

CSC255 Programming III

Assignment 3: Draw a shape with a different color

Total: 100 pts

For all programming assignments, include the following: Writing codes with suitable documentation comments

1. Write a GUI program using JavaFX codes that enables the user to control various aspects of drawing from a ComboBox as well as display all information in the TextArea.

The GUI program is similar to the following figure 1.

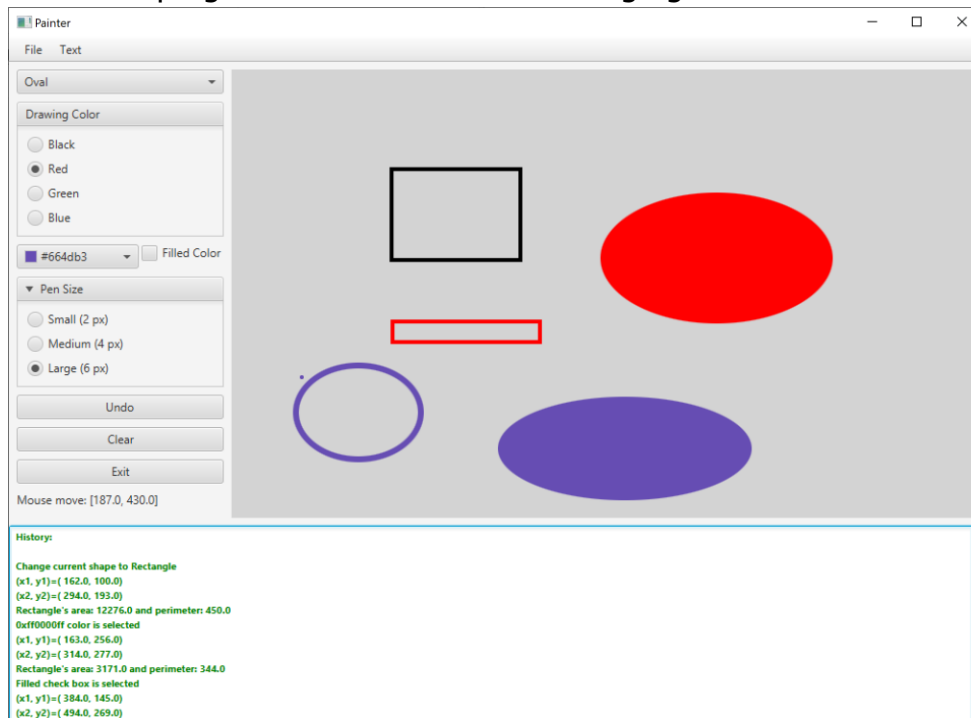
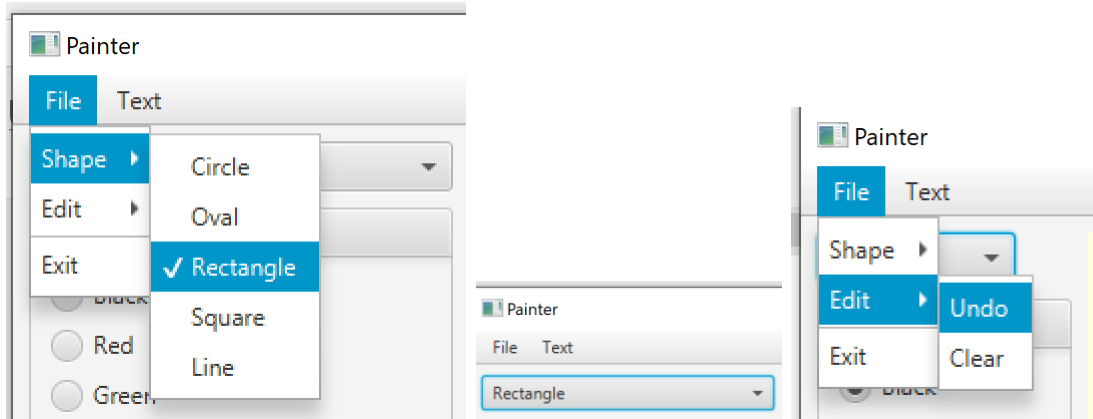
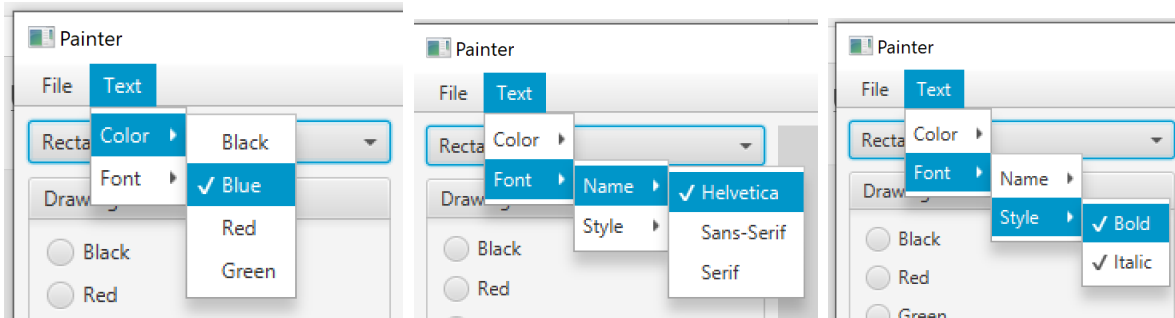


