Drawing and Painting application

Report

Prepared by:

ID	Name
6845	Salma Mohamed Hasan
7161	Youmna Ahmed Ali Ibrahim
6654	Alaa Hussein Salem
6974	Nour Hamdy

Algorithm

- **Interface package:**

1. Class GUI:

It's the interface of the project; contains the buttons and the panel board class. Each button will call the mode variable with a certain number to perform its function.

2.Class Panel :

We applied inheritance as we extended the JPanel frame and implemented MouseListener, MouseMotionListener events.

Two overridden methods were used from the implemented events: mouseDragged and mousePressed.

Another method used which is the paintComponent it's called when we use repaint() overridden method.

2.a. mousePressed() :

Variable mode is used to determine which shape or event is going to be used. If mode =0; call the line class and add it to the array list of shapes then call repaint method to draw the shape, if mode =1; call the rectangle class and add it to the array list of shapes then call repaint method to draw the first point, if mode =2; call the circle class and add it to the array list of shapes then call repaint method to draw the first point, if mode = 3; call the square class and add it to the array list of shapes then call repaint method to draw the first point, if mode =4; call the triangle class and add it to the array list of shapes then call repaint method to draw the first point, if mode =5; select the shape, if mode =7; delete the selected shape, if mode=9; copy the selected shape.

2.b. mouseDragged;

Same mode variable used in mousePressed method with the same sequence to add the second point and the third point for the triangle and the second point for the other shapes, as you drag the mouse the shape is formed. More conditions were used; if mode = 6; will resize the selected shape, if mode = 8; will move the selected shape.

- ## Classes Package:

It contains the shapes classes; Line, Rectangle, Square, Circle, Triangle, and an abstract class called Shapes is extended in the other classes of that package.

Each class has private variables, setters and getters and 3 methods; draw(), contains() and clone().

1. draw() method:

It sets the color and uses conditions to draw the shape in all cases; up to down, the opposite, etc.

2.contains() method;

Overriden method, used to know whether the point is inside the shape or not to select the shape.

3. clone() method:

Overriden from Cloneable interface to make another copy of that object.

- ## Functions package:

1.Move class:

It’s used to move the selected shapes. It contains five methods; moveRect(), moveLine(), moveCircle(), moveSquare(), moveTriangle().

They set new coordinates (x,y) for the selected shapes.

2.Resize class:

It’s used to resize the selected shapes. It contains five methods; resizeRect(), resizeLine(), resizeCircle(), resizeSquare(), resizeTri().

They use if conditions depending on the point we want to resize to it; point is nearer to left side, point is nearer to upper side, point is nearer to down side, point is nearer to right side.

OOP Concepts:

1.Abstraction:

In the abstract class called Shapes, to reduce complexity so when it is extended in other classes, the method inside it can be accessed.

2.Inheritance:

Extended Shapes abstract class, Cloneable interface, JPanel and implemented the MouseListener and MouseMotionListener interfaces.

3.Polymerphism:

//
//
///////////

4.Encapsulation:

In Classes package we used in each class setters and getters for the private variables.

Design Patterns

1.Factory:

Factory class is used to make objects needed in the project in one class then be called whenever an object is needed to be created. It's implemented in panel class when creating new objects for the shapes.

2.Iterator:

Instead of using the ordinary for loop to get what is inside the array list, we used Iterator and checked whether it hasNext() elements in the array lists or not then do the specific code written then next() to get the next element in the array list.

3.Prototype:

Implementing Cloneable interface and override its method clone to make a copy of the certain object needed. It copies all its content of the same type integer.

4.Singleton:

Used in GUI class as we don't need to create objects for it in the project, it's used once, so we made a private constructor and made getInstance() method, to create only one object for it whenever it's called.

5. Decorator:

//

Solid Principles

1.Single responsibility:

Each class should be responsible for a single functionality for example class factory, it's used to only create new objects, and all the classes in the Classes package, are responsible for creating the specified shape, similarly in the classes of the rest of the project.

