Programming Final Project

Paint

1) Yasmin Ashraf Mohamed Eshra

ID:6958

2)Rowan Samy Gamal El-din Aly El-Rabeaa

ID:6733

3) Habiba Hossam Mahmoud Bakr

ID:6794

All of us group 4 lab (2)

➢ Introduction:

This program represents a drawing and painting application which offers a big number of features which are drawing, coloring, moving and resizing. They allow the user to undo or redo any instructions they make to give a fully complete representation of the program.

➢ Implementation:

Our program consists of 3 packages:

The first package is paint which consists of 2 JFrames:

- GUI
- Colors

The second package is shapes and consists of 8 classes:

- shape
- shapesFactory
- Line
- Rectangle
- Square
- Circle
- Triangle
- Control

The Third package is effect and consists of 5 classes:

- SelectLi
- SelectRec
- SelectSqu
- SelectCir
- SelectTri

1-paint package:

It consists of the most important class which is the GUI that simulates the whole program and class Color.

a-GUI:

It is the first thing that appears on running the program and has buttons that the user presses on to do the action he likes to do.

1-Choose a shape "JComboBox":

- It allows the user to choose the shape he wants to draw by giving him a wide range of selections between (Line – Square – Rectangle – Triangle- Circle).

2-Colors button:

- It opens a JFrame which consists of a palette of colors and the user chooses the color he wants.

3-Fill shape "check box":

- On clicking it, there is a variable called checkFill will be set as true and the chosen shape by the user will be filled.
- Otherwise, it will remain false as it is.

4-Clear button:

- It clears all the drawing area from any drawn shapes.

5-Select button:

- It allows the user to select any shape he wants and operates on it either by re-sizing, moving, copying, and pasting or deleting.

6-Re-sizing:

- It allows the user to change the size of the selected shape by dragging one of its vertices and make it whether bigger or smaller.

7-Copy and paste buttons:

- It allows the user to make a copy of the selected shape and paste it anywhere he wants by pressing on the drawing area.

8-Move button:

- It allows the user to move the selected shape and put it anywhere he wants by pressing on the drawing area.

9-Delete button:

- It allows the user to delete the selected shape from the drawing area.

10-Undo and Redo:

- If the user wants to shift any of the actions mentioned above, he will press undo to return back as he was.
- And redo to go step forward to what he was.

---

b-Colors JFrame:

- It consists of the palette of colors where the user choses the color he wants.
- It appears when the user press color button in the main GUI frame and closes on pressing ok.

2-shapes package:

It covers all properties of the geometric shapes and controls the actions made on them by setting a mode for each shape.

a-Shape class:

- It is an abstract parent class which contains all the abstracted methods which is inherited by the child classes.

- It contains a constructor that sets the private variables color and fill that checks whether the shape is filled or not.
- It consists of 2 abstracted methods :

1-draw:

Prototype: public abstract void draw(Graphics g);

-It takes a parameter of type graphics.

-It allows us to set the points needed to draw the chosen shape.

2-contains:

Prototype: public abstract boolean contains(int x, int y);

-It takes the selected point on the drawing area and sees whether it is in the shape or not.

-It returns true if the point is in the shape and false if the selected point lies outside the shape.

---

b-ShapesFactory class: (design pattern)

- It contains a private constructor that prevents us to make an instance of this class by accessing the constructor directly.
- So, we made this method:

Prototype: public static ShapesFactory getInstance( )

-It allows us to make only one instance from this class and if it is called many times in different places in the code, every time it makes the same instance.

- It also contains this method:

prototype: public Shape createShape(String name, int x1, int y1, int int x2, int y2, Color color, Boolean fill)

-It allows us to create new instance of shape according to the name of the shape (Line – Square – Rectangle – Triangle- Circle) called when calling the function.

---

c-Line class:

- It is a child class that extends shape and contains a constructor that sets the starting point and the end point of the line and sends to the constructor in shape to set the color and see if the user check it to be filled or not.
- It Override the 2 abstract methods draw and contains.
- It implements the interface “Cloneable” and Override the method **clone( )** which is responsible for copying the shape from one place to another in the drawing area.

---

d)Rectangle class:

- It is a child class that extends shape and contains a constructor that sets the starting point and the end point of the rectangle and sends to the constructor in shape to set the color and see if the user checks it to be filled or not.
- It Override the 2 abstract methods draw and contains.
- In draw method, we make conditions to make the user draw in any direction he wants.
- It implements the interface “Cloneable” and Override the method **clone( )** which is responsible for copying the shape from one place to another in the drawing area.

---

e)Square class:

- It is a child class that extends shape and contains a constructor that sets the starting point and the end point of the square and sends to the constructor in shape to set the color and see if the user check it to be filled or not.
- It Override the 2 abstract methods draw and contains.
- In draw method, we make conditions to make the user draw in any direction he wants and we set it's four sides to be equal.
- It implements the interface "Cloneable" and Override the method **clone( )** which is responsible for copying the shape from one place to another in the drawing area.

---

f)Circle class:

- It is a child class that extends shape and contains a constructor that sets the starting point and the end point of the Circle and sends to the constructor in shape to set the color and see if the user checks it to be filled or not.
- It Override the 2 abstract methods draw and contains.
- In draw method, we make conditions to make the user draw in any direction he wants and we set it's four sides to be equal and draws a circle inside it.
- It implements the interface "Cloneable" and Override the method **clone( )** which is responsible for copying the shape from one place to another in the drawing area.

g) Triangle class:

- It is a child class that extends shape and contains a constructor that sets the first vertices and the second vertices of the Triangle and sends to the constructor in shape to set the color and see if the user checks it to be filled or not.

- It consist of method called “setArray()” :
  -It allows us to set the vertices of the triangle in an array as to draw a triangle we need an array of points.
- It consists of 2 methods to get the third point that is set manual in the array (third vertices (2*x1-x2, y2))
- It Override the 2 abstract methods draw and contains.
- It implements the interface “Cloneable” and Override the method **clone( )** which is responsible for copying the shape from one place to another in the drawing area.

---

h-Control class:

- It extends JPanel class, so it Override paintComponent method, as well as it implements MouseListener, MouseMotionListener classes so it Override mousePressed and mouseDragged methods.
- Therefore, it contains several Override methods which are:

1- Prototype: protected void paintComponent(Graphics g)
Description: This method loops on the array list of shapes and repaints them on calling.

2- Prototype: public void mousePressed(MouseEvent me)

Description:

This method reads the mode of the operation from the GUI and determines the task depending on the mode:

- mode from 0 till 4: Creates a new instance of the chosen shape and adds to the array list and repaints.
- mode 5: It loops on the array list backwards and checks if any of the shapes contains the selected point on the drawing area.
- mode 8 and 9: We called a variable of type shape and if this variable doesn’t equal null he calls determineShape method and repaints the array.

- mode 10: It sets the delete variable of the selected shape true and adds it to the undo array then removes it.
- mode -1:is a default mode for our program and if none of the previous modes is set a message dialogue will appear informing the user to select a shape first.

3- Prototype: public void mouseDragged(MouseEvent me)
Description: this method determines the suitable task according to the mode as:

- mode from 0 till 4: takes the second point of the shape and casts it to the chosen shape and sets its dimensions.
- mode 6 or 7: takes the second point of the shape and calls determineShape then repaints.

- In addition, two other methods:

4- Prototype: public static double distance(int x1, int y1, int x2, int y2)
Description: takes the co-ordinates of two points and returns the distance between them.

5- Prototype: public void determineShape(Shape selectedShape)
Description: It contains conditions that determines the instant of the parameter shape then:

- If mode is 6: it calls the re-size method according to the determined shape.
- If mode is 7: it calls the copy method according to the determined shape.
- If mode is 8: it calls the paste method according to the determined shape.
- If mode is 9: it calls the move method according to the determined shape.

3- effect package:

- This package includes different classes for each shape.
- Each class includes the following methods:

- resize:
  - It creates a new instance according to the selected shape in drawing area and set its dimensions according to the dragged point.
  - It adds the new instant to the array list <shape> and remove the old version of the selected shape then adds it to the undo list.
- Copy: It clones the copied shape.
- Paste:
  - It creates a new instance according to the copied shape in drawing area and set its dimensions, color, and fill according to the copied shape.
  - It adds the new instant to the array list <shape>.
- Move:
  - It creates a new instance according to the selected shape in drawing area and set its dimensions according to the dragged point.
  - It adds the new instant to the array list <shape> and remove the old version of the selected shape then adds it to the undo list.

Design Patterns

➢ Creational Patterns:

1- Singleton:

We applied it in shapeFactory class where we have only one instance of a class as we made a private constructor so it can only be accessed using getInstance( ) method.

2- Factory:

- This pattern provides a simple decision-making class that returns one of several possible subclasses of an abstract class depending on the provided data.
- We applied this concept in ShapeFactory class where we made createShape method that returns a new instance of the shape depending on the name of the shape sent to the function.
- So, every time we want to create a new instance of any shape, we call createShape method.

3- Prototype:

- This pattern involves implementing a prototype interface which tells to create a copy of the current object rather than creating new instances.
- It starts with an initialized class which is clone that Override from Cloneable class.

➢ Behavioral Pattern:

4- Iteration:

- This pattern is used to access the elements of a collection of objects in sequential manner without any need to know its underlying representation.
- We applied this concept to loop on the array list in Control class.

5-Template design pattern:

- It defines an algorithm as a skeleton of operations and leave the details to be implemented by the child classes. The overall structure and sequence of the algorithm is preserved by the parent class.
- We applied it as we created an abstract class “shapes” and all shapes inherit from it same methods “draw() and contain()” .

➢ Structural Pattern:

6- Composite:

- It allows a client object to treat both single components and collections of components identically.
- We applied this concept in array list of shapes as we represented part whole hierarchies of objects where we collected all types of shapes in one array list to be treated all in the same way.

Solid Principles

1-Single responsibility principle:

- It states that a class should have one and only one reason for the change where each class should have one purpose and two or more separate responsibilities should be separated into distinct classes.
- So, it reduces the code complexity and better testability.
- We applied this principle by implementing a class for each shape that has its own functions to be accessed with.

2- Open/Closed principle:

- It states that every class should be opened for extension but closed for modification.
- We tried to apply it as much as we can as if the requirements of the application changed, we are able to modify the program with new behaviors that satisfy those changes.

3- Liskov substitution principle:

- It focuses on the behavior of the superclass and its subclasses and states that objects can be replaced with the instances of their subtypes without breaking the code.
- This principle is applied all over the code using instances of square, rectangle,….etc. to access the shape (super class) variables.

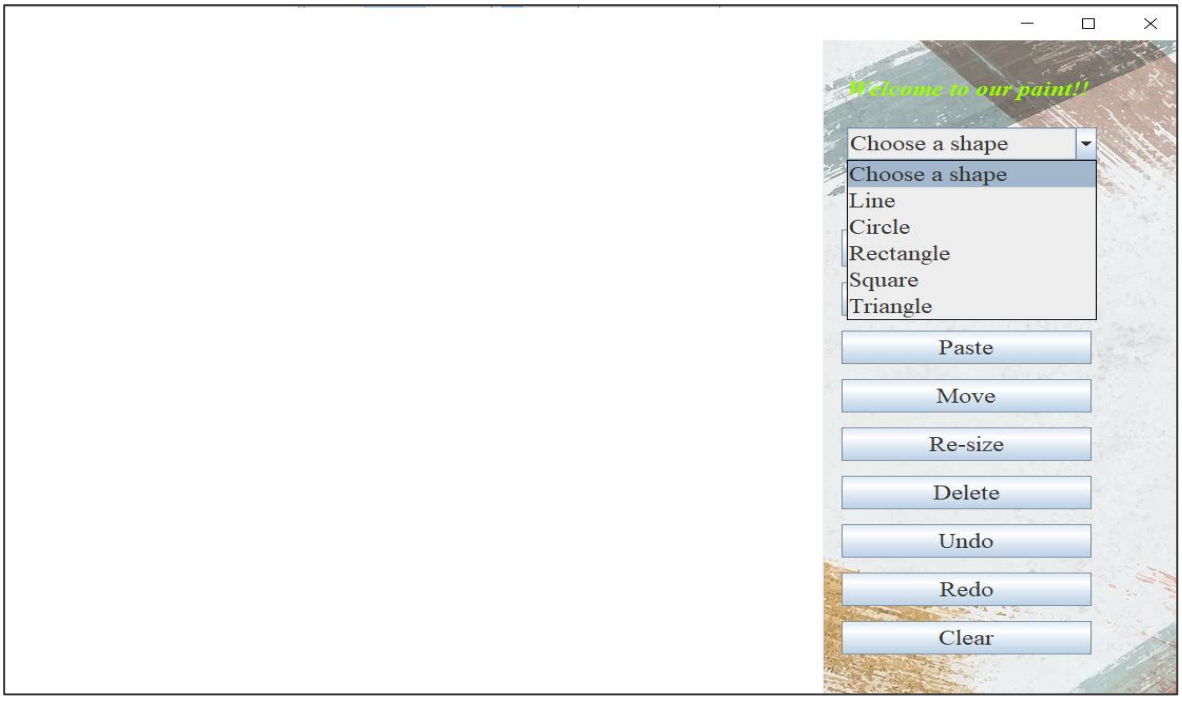
4-Dependency inversion principle:

- It states that high level should not depend on low level modules.
- We tried to apply it as much as we can as the GUI don't call the methods its self but they are linked together using the modes.