Figure 1

- a. Set up the menu bar for File and Text menu. See the following figures.
 - i. **File** menu includes **Shape**, **Edit**, **separator**, and **Exit** submenu.
 - **Shape** submenu include five (5) submenus, which create from RadioMenuItem object: **Circle** (the default), **Oval**, **Rectangle**, **Square**, and **Line** etc.
 - Event handle for each submenu: Shape submenu includes shapes which match with the shapes in the ComboBox. i.e., if you select "Rectangle", the ComboBox will also display "Rectangle". (See the following figure)



- **Edit** submenu includes **Undo** and **Clear** submenus.
 - a. Event handlers for each submenu.
- **Exit** submenu – event handling for ending the program.
- ii. **Text** menu includes all submenus for changing **TextArea's** information text **color**, and **font**.
Text menu includes **Color** and **Font** submenus. See below figures.



- **Color** submenu:
 - a. Includes four RadioMenuItems.
 - b. Allow the user to select Black, Blue (the default), Red, and Green color for the text color. Need to use ToggleGroup to group together.
- **Font** submenu:
 - a. Includes **Name** and **Style** submenus.
 - i. **Name** submenu: allow the user to select Helvetica (default), Sans-Serif or Serif RadioMenuItems for changing font name.
 - 1. Need **ToggleGroup** to group together
 - ii. **Style** submenu: Provide a **Bold** CheckMenuItem and an **Italic** CheckMenuItem, which, if checked, makes the text bold, italic or both.
- Event handlers for all necessary submenus.

This program allows the user to:

- b. Select which shape to draw from a comb box. A **ComboBox** class should provide options including "Circle", "Oval", "Rectangle", "Square", "Line", etc. (Figure 2)

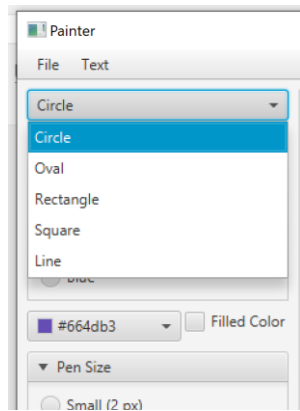


Figure 2

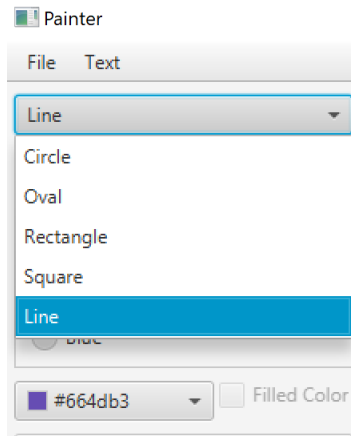


Figure 3

- i. The first item in the ComboBox should be **Circle** (the default shape).
- ii. Specify whether a shape should be filled or empty when it is drawn. The user should click a **"Filled Color"** CheckBox to indicate filled or empty. In addition, if the drawing comboBox selects the "Line" option, then this check box will be disabled to click. (see Figure 3)

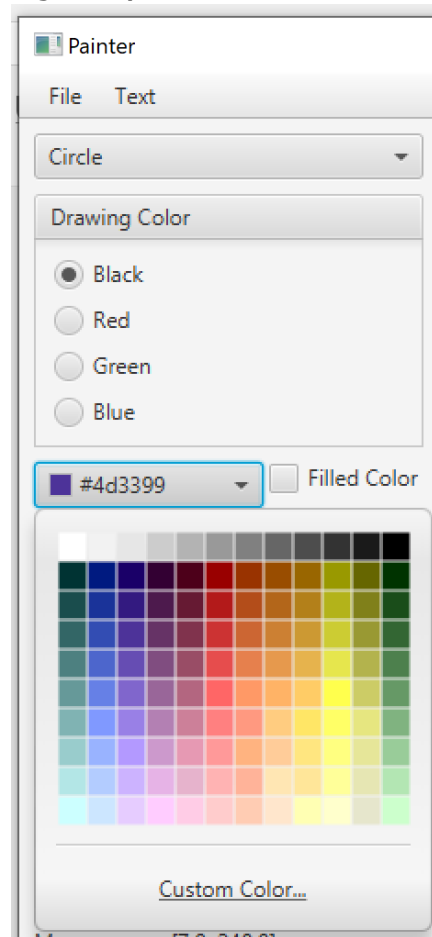


Figure 4

- iii. Select the drawing color from four "Drawing Color": Black (**the default color**), Red, Green, and Blue radio buttons as well as choose the drawing color from a **ColorPicker** dialog box. (see Figure 4)

- iv. Pen Size with 2 pixels, 4 px (the default size) and 6 px for line width if the "Filled Color" check box is not selected.
- v. Three buttons: Undo, Clear and Exit buttons.
 - Undo the previous drawing from **Undo** button and undo submenu under File menu
 - **Clear** button and Clear submenu under File menu remove all drawings.
 - **Exit** button and Exit submenu under File menu: terminate the program
- vi. (Bottom-Left) Label will display the appropriate coordinates when the mouse moves. i.e., Mouse move: [120.0, 342.0]

See Figure 5

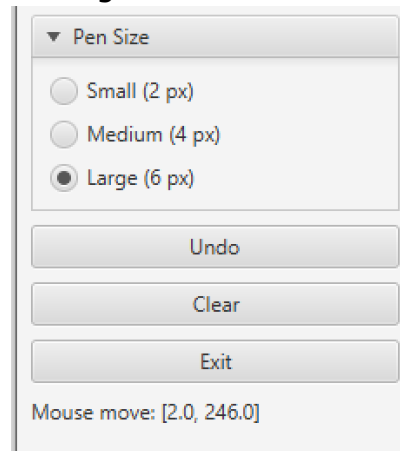


Figure 5

- vii. Set the **TextArea** with the default text color to **blue** and background color to **white**. Un-Editable TextArea object with the scroll bar displays all information includes shape, coordinates (x1, y1) and (x2, y2), area and perimeter which invokes from getArea() and getPerimeter() methods for all bounded shapes as well as getName() for all shapes from classes you have created from the previous assignment.
 - The coordinate displays with one decimal places
 - The values of area and perimeter display with two decimal places.
 - In addition, the text area will display all history (Figure 6)

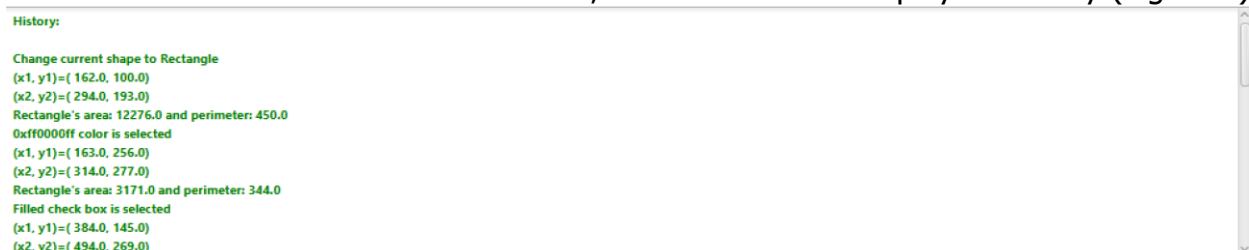


Figure 6

- viii. Event handlers for all necessary objects.

- c. Allow the user to press the mouse button, drag the mouse or release the mouse button to able to draw any shape based on the user choosing from

the combo box or shape submenu under File menu, the appropriate color, pen size and whether filled color or not. (see Figure 7)

- i. Hint: you can either use "Circle", "Ellipse", "Line" and "Rectangle" classes under `javafx.scene.shape` package (for example: `import javafx.scene.shape.Ellipse;`) for drawing "Circle", "Oval", "Rectangle", "Square" and "Line" shape onto the pane or use any methods from `javafx.scene.canvas`, which can fill/or stroke any shapes with `GraphicsContext`
- ii. When the mouse is pressed, draw a shape with the appropriate top-left corner.
- iii. When the mouse is released/dragged, draw a shape with the appropriate bottom-right corner, width and height. [Hint: The **mousePressed** method should capture the set of coordinates at which the user presses and holds the mouse button initially, and the **mouseReleased** method should capture the set of coordinates at which the user releases the mouse button. Both methods should store the appropriate coordinate values.]

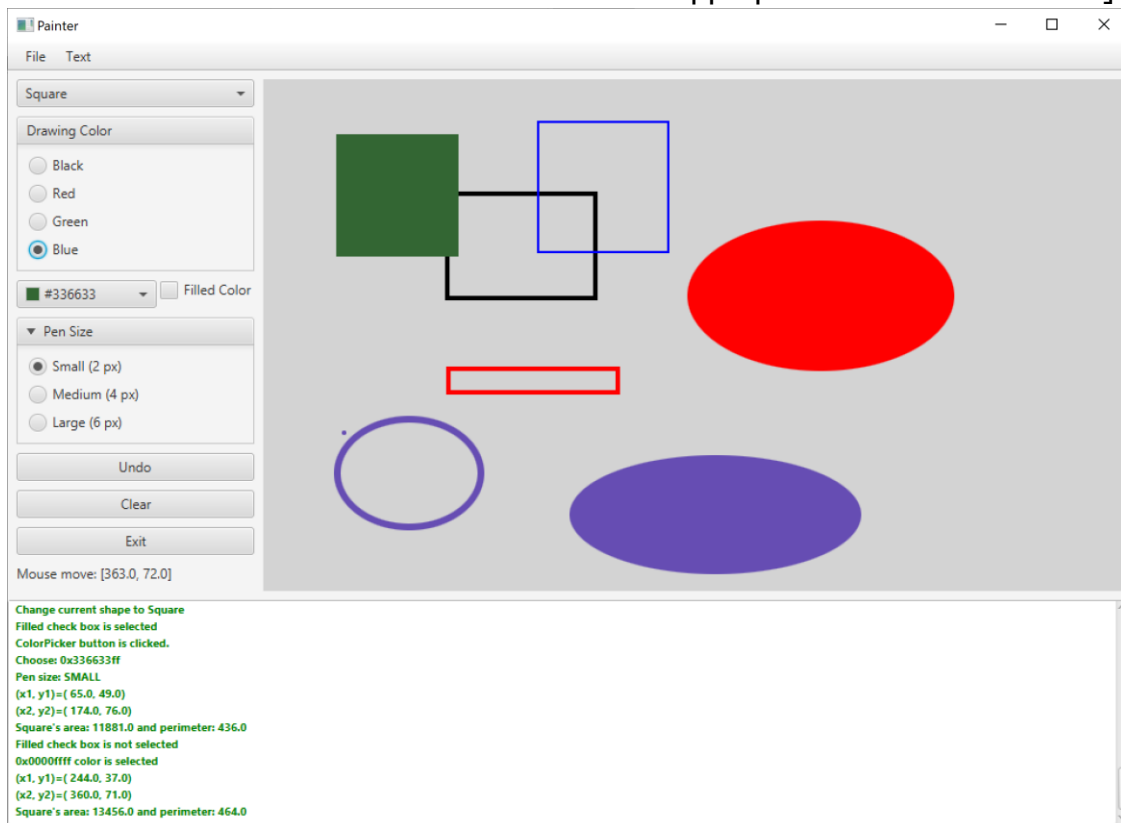


Figure 7

2. Your program is similar to the above figures or with more creatives and features.
3. CANNOT use any GUI builder, scene builder and GUI designer to create your interface. You have to use JavaFX codes to design and create the program.
4. All source codes MUST include the code documentation (For each source file).
5. Copy will receive zero (0) points.
6. Zip all file under `xxA3.zip` file. Where `xx` is your name's initials.