2.Open-closed principle:

The Software components are open for extension, but not for modification. If we want to add new shape, we will just create a new class and extends from Shapes class as the Line, Rectangle,

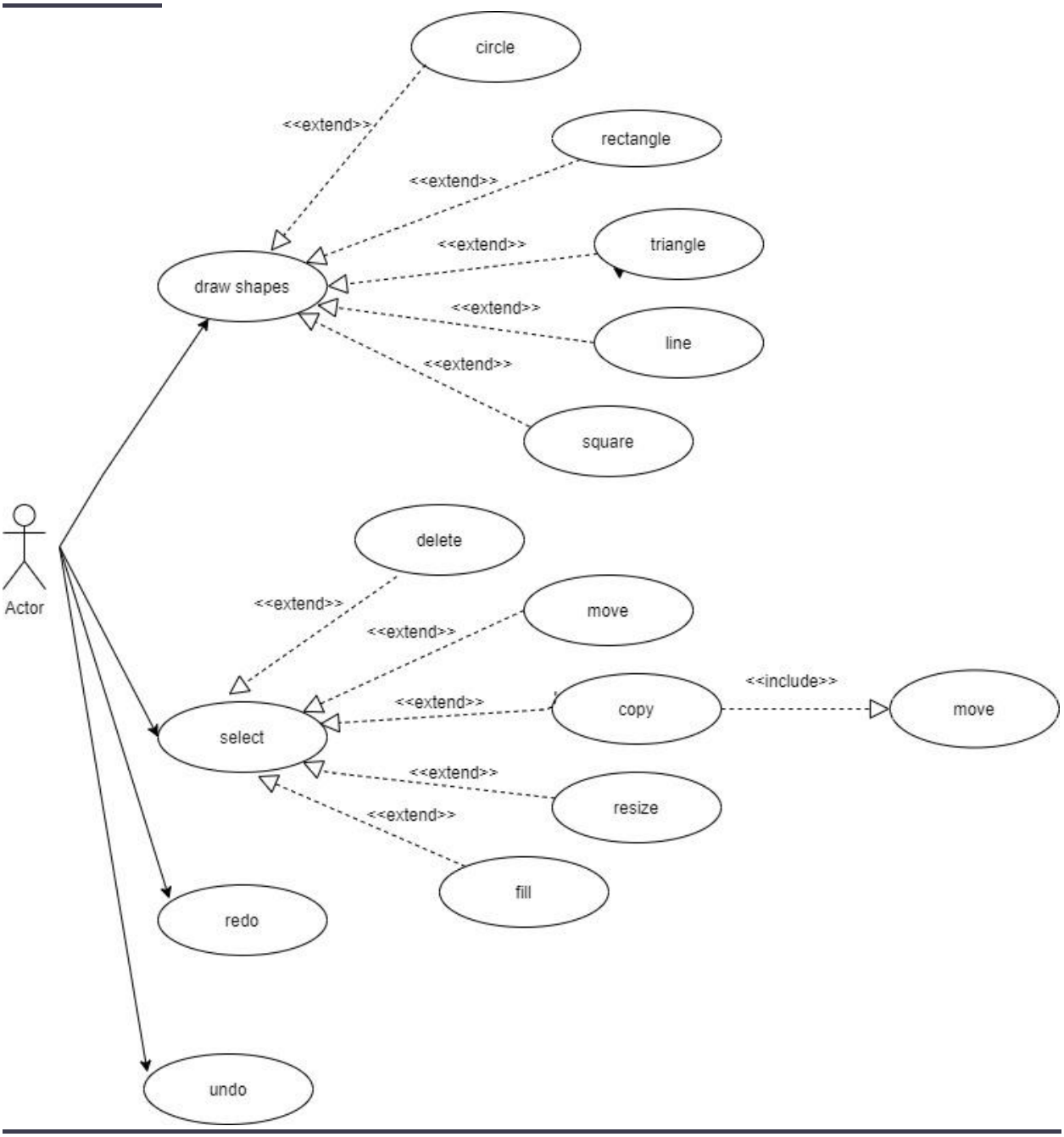
Square, Triangle and Circle classes. Any new shape could be added.

3. Dependency inversion principle:

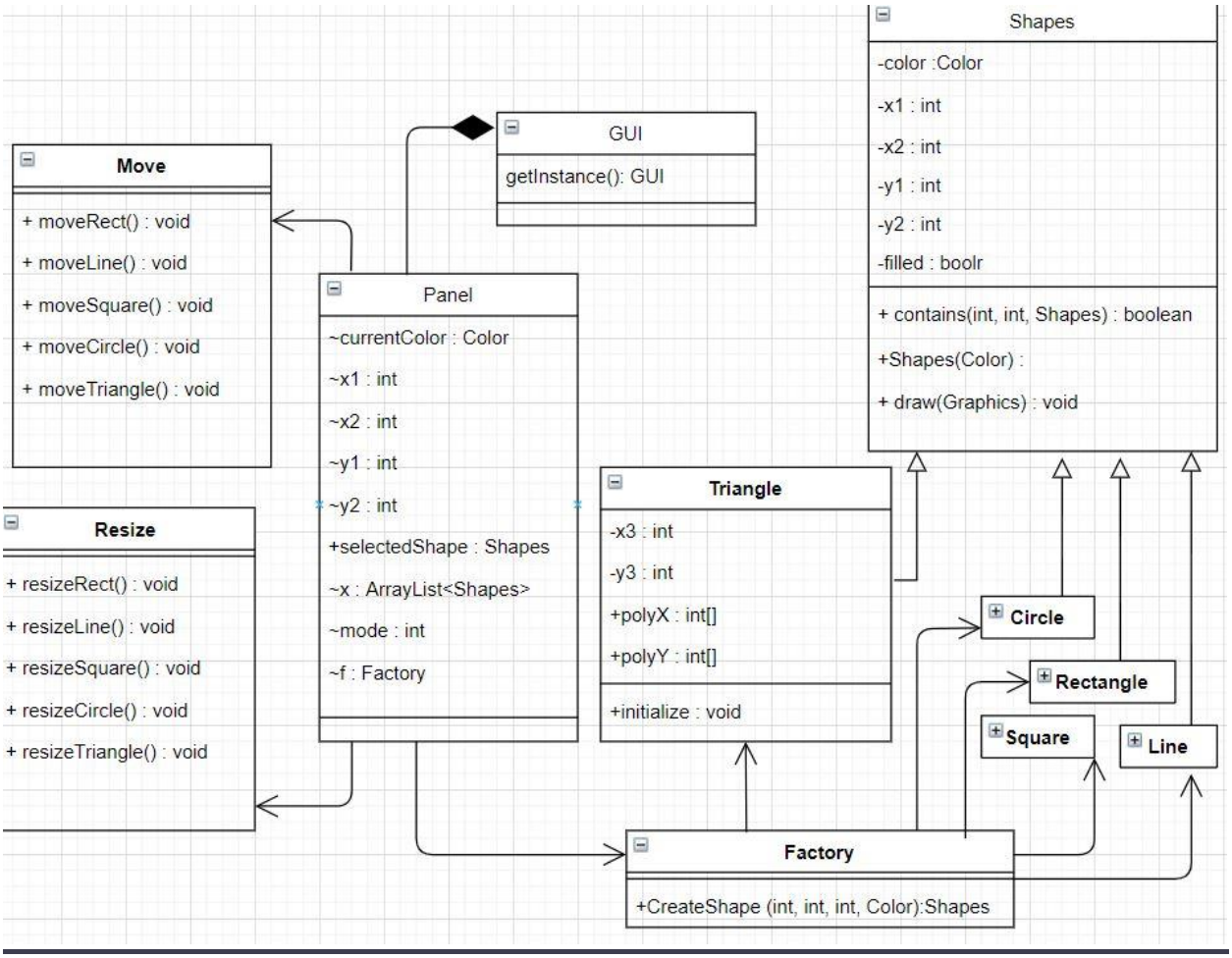
The implemented methods in the Shapes abstract class, should not be all used in the classes that extends shapes.