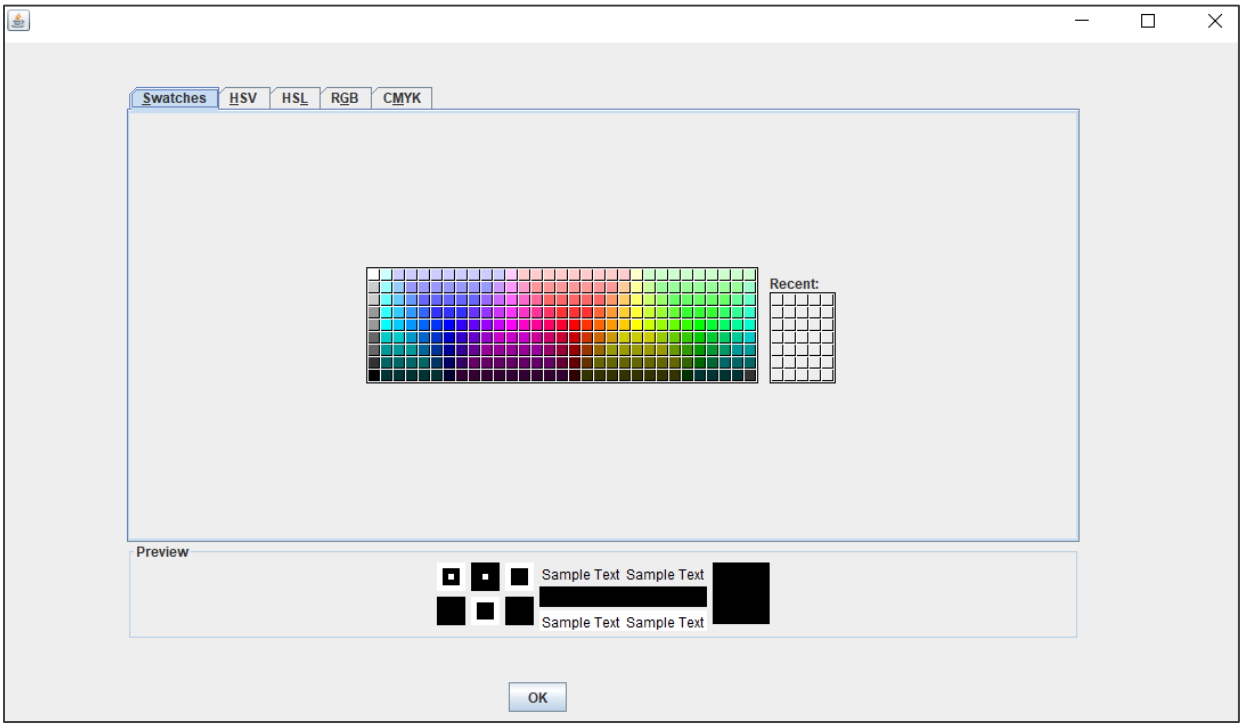
Sample Run

- The main GUI that appears on running the program:

- Then the user can select the shape he wants from the shape combo box.

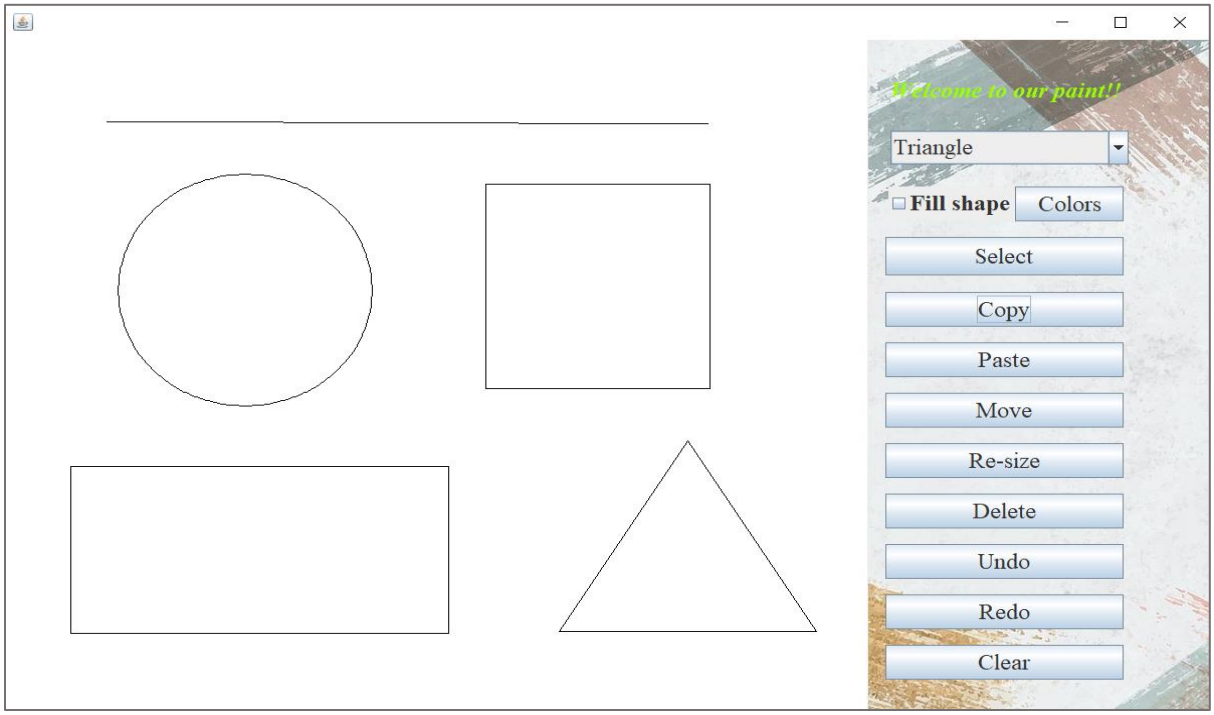


- He can choose the color he wants as well by pressing on color button so it opens the color palette.

