inheritance which has classes from a parent class with the same functions(Rectangle,Square,Circle)

Michael Victorino
216864621
EECS 3311 Section A
Display Shapes software project
Naeiji Alireza

Design Diagram 2



- Alternative UML design diagram of Display Shapes application
- The big difference between the first design diagram and this design diagram is the DisplayPanelBuilder class
    - DisplayPanelBuilder is based on the Builder design pattern
    - The class builds the individual components of DisplayPanel such as the button panel and draw panel
- The advantage of this design over the first one would be scalability if we wanted to make multiple custom panels and we require multiple types of builders to construct different panels

Michael Victorino
216864621
EECS 3311 Section A
Display Shapes software project
Naeiji Alireza

Part III: Implementation

Sorting Implementation

The sorting algorithm I used for the sorting implementation was **Bubble Sort**. There were other algorithms I researched to decide which sorting algorithm to choose such as Insertion Sort, Merge Sort, Quick Sort. Whilst Insertion Sort was another great choice, I decided on bubble sort for a few good reasons

- Needed a simple and easy to implement sorting algorithm
- Data set to be sorted is small enough where performance does not matter and/or are not particularly concerned about performance

A short explanation on bubble sort:

- Compares adjacent elements in list if they are in the incorrect position
- If the current element has higher priority than the next element, elements are swapped
- Largest elements gets put last each pass, and repeats until list is fully sorted

Design Implementation

For the implementation, I chose to use Design Diagram 1 to implement the Display Shapes application. The coding implementation for this diagram is fairly straightforward:

- The program starts from DisplayPanel, a JPanel implemented for the requirements to draw shapes into the screen
- DisplayPanel will contain functions to generate shapes, and will contain the main function to start the program
- Main function builds up the JFrame, the button panel, the button action listeners for when the user presses the button
- When user presses the “Load Shapes” button, we invoke generateShapes() function that gets the singleton instance of ShapeFactory based on the factory design pattern
- ShapeFactory returns a shape(Rectangle, Circle, Square) to DisplayPanel and the DisplayPanel will refresh to display the shapes
- The shapes are generated with a random colour and size(depending on the type of shapes)
- When the user presses the “Sort Shapes” button, it will invoke the SortingTechnique class that will sort the list of shapes given through the sort(List<Shapes) function
- DisplayPanel will refresh to display the re-sorted shapes

Michael Victorino
216864621
EECS 3311 Section A
Display Shapes software project
Naeiji Alireza

Tools Used

- IDE used is IntelliJ IDEA 2019.3.5(Ultimate Edition)
- Project SDK 11(java version “11.0.6”)
- Version control used is Git
- Hosted on GitHub
- Google Docs for Documentation for the report

Interface Snapshots

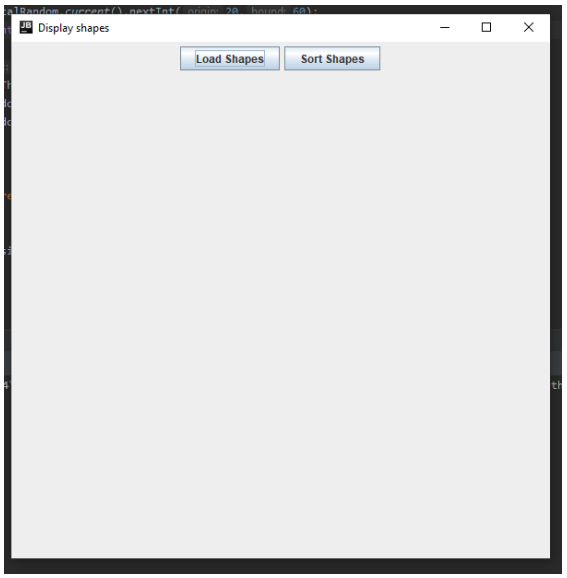


Figure 4: Display Shapes application when user first opens it which only shows button interface

Michael Victorino
216864621
EECS 3311 Section A
Display Shapes software project
Naeiji Alireza

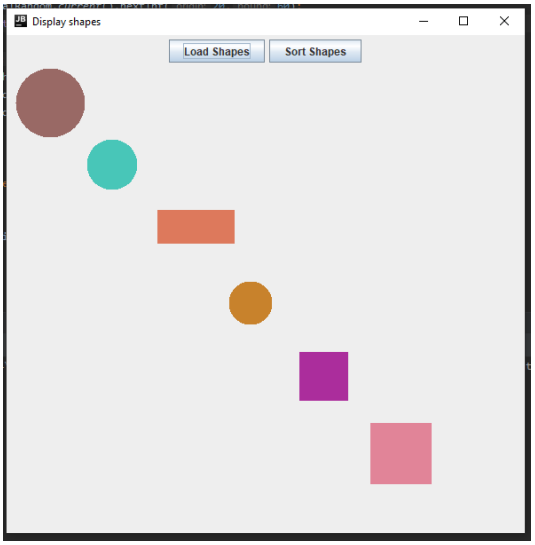


Figure 5: Display Shapes interface after user presses “Load Shapes” button displaying 6 random shapes in the application

Figure 6: Display Shapes interface after user presses “Sort Shapes” button and rearranges positioning of shapes from Figure 5 to the smallest to largest surface area

Video on Display Shapes

- Shows how to open Display Shapes project through Linux and Windows(Eclipse and IntelliJ)

https://youtu.be/PS7I57f2sN0

Michael Victorino
216864621
EECS 3311 Section A
Display Shapes software project
Naeiji Alireza

Part IV: Conclusion

What Went Well

- Implementation was smooth and fast
- Coding the design patterns required was easy as well since I had prior knowledge of the design patterns used(Singleton and Factory)
- Coding the sorting technique was easy as well since I had prior knowledge of the sorting algorithm(Bubble Sort)
- Keeping in mind of object-oriented design went well as I have plenty of experience with object-oriented programing before

What Went Wrong

- Wanted to implement the Builder design pattern as I never have implemented it to a project before but did not want to ruin the quality of the project
- Little design decisions such as wanting Shape classes be an extension of JComponent but reading up on it and realized it was not a recommended by experienced Java programmers
- Required better unit testing for all classes

Learning from the Project

- Improved usage of design patterns
- Retained knowledge of object-oriented design and programming
- Improved documenting of software projects

Recommendations for Next Project

- Unit testing is important
- Git or version control is important for individual,group and company projects
- Planning your software ahead of time eases implementation of the software