UML diagrams

1. Use case:

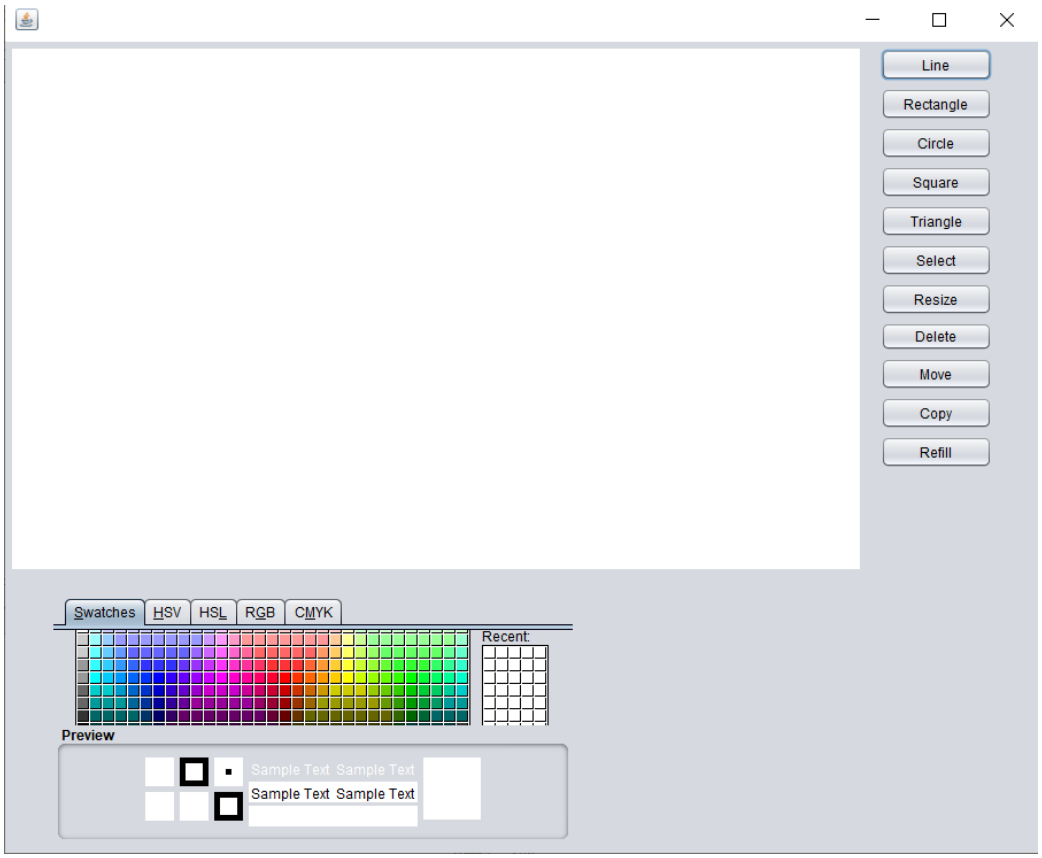


2. Class Diagram:



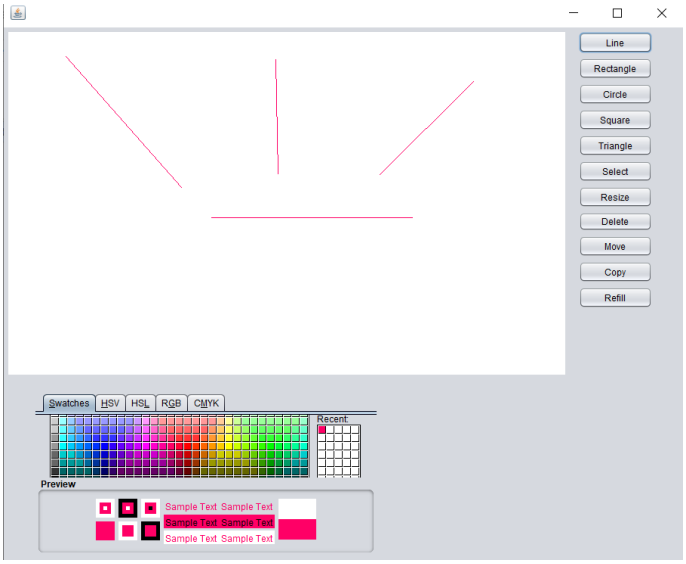
User manual

Once you run the application, an interface will appear with an empty space to draw in and the buttons of the operations used.