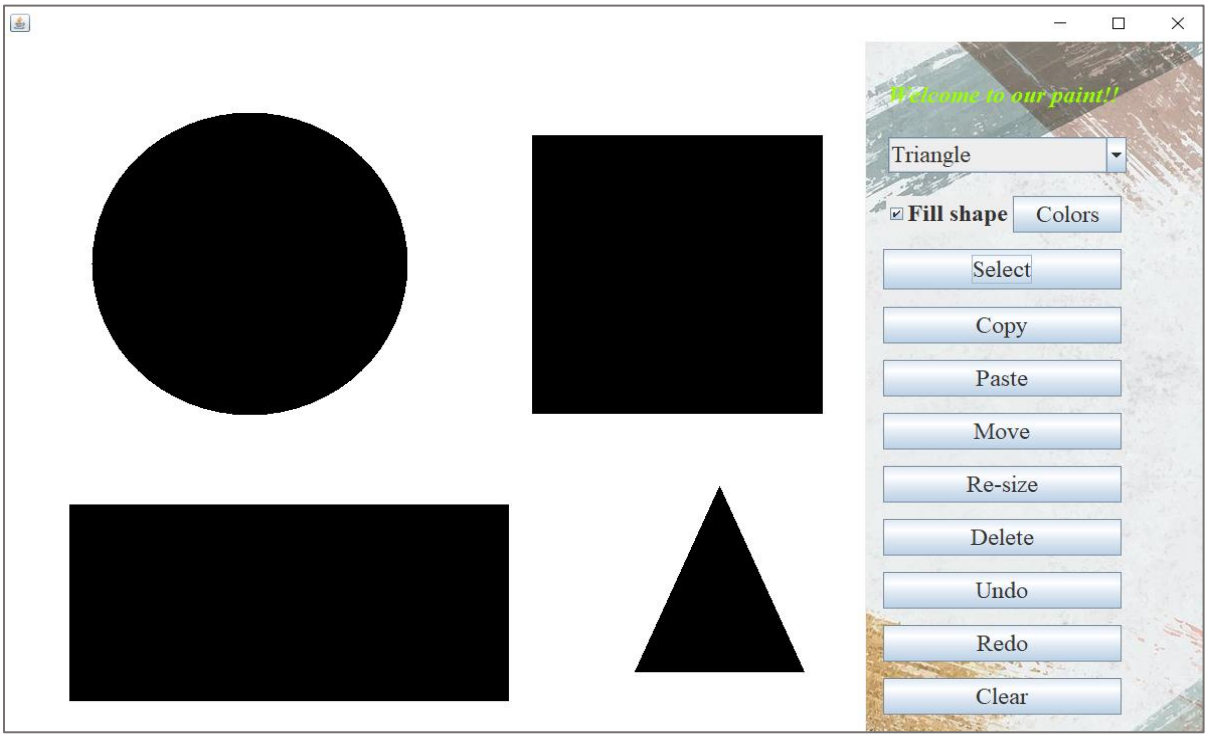


- He can also choose whether the shape is filled or not by ticking the check box.

  1- Not filled:

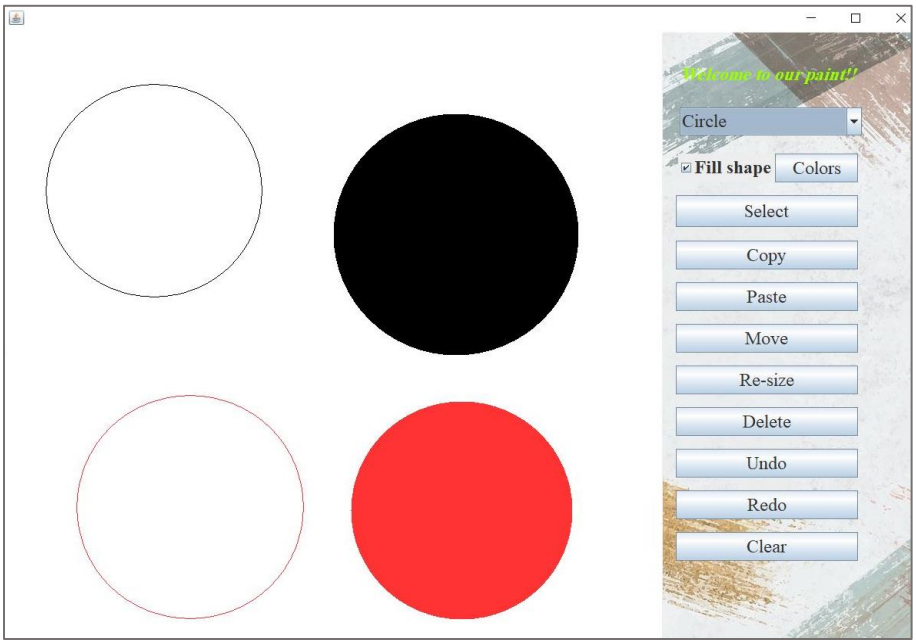


2- Filled (check box ticked)

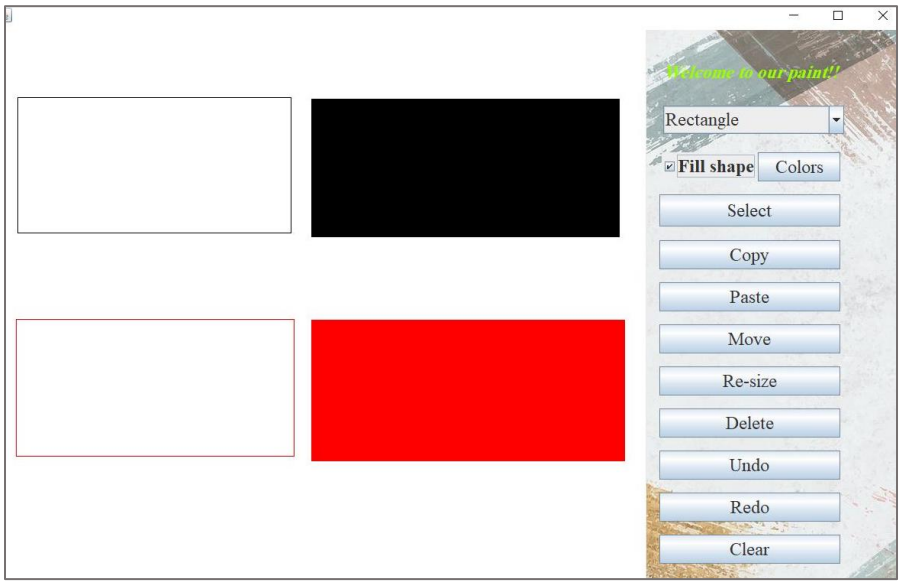


- Then the user can draw whatever he wants with the color he wants whether filled or not.

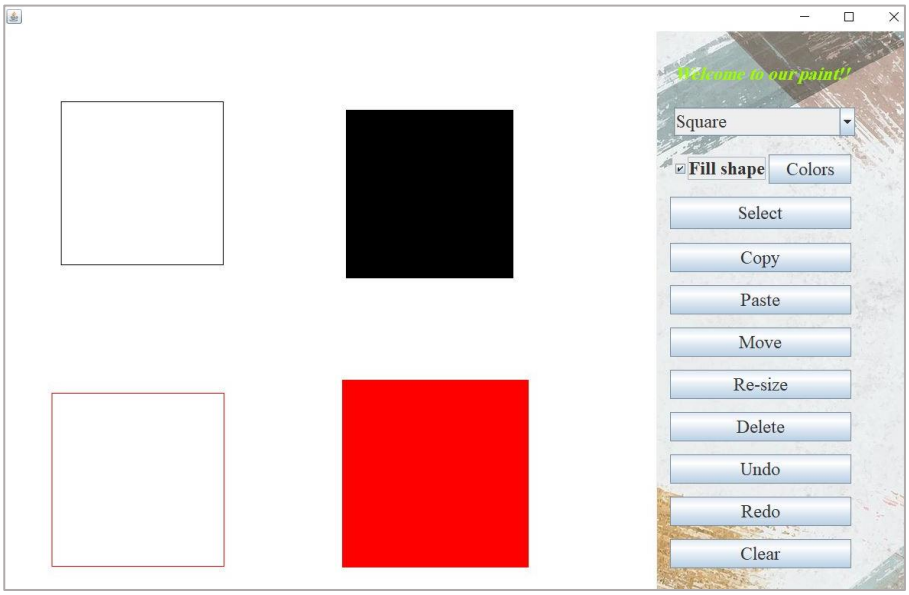
1- Circle:



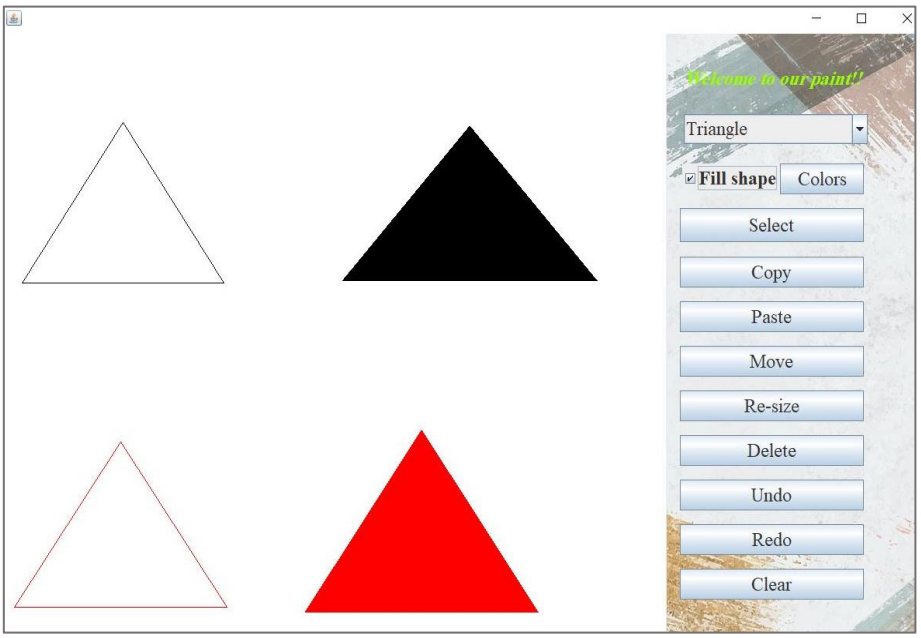
2-Rectangle:



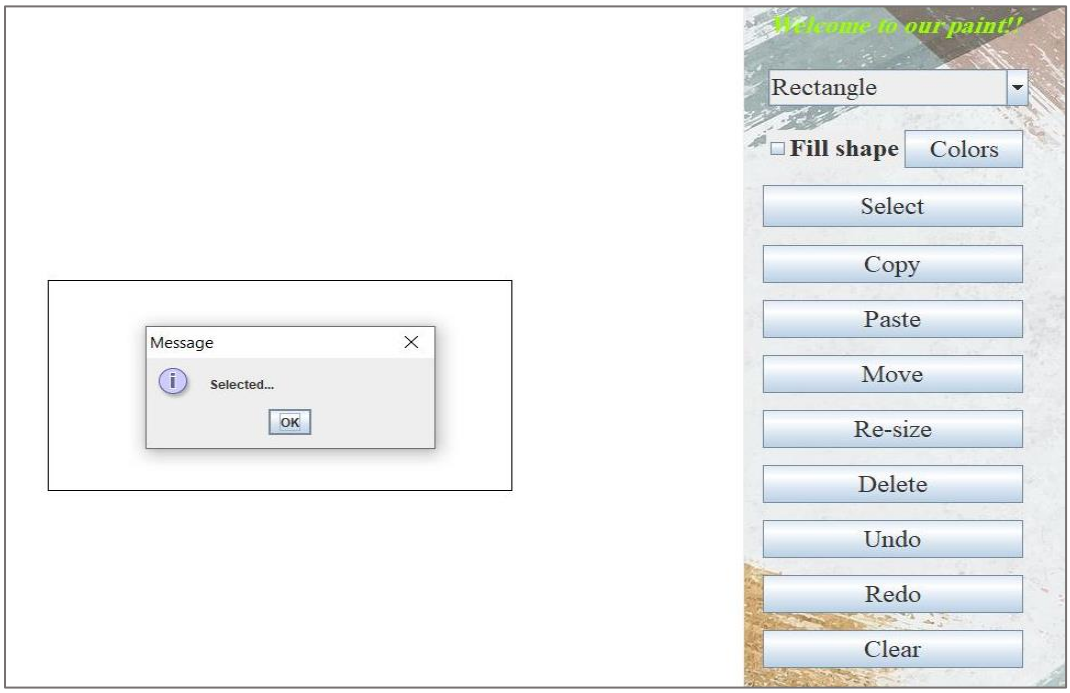
3-Square:



4-Triangle:

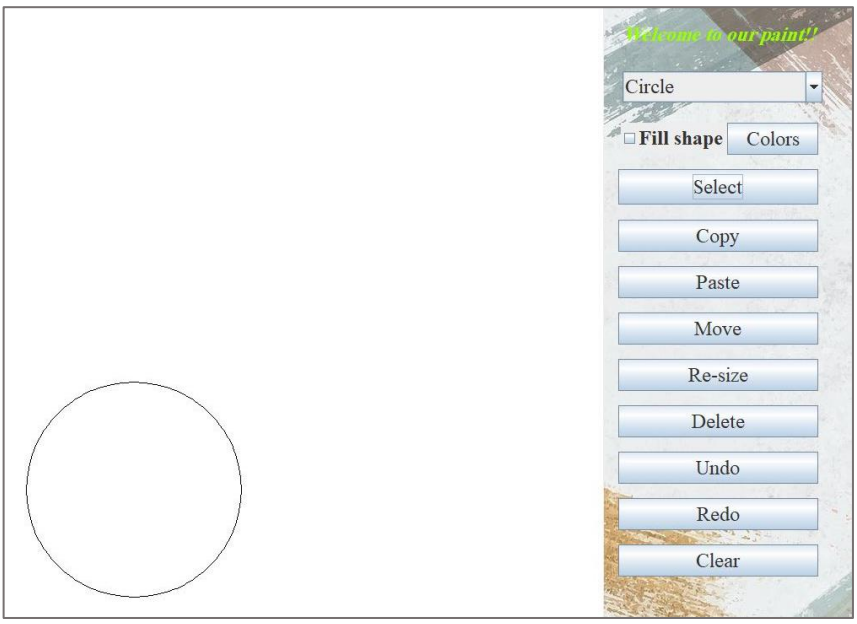


- The user can choose any effect he wants but he must select the shape first.

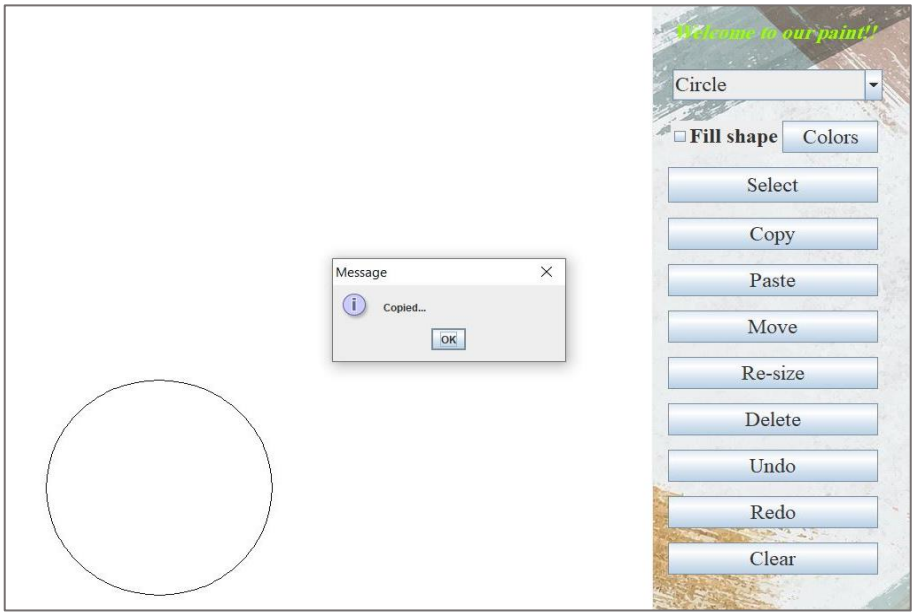


1-Copy and paste:

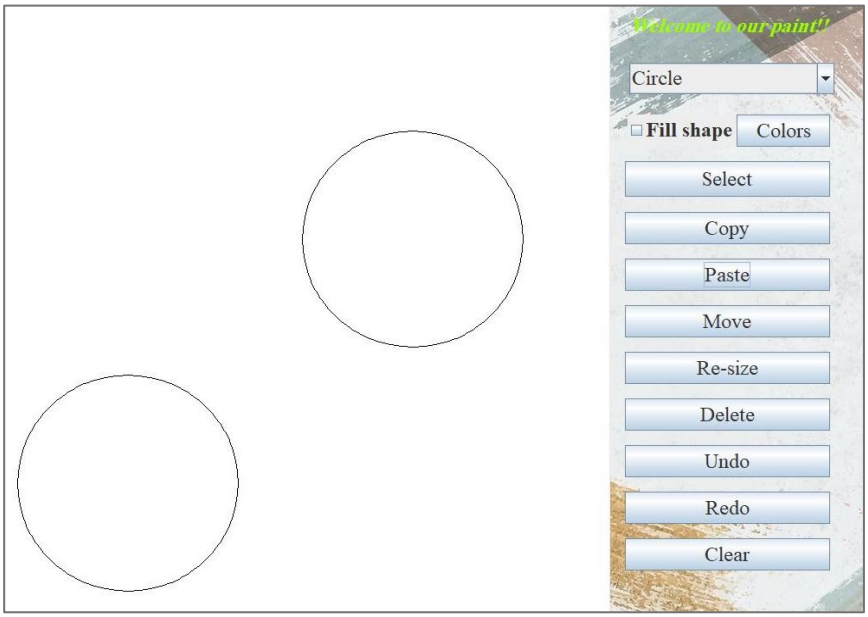
Before copying:



Copying:

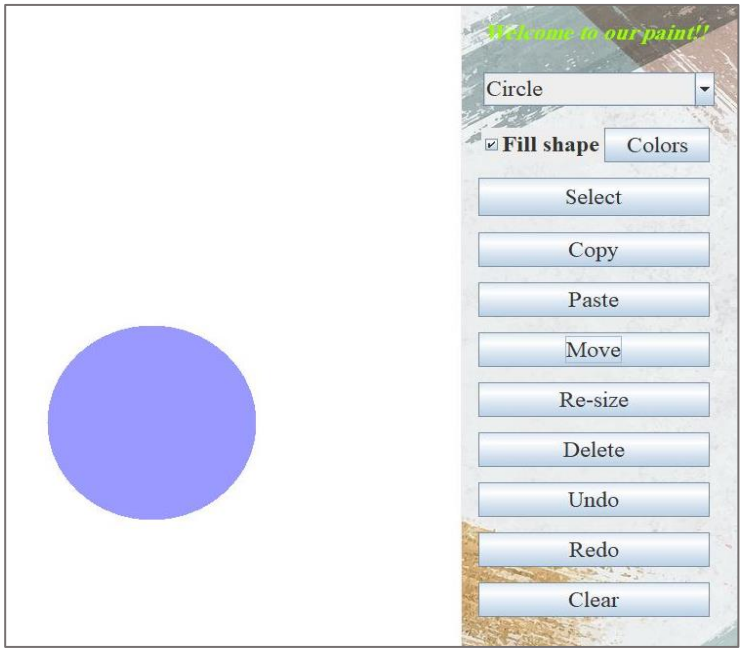


Paste:

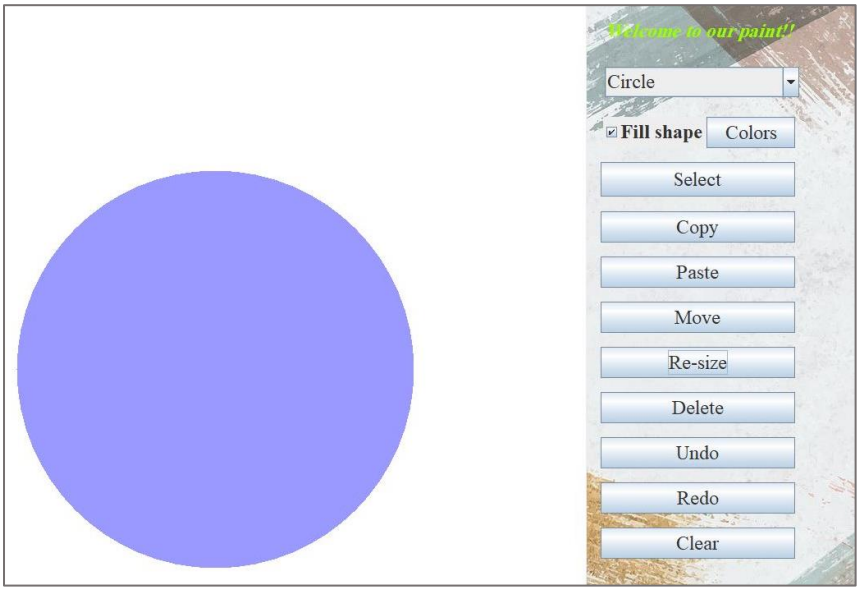


2-Re-size:

Before re-size:

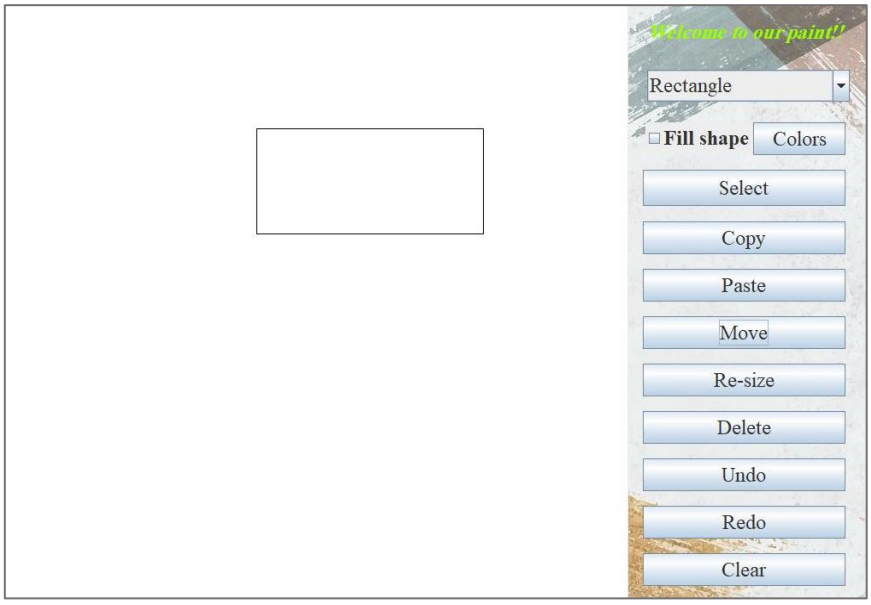


After re-size:



3-Move:

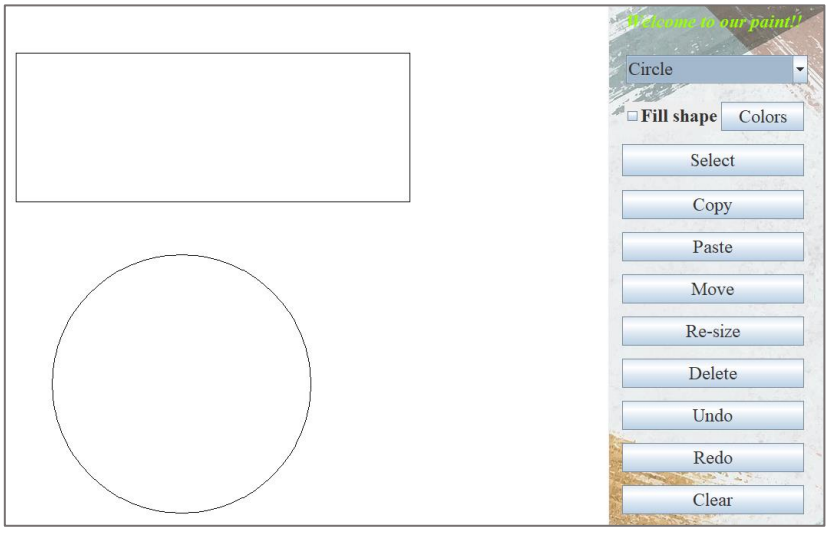
Before move:



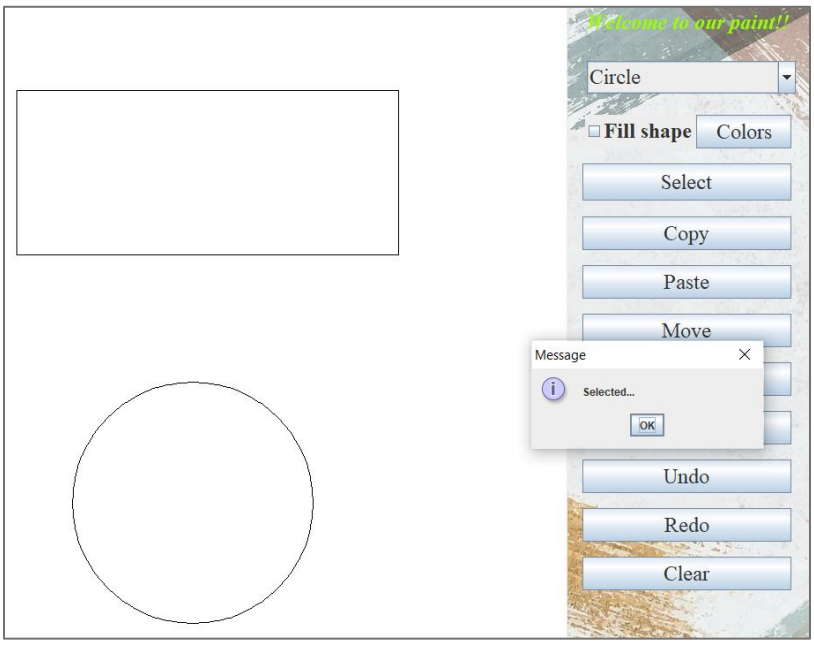
After move:

4-Delete:

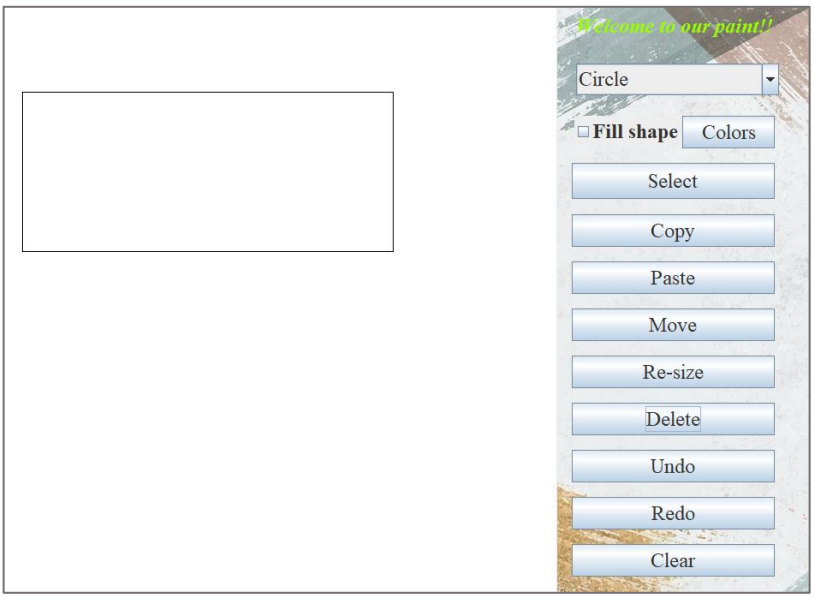
Before delete:



Selecting the circle to delete it:

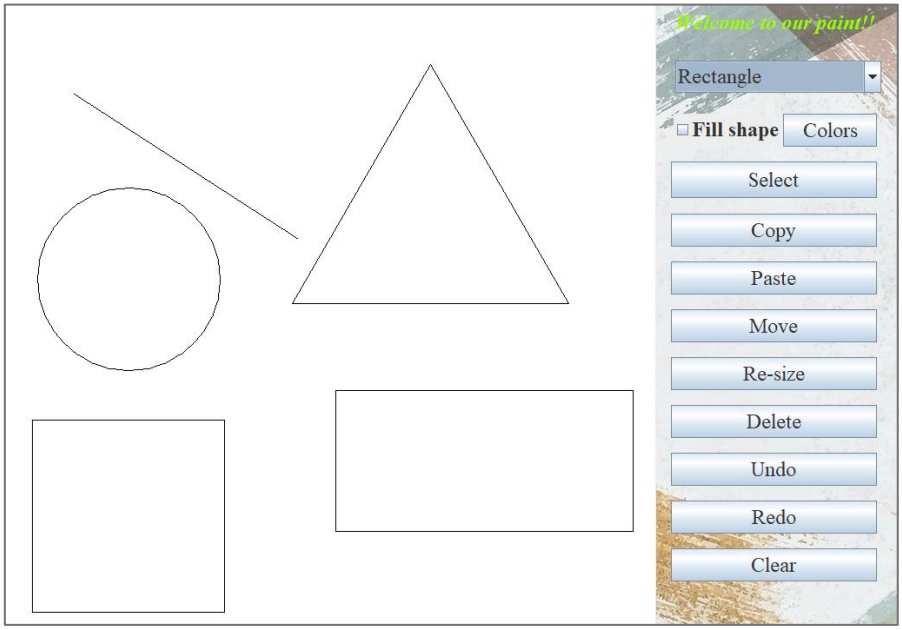


Deleted:

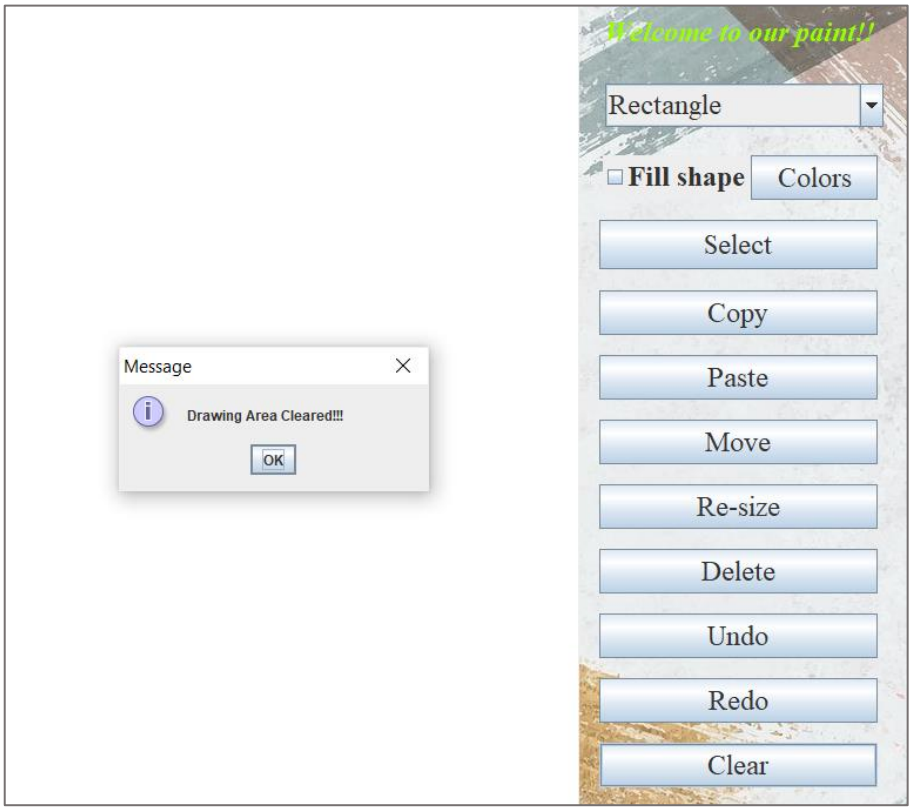


5-clear:

So, we want to clear this area.

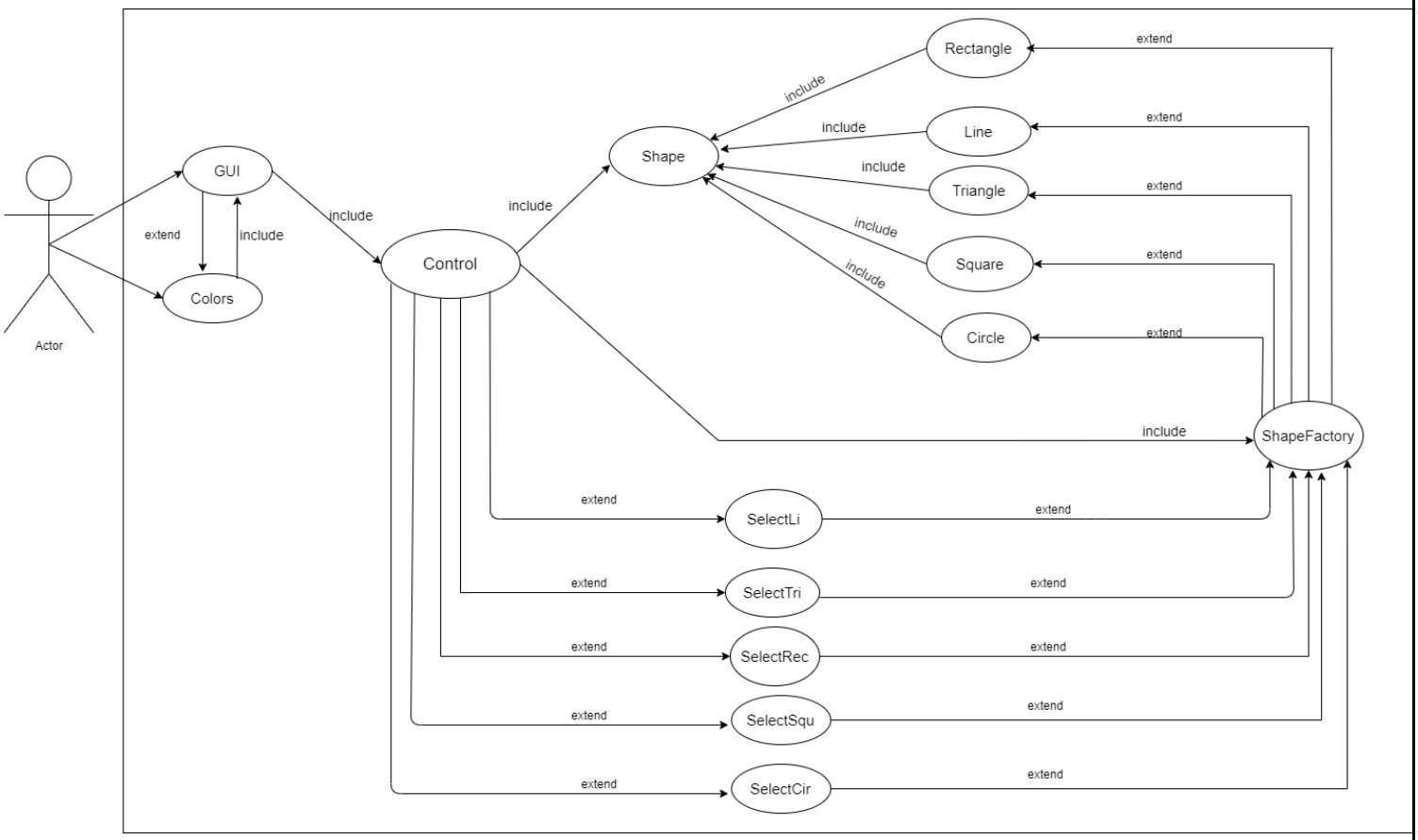


Cleared successfully.



UML

- Usecase diagram:



➢ Class diagram: