BSCCS2005: Graded with Solutions
Week 12

1. Consider the code given below.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

class ButtonPanel extends JPanel{
    private JButton redButton;
    public ButtonPanel(){
        redButton = new JButton("Red");
        redButton.addActionListener(
            //CODE SEGMENT
        );
        add(redButton);
    }
}
class ButtonFrame extends JFrame implements WindowListener{
    private Container contentPane;
    public ButtonFrame(){
        setTitle("Button Demo");
        setSize(300, 200);
        addWindowListener(this);
        contentPane = this.getContentPane();
        contentPane.add(new ButtonPanel());
    }
    // define seven methods for implementing WindowListener
}
public class FClass{
    public static void main(String[] args) {
        EventQueue.invokeLater(
                () -> {
                    JFrame frame = new ButtonFrame();
                    frame.setVisible(true);
                }
                );
    }
}
```

`ActionListener` is a functional interface that has the abstract method `actionPerformed()`. Since `addActionListener` requires an instance of `ActionListener` as parameter, we can supply a lambda expression. Identify the appropriate option(s) to be written in place of `CODE-SEGMENT` such that the listener sets the panel background to red when the button is clicked.

√ `(ActionEvent evt) -> {`

```
                Color c = Color.red;
                setBackground(c);
                repaint();
            }
◯ () -> {
                Color c = Color.red;
                setBackground(c);
                repaint();
            }
√ ea -> {
                setBackground(Color.red);
                repaint();
            }
◯ () -> {
                this.setBackground(Color.red);
                repaint();
            }
```

Solution: The abstract method `actionPerformed()` accepts an argument of type `ActionEvent`. The lambda method without any arguments is not valid.

2. Consider the code given below.

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

class ButtonFrame extends JFrame implements ActionListener, WindowListener{
    private Container contentPane;
    private JPanel panel;
    private JButton colBtn;
    private int i = 0;
    private String[] colors = new String[]{"red", "yellow", "green"};
    public ButtonFrame(){
        //set the JFrame of a given size
        panel = new JPanel();
        colBtn = new JButton("Color");
        colBtn.setActionCommand(colors[i]);
        //add colBtn to panel, panel to the current contentPane
        colBtn.addActionListener(this);
        addWindowListener(this);
    }
    public void actionPerformed(ActionEvent evt){
        Color c;
        String s = evt.getActionCommand();
        if(s == "red")
            c = Color.red;
        else if(s == "yellow")
            c = Color.yellow;
        else
            c = Color.green;
        panel.setBackground(c);
        panel.repaint();
        i = (i + 1) % 3;
        colBtn.setActionCommand(colors[i]);
    }
    // define seven methods for implementing WindowListener
}
public class FClass{
    public static void main(String[] args) {
        EventQueue.invokeLater(
                () -> {
                    JFrame frame = new ButtonFrame();
                    frame.setVisible(true);
                }
```

```
        );
    }
}
```

Choose the correct option regarding the code.

- ◯ The initial color of the panel is red and it remains unchanged with the button clicks.
- ◯ The initial color of the panel is red and it toggles between red and yellow with the button clicks.
- √ The initial color of the panel is red, with next button click it becomes yellow, with one more button click it becomes green, and it goes on repeating.
- ◯ The initial color of the panel is red, with next button click it becomes yellow, with one more button click it becomes green, and followed by any arbitrary color.

Solution: The color of the panel is decided by the strings in `String[] colors = new String[]"red", "yellow", "green";`.
Initially the color is `colors[0]` which is red.
For each button click `i` becomes `i = (i + 1) % 3;`, i.e. $1, 2, 0, 1, 2, 0, 1, 2, \cdots$. Thus, the colors becomes "red", "yellow", "green", repeatedly.

3. Consider the Java program given below.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class GUITest extends JFrame implements ActionListener{
    JButton b1,b2,b3;
    JPanel panel;
    public GUITest(){
        panel=new JPanel();
        b1=new JButton("Red");
        b2=new JButton("Green");
        b3=new JButton("Blue");
        b1.addActionListener(this);
        b2.addActionListener(this);
        b3.addActionListener(this);
        panel.add(b1);
        panel.add(b2);
        panel.add(b3);
        add(panel,"South");
        setVisible(true);
        setSize(400,400);
    }
    public void actionPerformed(ActionEvent e) {
        ----------------------
        *****CODE SEGMENT*****
        ----------------------
    }
    public static void main(String[] args){
        new GUITest();
    }
}
```

Choose the correct code segment inside method `actionPerformed()` such that whenever either of the three buttons (Red/Green/Blue) is clicked, the panel background color changes accordingly.

◯
```
if(e.equals(b1))
    panel.setBackground(Color.red);
if(e.equals(b2))
    panel.setBackground(Color.green);
if(e.equals(b3))
    panel.setBackground(Color.blue);
```

√
```
if(e.getSource().equals(b1))
```

```
    panel.setBackground(Color.red);
if(e.getSource().equals(b2))
    panel.setBackground(Color.green);
if(e.getSource().equals(b3))
    panel.setBackground(Color.blue);
```

◯
```
if(e.getSource().equals("Red"))
    panel.setBackground(Color.red);
if(e.getSource().equals("Green"))
    panel.setBackground(Color.green);
if(e.getSource().equals("Blue"))
    panel.setBackground(Color.blue);
```

◯
```
if(e.equals("Red"))
    panel.setBackground(Color.red);
if(e.equals("Green"))
    panel.setBackground(Color.green);
if(e.equals("Blue"))
    panel.setBackground(Color.blue);
```

Solution:

4. Consider the code given below.

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

class ColorThread extends Thread{
    JPanel inputPanel;
    Color inputCol;
    Thread wTh;
    public ColorThread(JPanel ip, Color col, Thread th) {
        inputPanel = ip;
        inputCol = col;
        wTh = th;
    }
    public void run() {
        try {
            if(wTh != null)
                wTh.join();
            inputPanel.setBackground(inputCol);
            sleep(1000);
        }catch(InterruptedException e) {}
    }
}
public class FClass implements ActionListener{
    JFrame frm;
    JPanel inputPanel;
    JButton btnStart;
    FClass(){
        frm = new JFrame("Traffic Light");
        frm.setSize(200, 200);
        btnStart = new JButton("Start");
        btnStart.addActionListener(this);
        inputPanel = new JPanel();
        inputPanel.add(btnStart);
        frm.add(inputPanel);
        frm.setVisible(true);
    }
    public void actionPerformed(ActionEvent e) {
        Thread th1 = new ColorThread(inputPanel, Color.red, null);
        Thread th2 = new ColorThread(inputPanel, Color.yellow, th1);
        Thread th3 = new ColorThread(inputPanel, Color.green, th2);
        th1.start();
        th2.start();
```

```
        th3.start();
    }
    public static void main(String[] args){
        new FClass();
    }
}
```

Choose the option that correctly describes what happens if the button labelled "Start" is clicked.

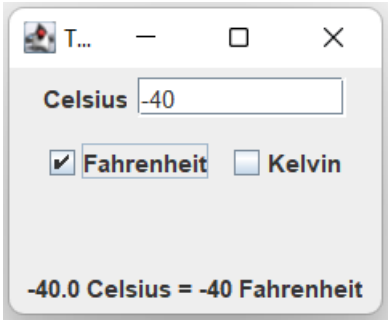
- √ The background color of panel `inputPanel` becomes red first, followed by yellow, further followed by green.
- ◯ The background color of panel `inputPanel` becomes red first, followed by yellow, further followed by green. The same sequence is repeated until the program terminates.
- ◯ The background color of panel `inputPanel` becomes red, yellow and green. However, the sequence in which the colors get rendered cannot be predicted.
- ◯ The background of panel `inputPanel` takes only one color. It becomes either red or yellow or green.

5. Consider the code given below.

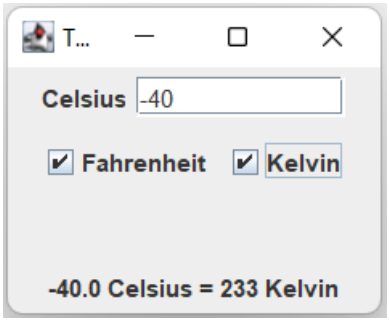
```
import javax.swing.*;
import java.awt.event.*;
public class FClass implements ActionListener{
    JFrame frm;
    JLabel lblCel;
    JTextField txtCel;
    JCheckBox chbFahr, chbKelv;
    JLabel lblMsg;
    FClass(){
        frm = new JFrame("Temperature conversion");
        frm.setSize(300, 200);
        lblCel = new JLabel("Celsius");
        txtCel = new JTextField(10);
        chbFahr = new JCheckBox("Fahrenheit");
        chbKelv = new JCheckBox("Kelvin");
        chbFahr.addActionListener(this);
        chbKelv.addActionListener(this);
        lblMsg = new JLabel();

        //add inputPanel, outputPanel and btnPanel to
        //the "North", "Bottom" and "Center" of the JFrame
        //add lblCel and txtCel to inputPanel
        //add lblMsg to outputPanel
        //add chbFahr and chbKelv to btnPanel
        frm.setVisible(true);
    }
    public static void main(String[] args){
        new FClass();
    }
    public void actionPerformed(ActionEvent e){
        if(chbFahr.isSelected()){
            double valCels = Double.parseDouble(txtCel.getText());
            int valFahr = (int)(((valCels * 9) / 5) + 32);
            lblMsg.setText(valCels + " Celsius = "+ valFahr + " Fahrenheit");
        }
        if(chbKelv.isSelected()){
            double valCels = Double.parseDouble(txtCel.getText());
            int valKelv = (int)(valCels + 273);
            lblMsg.setText(valCels + " Celsius = "+ valKelv + " Kelvin");
        }
        if(chbFahr.isSelected() && chbKelv.isSelected()){
            lblMsg.setText("Invalid option");
```

```
        }
    }
}
```

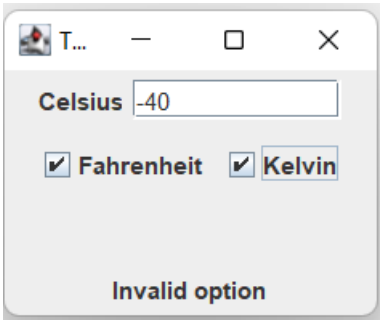
Which of the possible GUI(s) is/are generated by the given code.



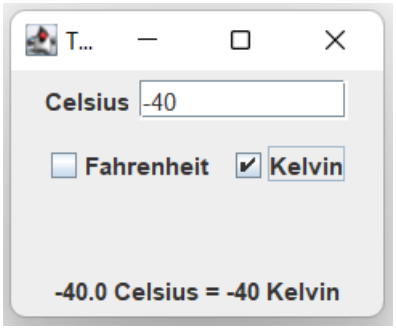
√



○



√

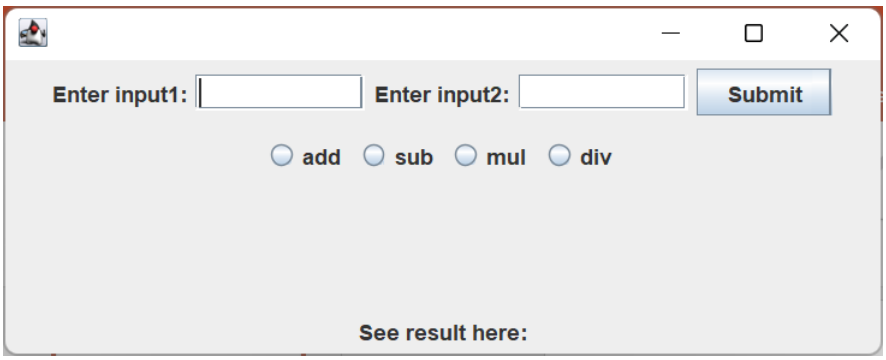


○

6. Consider the Java program given below.

```
import javax.swing.*;
import java.awt.*;
public class Calculator extends JFrame{
    JPanel inputPanel,outputPanel,opPanel;
    JLabel label1,label2,label3;
    JRadioButton add,sub,mul,div;
    JTextField input1,input2;
    JButton button;
    public Calculator() {
        label1=new JLabel("Enter input1:");
        label2=new JLabel("Enter input2:");
        input1=new JTextField(10);
        input2=new JTextField(10);
        button=new JButton("Submit");
        inputPanel=new JPanel();
        inputPanel.add(label1);
        inputPanel.add(input1);
        inputPanel.add(label2);
        inputPanel.add(input2);
        inputPanel.add(button);
        //LINE 1
        add=new JRadioButton("add");
        sub=new JRadioButton("sub");
        mul=new JRadioButton("mul");
        div=new JRadioButton("div");
        opPanel=new JPanel();
        opPanel.add(add);
        opPanel.add(sub);
        opPanel.add(mul);
        opPanel.add(div);
        //LINE 2
        label3=new JLabel("See result here:");
        outputPanel=new JPanel();
        outputPanel.add(label3);
        //LINE 3
        setVisible(true);
        setSize(500,200);
    }
    public static void main(String[] args) {
        new Calculator();
    }
}
```

Choose the correct options for `LINE` 1, 2 and 3, such that the above program produces the GUI given below:



◯
```
LINE 1:add(inputPanel,"Center");
LINE 2:add(opPanel,"North");
LINE 3:add(outputPanel,"South");
```

◯
```
LINE 1:add(inputPanel,"North");
LINE 2:add(opPanel,"South");
LINE 3:add(outputPanel,"Center");
```

√
```
LINE 1:add(inputPanel,"North");
LINE 2:add(opPanel,"Center");
LINE 3:add(outputPanel,"South");
```

◯
```
LINE 1:add(inputPanel,"South");
LINE 2:add(opPanel,"Center");
LINE 3:add(outputPanel,"North");
```

Solution: To obtain the GUI given, we should add `inputPanel` to the `South`, `opPanel` to the `Center` and `outputPanel` to the `North` of the frame.