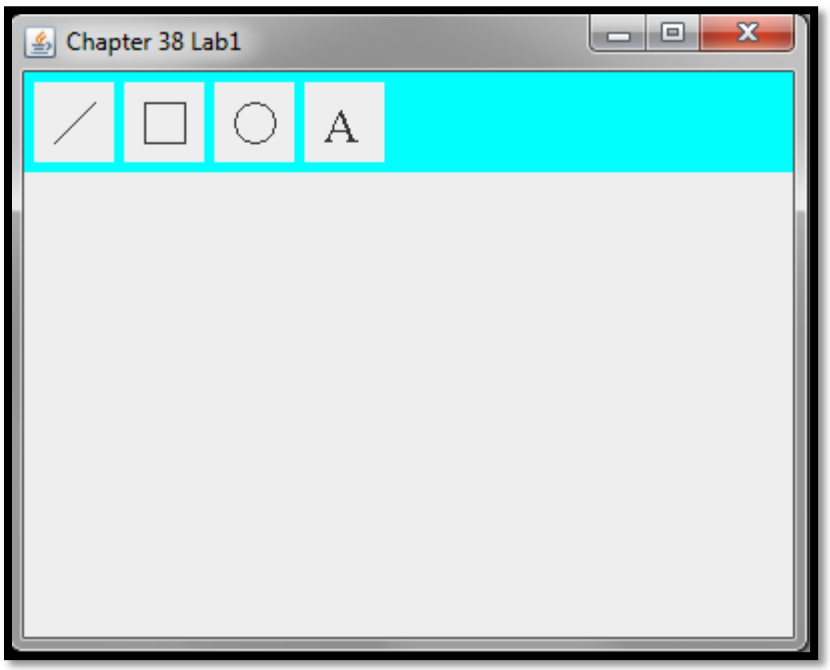
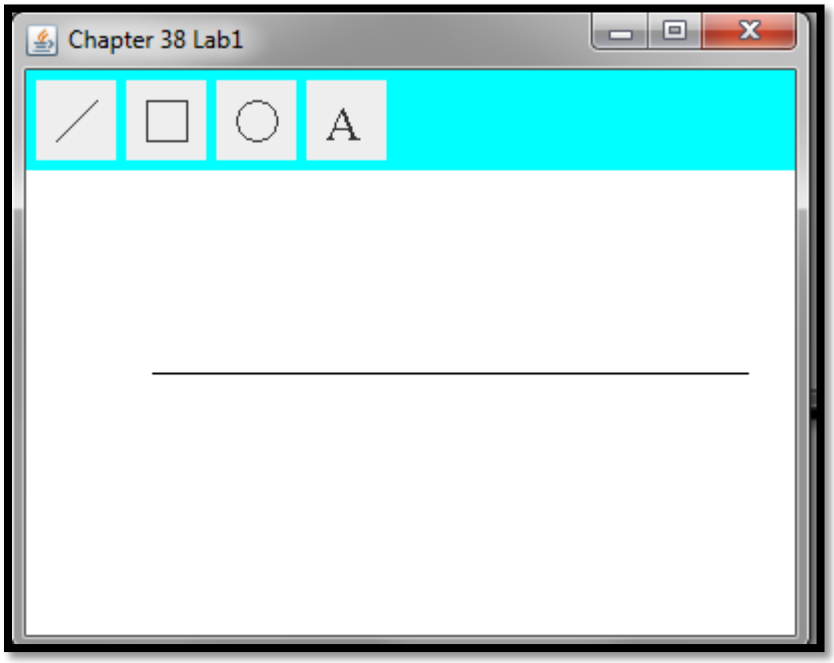
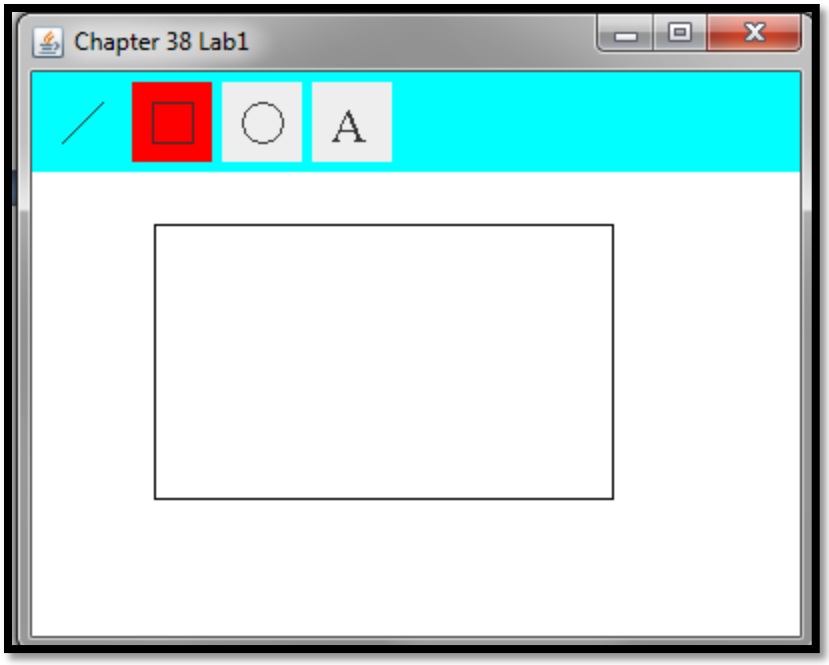
# CIT 249: Java II
# Chapter 38 Lab1

In this lab we will complete a program that will enable the user to choose options and draw shapes or get characters from the keyboard based on the selected options, see following screenshots. :

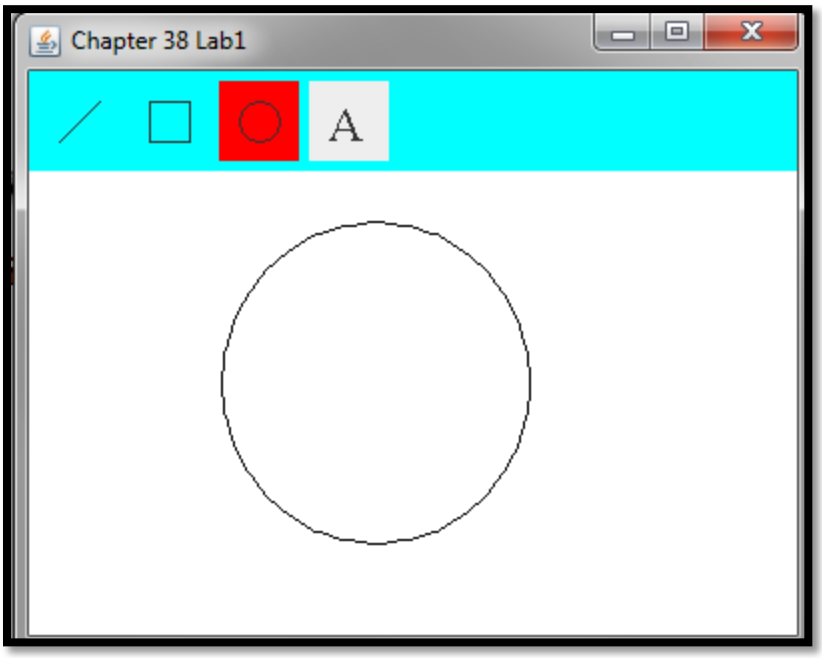


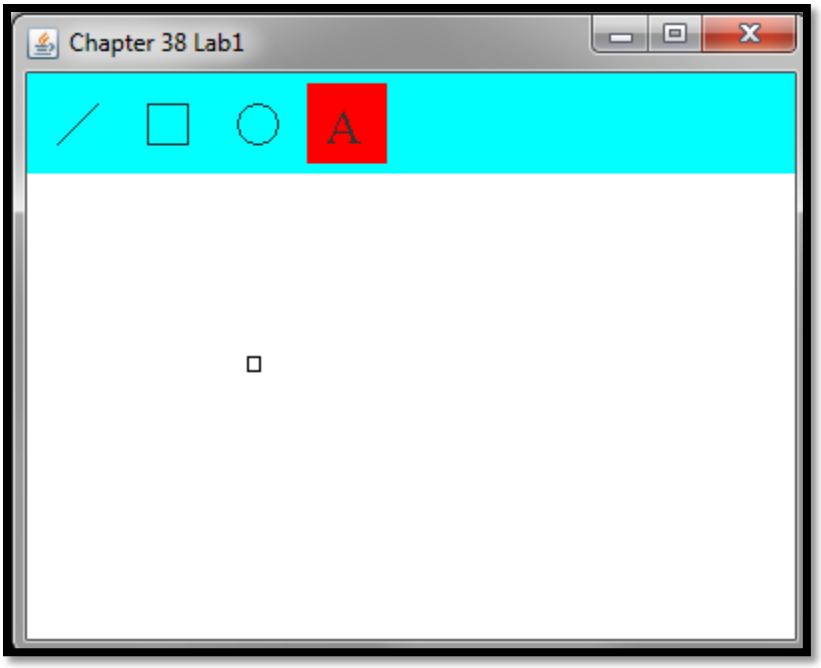

When the first icon is selected you can draw a line on the canvas




When the rectangle icon is selected it turns red and you can draw a rectangle on the canvas




When the circle icon is selected it turns red and you can draw a circle




If the A icon is selected it will display the letter you typed on the canvas. In this case I pressed the Print Screen key

Let's get started!

1. Open a new document window and say the file as Ch38Lab1.java. The purpose of the program is to:

   /* Purpose: draw shapes onto a canvas based on selected icons. Mouse and keyboard listeners will be used
   */

2. Type the import statements that will import all classes in the java.awt, java.awt.event and javax.swing packages.
3. Type the class header and opening brace having the class extend the JApplet class.

4. Type the following code that will declare or construct several private variables.

```
//declare integer, KeyListener object, and construct MyIcon object array and
//DisplayCanvas object.

private int currentType = 0;
private MyIcon[] c = new MyIcon[4];
private DisplayCanvas canvas = new DisplayCanvas();
private KeyListener keyListener;
```

- Here we construct an integer setting it to zero.
- Construct an array of MyIcon objects, setting the limit to 4.
- Construct a new DisplayCanvas object.
- Declare a KeyListener object.

5. Next is our constructor method. Type:

```
public Ch38Lab1()
{
   //construct panel, setting attributes, and add icons to panel
  JPanel p = new JPanel();
  p.setBackground(Color.cyan);
  p.setLayout(new FlowLayout(FlowLayout.LEFT));

  for (int i = 0; i < 4; i++)
    p.add(c[i] = new MyIcon(i));

   //set layout for frame/applet and add panel and canvas object to it
  setLayout(new BorderLayout());
  add(p, BorderLayout.NORTH);
  add(canvas, BorderLayout.CENTER);
 }
```

- We create a new Panel and set its background color to cyan and its layout to FlowLayout with left alignment.
- Using a for loop we construct each element in the MyIcon array of *c*, adding the array elements to the panel at the same time. Each element in an array must be constructed.
- Set the frame's layout to BorderLayout.
- Add the panel to the NORTH.

- Add the DisplayCanvas object to the CENTER.

6. Next we create an inner class. Start by typing:

```
//inner class MyIcon
public class MyIcon extends JPanel
{
```

- This class will construct a panel in which an array will be formed from.

7. We construct an integer. Type:

```
private int type = 0;
```

8. Our constructor method comes next. Type:

```
public MyIcon(int t)
{
    type = t;

   //add MouseListener for the icon objects
    addMouseListener(new MouseAdapter()
    {
     public void mousePressed(MouseEvent e)
     {
      if ((currentType == 3) && (type != 3))
       canvas.removeKeyListener(keyListener);

      //sets to focus on the canvas if currentType does not equa 3 and type equals 3
      if ((currentType != 3) && (type == 3))
      {
       canvas.addKeyListener(keyListener);
       canvas.requestFocus();
       System.out.println("Key listener added");
      }

      if (currentType != type)
      {
       c[currentType].setBackground(Color.cyan);
       c[type].setBackground(Color.red);
       currentType = type;
      }
     }
    });
  }
```

- We add a MouseListener.
- If the mouse is pressed:
    - If *currentType* equals 3 AND *type* DOES NOT equal 3, remove the listener from the *DisplayCanvas* object.
    - If *currentType* DOES NOT equal 3 AND *type* equals 3, add a listener to the *DisplayCanvas* object and set its focus on it and at the command prompt display the words “Key Listener added”

- If *currentType* DOES NOT equal *type*, set the background color of the *MyIcon* object with the element of *currentType* to cyan; set the *MyIcon* object with an element of *type* to a background color of red.
- Set *currentType* to equal *type*.

9. We next type our paintComponent() method.

```
public void paintComponent(Graphics g)
{
   //invoke the superclass paintComponent() method passing the value of the Graphics
   //object
    super.paintComponent(g);

    int width = getSize().width;
    int height = getSize().height;

    //value of the type variable controls which object is drawn
    switch (type)
    {
      case 0:
      g.drawLine(10, height-10, width-10, 10);
      break;

      case 1:
      g.drawRect(10, 10, width-20, height-20);
      break;

      case 2:
      g.drawOval(10, 10, width-20, height-20);
      break;

      case 3:
      g.setFont(new Font("TimesRoman", Font.PLAIN, 24));
      g.drawString("A", 10, 30);
    }
  }
```

- First we invoke the superclass' paintComponent() constructor method.
- We create two integers that represent the width and height.
- We have a switch statement:

    - If *type* = 0 we draw a line
    - If *type* = 1 we draw a rectangle
    - If *type* = 2 we draw an oval
    - If *type* = 3 we set the font to 24 points and draw the A for text.
- All are for the initial display that you click to see what will be drawn.

10. Next we set the preferred size of the *MyIcon* objects. Type:

```
public Dimension getPreferredSize()
{
    //responsible for setting preferred size of the icons
     return new Dimension(40, 40);
}
```

11. Close the MyIcon class.
12. Next we create another inner class. Type:

```
public class DisplayCanvas extends JPanel
{
   private Point start = new Point(20, 20); //line start point
   private Graphics g;
```

- This class extends JPanel and will be used to construct a panel.
- We first construct a new *Point* object that will set the starting point for a line that is drawn.
- We declare a new *Graphics* object.

13. We start creating the constructor method. Type:

```
public DisplayCanvas()
{
    //constructs the keyListener object
     keyListener = new KeyAdapter()
     {
       public void keyPressed(KeyEvent e)
       {
         char keyChar = e.getKeyChar();
         g = getGraphics();
         g.clearRect(0, 0, getSize().width, getSize().height);
         g.drawString(String.valueOf(keyChar), start.x, start.y);
         System.out.println("Key pressed");
       }
     };
```

14. We add a listener for the dragging of the mouse. Type:

```
//add MouseListener in order to draw the objects
addMouseMotionListener(new MouseMotionAdapter()
{
      public void mouseDragged(MouseEvent e)
      {
        g = getGraphics();
        g.clearRect(0, 0, getSize().width, getSize().height);
        switch (currentType)
        {
          case 0: g.drawLine(start.x, start.y, e.getX(), e.getY()); break;
          case 1: g.drawRect(start.x, start.y, e.getX()-start.x, e.getY()-start.y); break;
          case 2: g.drawOval(start.x, start.y, e.getX()-start.x, e.getY()-start.y); break;
        }
      }
    });
```

15. We add a listener for when the mouse is pressed. Type:

```
addMouseListener(new MouseAdapter()
{
      public void mousePressed(MouseEvent e)
      {
        start.move(e.getX(), e.getY());
      }
     });
```

16. Close the constructor method and the class.
17. Finally we create our main method of our Ch38Lab1 class. Type:

```
public static void main(String[] args)
{
   Ch38Lab1 applet = new Ch38Lab1();
   JFrame frame = new JFrame();
   frame.setTitle("Chapter 38 Lab1");
   frame.add(applet, BorderLayout.CENTER);
   applet.init();
   applet.start();
   frame.setSize(400,320);
   frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   frame.setLocationRelativeTo(null); // Center the frame
   frame.setVisible(true);
 }
```

18. Close the Ch38Lab1 class.
19. Compile the program and fix any errors if necessary.
20. Compress all files into a single zip or rar file and submit.