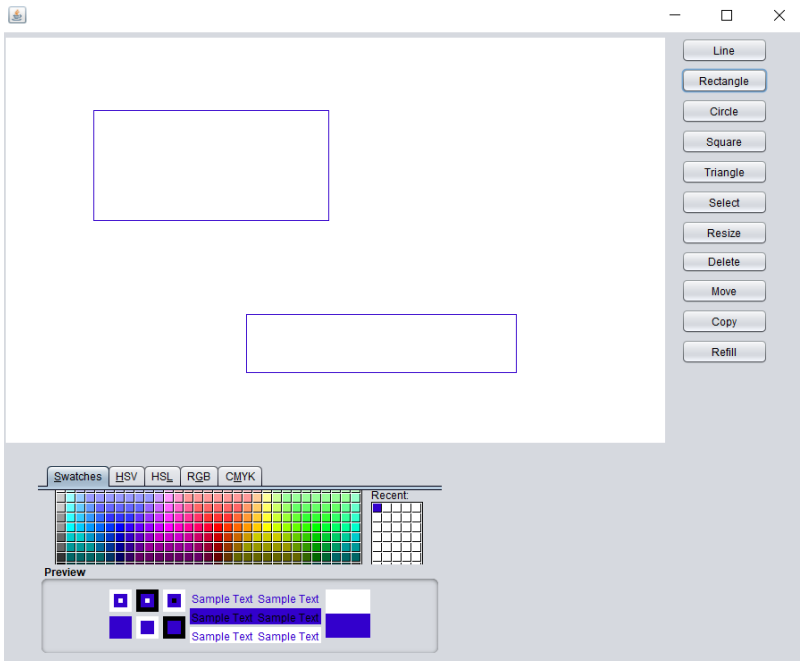


If you want to draw line, circle, rectangle, square and triangle; all you need is to press on the color you choose first then on the button of the specific shape you want to draw.

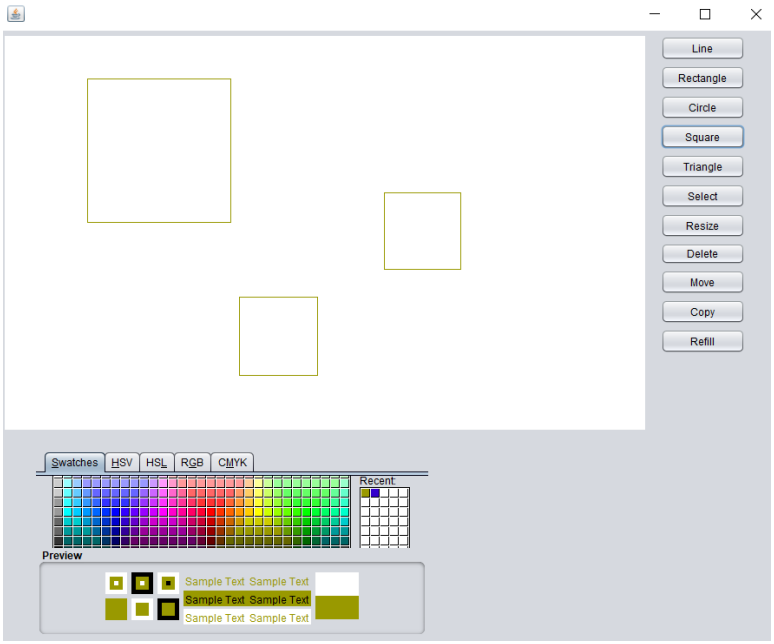
-Drawing line:



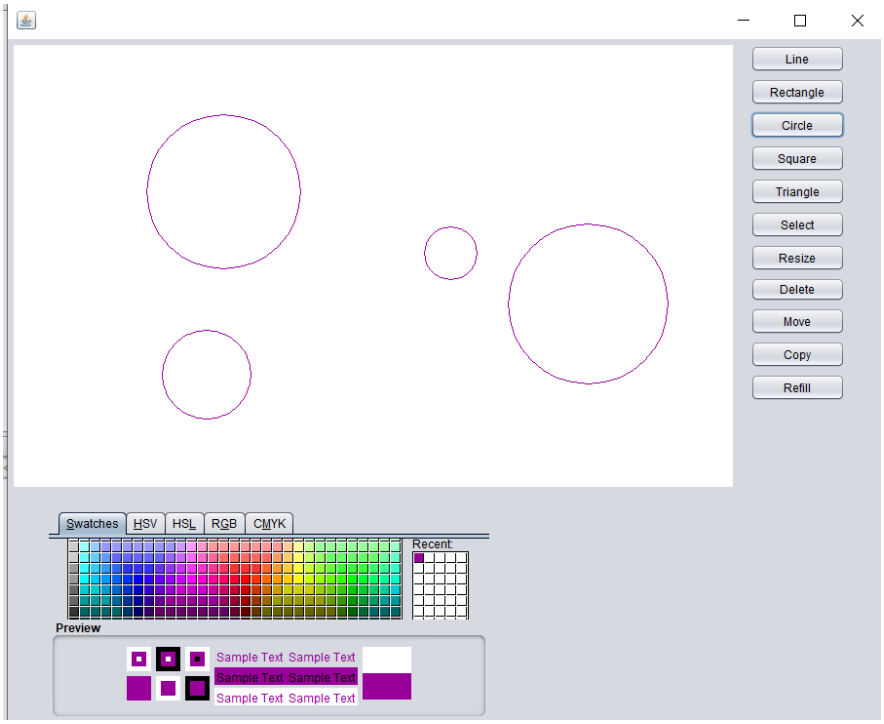
-Drawing rectangle:



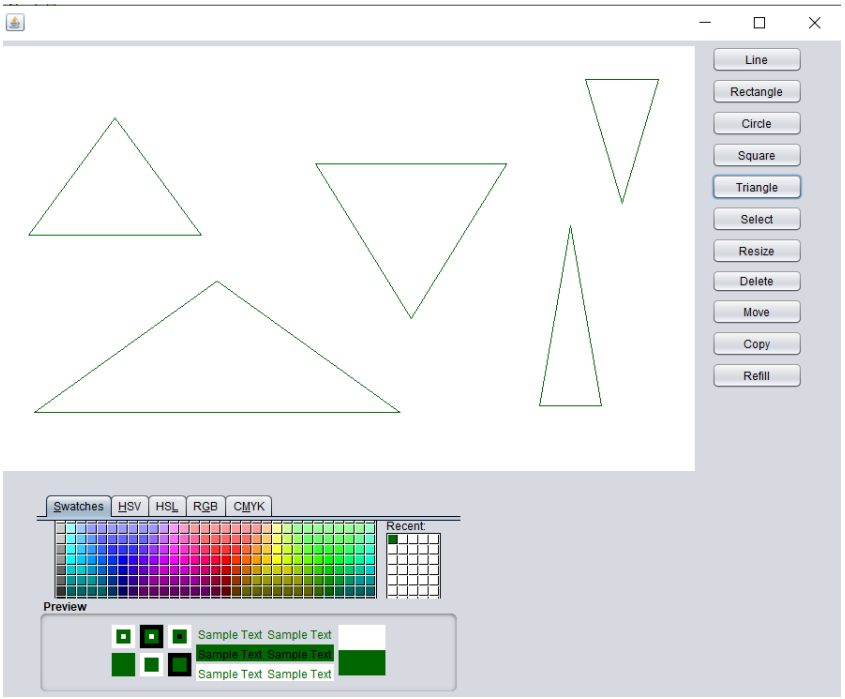
-Drawing square:



-Drawing circle:



-Drawing triangle:



If you want to select a specific shape, press on the select button.

For resizing you must select the shape first then click on the resizing button and resize the selected shape you want; triangle after resizing: