BSCCS2005: Practice with Solutions
Week 12

1. Consider the code given below.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class FClass implements ActionListener{
    JFrame frm;
    JTextField inText, outText;
    JButton cmBtn, ftBtn;
    JLabel lblIn, lblOut;
    JPanel inputPanel, outputPanel, btnPanel;
    FClass(){
        frm = new JFrame("Height Converter");
        frm.setSize(340, 140);
        lblIn = new JLabel("Height (inch)");
        inText = new JTextField(10);
        lblOut = new JLabel("Output");
        outText = new JTextField(10);
        cmBtn = new JButton("Convert to CM");
        ftBtn = new JButton("Convert to FT");
        cmBtn.setActionCommand("cm");
        ftBtn.setActionCommand("ft");
        cmBtn.addActionListener(this);
        ftBtn.addActionListener(this);

        //add inputPanel, outputPanel and btnPanel to
        //the "North", "Center" and "Bottom" of the JFrame

        //add lblIn and inText to inputPanel
        //add lblOut and outText to outputPanel
        //add cmBtn and ftBtn to btnPanel
        frm.setVisible(true);
    }
    public void actionPerformed(ActionEvent e) {
        //code-segment: to be added
    }
    public static void main(String[] args){
        new FClass();
    }
}
```

Choose the option that has correct implementation of the method `actionPerformed` such that it converts a height given in inches to cm by clicking the button labelled “Convert to CM” and to feet (ft) by clicking the button labelled “Convert to FT”.

◯
```
double in = Double.parseDouble(inText.getText());
JButton btn = e.getSource();
double out = 0.0;
if(btn.equals(ftBtn))
    out = in * 0.0833333;
else if(btn.equals(cmBtn))
    out = in * 2.54;
outText.setText(out + "");
```

✓
```
double in = Double.parseDouble(inText.getText());
JButton btn = (JButton)e.getSource();
double out = 0.0;
if(btn.equals(ftBtn))
    out = in * 0.0833333;
else if(btn.equals(cmBtn))
    out = in * 2.54;
outText.setText(out + "");
```

◯
```
double in = Double.parseDouble(inText.getText());
String s = e.getActionCommand();
double out = 0.0;
if(s.equals(ftBtn))
    out = in * 0.0833333;
else if(s.equals(cmBtn))
    out = in * 2.54;
outText.setText(out + "");
```

✓
```
double in = Double.parseDouble(inText.getText());
String s = e.getActionCommand();
double out = 0.0;
if(s.equals("ft"))
    out = in * 0.0833333;
else if(s.equals("cm"))
    out = in * 2.54;
outText.setText(out + "");
```

2. Consider the Java program given below.

```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
public class GUITest extends JFrame implements WindowListener{
    JLabel label;
    Container contentPane;
    String str="";
    public GUITest() {
        label=new JLabel();
        addWindowListener(this);
        contentPane = this.getContentPane();
        contentPane.add(label);
        setVisible(true);
        setSize(700,200);
    }
    public void windowOpened(WindowEvent e) {
        str+="Window Opened ";
        label.setText(str);
    }
    public void windowClosing(WindowEvent e) {
        str+="Window Closing ";
        label.setText(str);
    }
    public void windowClosed(WindowEvent e) {
    }
    public void windowIconified(WindowEvent e) {
        str+="Window Iconified ";
        label.setText(str);
    }
    public void windowDeiconified(WindowEvent e) {
        str+="Window Deiconified ";
        label.setText(str);
    }
    public void windowActivated(WindowEvent e) {
        str+="Window Activated";
        label.setText(str);
    }
    public void windowDeactivated(WindowEvent e) {
        str+="Window Deactivated ";
        label.setText(str);
    }
    public static void main(String[] args) {
        new GUITest();
    }
}
```

Method description:

`void windowOpened(WindowEvent e):`
Invokes the first time a window is made visible.

`void windowClosing(WindowEvent e):`
Invoked when the user attempts to close the window.

`void windowClosed(WindowEvent e):`
Invoked when a window has been closed.

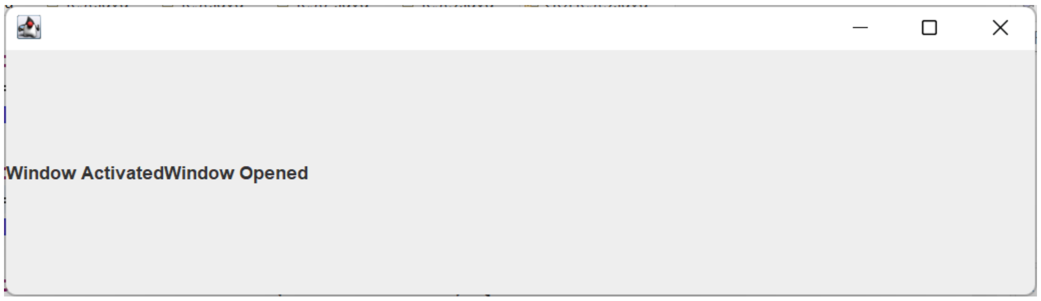
`void windowIconified(WindowEvent e):`
Invoked when a window is changed from a normal to a minimized state.

`void windowDeiconified(WindowEvent e):`
Invoked when a window is changed from a minimized to a normal state.

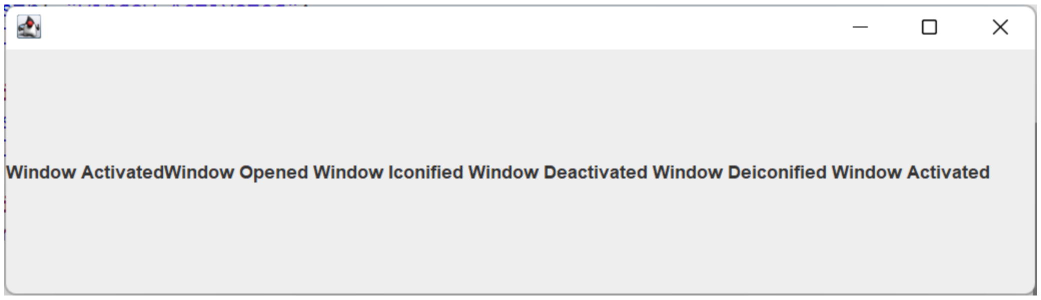
`void windowActivated(WindowEvent e):`
Invoked when the window is set to be the active window.

`void windowDeactivated(WindowEvent e):`
Invoked when a window is no longer the active window.

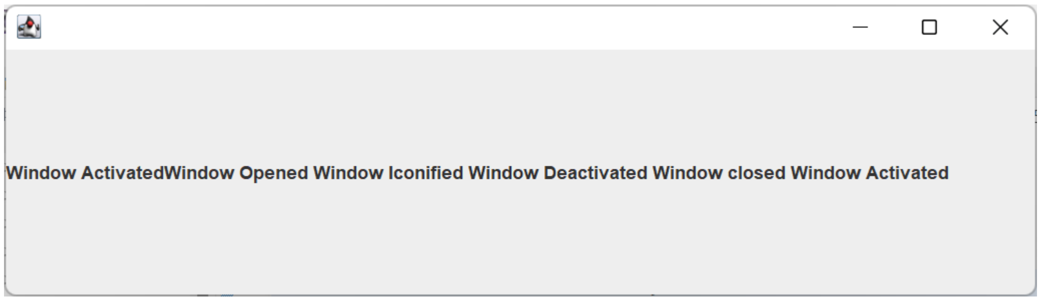
Which of the following possible GUIs generated by the given code:

√ 

```
Window ActivatedWindow Opened
```

✓ 

```
Window ActivatedWindow Opened Window Iconified Window Deactivated Window Deiconified Window Activated
```

○ 

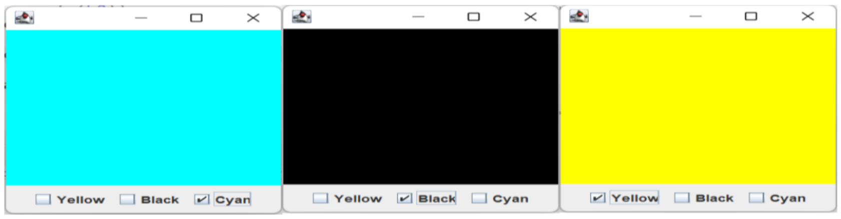
```
Window ActivatedWindow Opened Window Iconified Window Deactivated Window closed Window Activated
```

○ All of the above.

Solution:

3. Consider the Java program given below.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class GUITest extends JFrame implements ActionListener{
    JCheckBox b1,b2,b3;
    JPanel panel1,panel2;
    public GUITest(){
        panel1=new JPanel();
        panel2=new JPanel();
        b1=new JCheckBox("Yellow");
        b2=new JCheckBox("Black");
        b3=new JCheckBox("Cyan");
        b1.addActionListener(this);
        b2.addActionListener(this);
        b3.addActionListener(this);
        panel1.add(b1);
        panel1.add(b2);
        panel1.add(b3);
        add(panel1,"South");
        add(panel2,"Center");
        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 checkboxes (Yellow/Black/Cyan) is clicked on panel1, background color of panel2 changes accordingly.

[✓ `if(e.getSource().equals(b1))`]

```
√ if(e.getSource().equals(b1))
      panel2.setBackground(Color.yellow);
  if(e.getSource().equals(b2))
      panel2.setBackground(Color.black);
  if(e.getSource().equals(b3))
      panel2.setBackground(Color.cyan);
```

```
○ if(e.getSource().equals(b1))
      panel1.setBackground(Color.yellow);
  if(e.getSource().equals(b2))
      panel1.setBackground(Color.black);
  if(e.getSource().equals(b3))
      panel1.setBackground(Color.cyan);
```

```
○ if(e.equals(b1))
      panel2.setColor(Color.yellow);
  if(e.equals(b2))
       panel2.setColor(Color.black);
  if(e.equals(b3))
       panel2.setColor(Color.cyan);
```

```
○ if(e.getSource().equals(b1))
      panel2.setColor(Color.yellow);
  if(e.getSource().equals(b2))
      panel2.setColor(Color.black);
  if(e.getSource().equals(b3))
      panel2.setColor(Color.cyan);
```

Solution:

4. Consider the Java program given below.

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class GUITest extends JFrame implements KeyListener{
    JLabel label;
    Container contentPane;
    public GUITest() {
        label=new JLabel();
        addKeyListener(this);
        contentPane = this.getContentPane();
        contentPane.add(label);
        setVisible(true);
        setSize(200,200);
    }
    public void keyTyped(KeyEvent e) {
        label.setText("KeyTyped");
    }
    public void keyPressed(KeyEvent e) {
        label.setText("keyPressed");
    }
    public void keyReleased(KeyEvent e) {
        label.setText("keyReleased");
    }
    public static void main(String[] args) {
        new GUITest();
    }
}
```

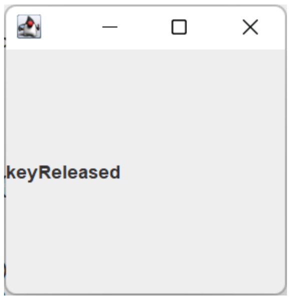
Method description:

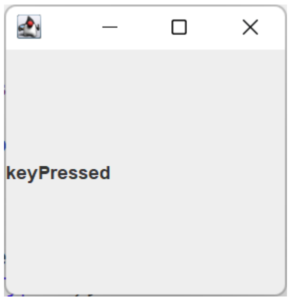
`void keyTyped(KeyEvent e):`
Invoked when a key has been typed.

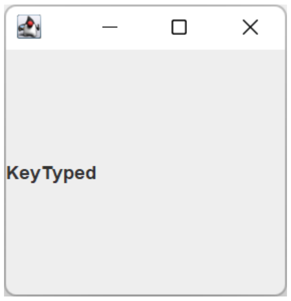
`void keyPressed(KeyEvent e):`
Invoked when a key has been pressed.

`void keyReleased(KeyEvent e):`
Invoked when a key has been released

Which of the following possible GUIs generated by the given code:

○ 
keyReleased

○ 
keyPressed

○ 
KeyTyped

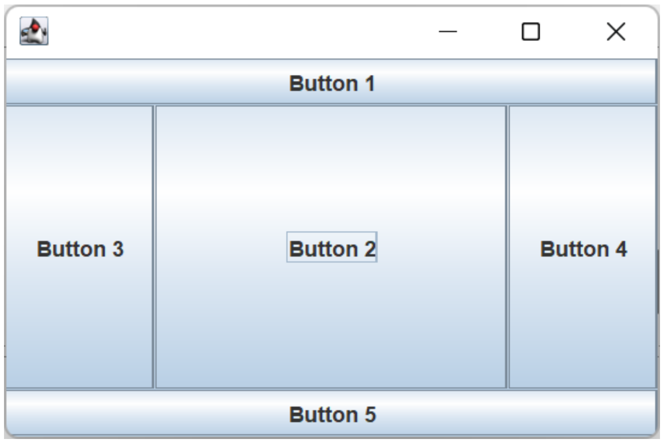
[✓ All of the above.]

Solution:

5. Consider the Java program given below.

```
import javax.swing.*;
import java.awt.*;
public class GUITest extends JFrame{
    JButton b1,b2,b3,b4,b5;
    LayoutManager manager;
    public GUITest(){
        //LINE 1
        setLayout(manager);
        b1=new JButton("Button 1");
        b2=new JButton("Button 2");
        b3=new JButton("Button 3");
        b4=new JButton("Button 4");
        b5=new JButton("Button 5");
        add(b1,"North");
        add(b2,"Center");
        add(b3,"West");
        add(b4,"East");
        add(b5,"South");
        setVisible(true);
        setSize(400,400);
    }
    public static void main(String[] args){
        new GUITest();
    }
}
```

Choose the correct option such that the code produces the output given below:



[✓ `manager=new BorderLayout();`]

○ `manager=new BorderLayout(6);`

○ `manager=new FlowLayout();`

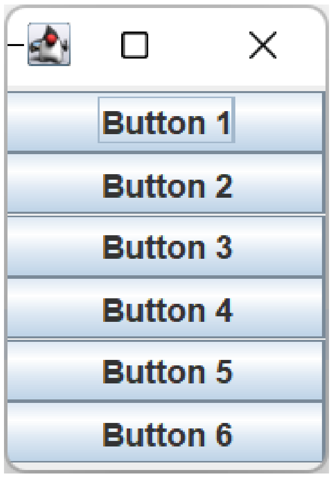
○ `manager=new GridLayout();`

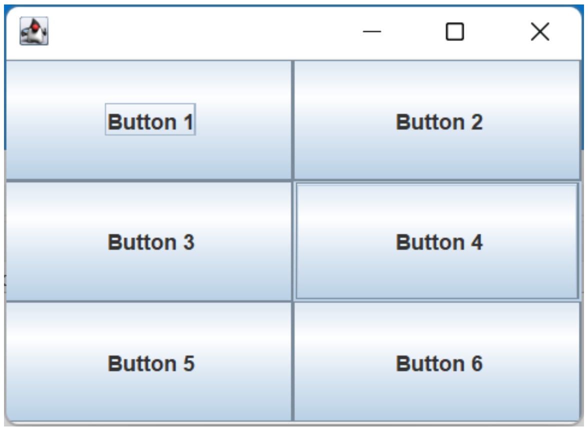
Solution: In above program all buttons are arranged to the frame/window using BorderLayout.

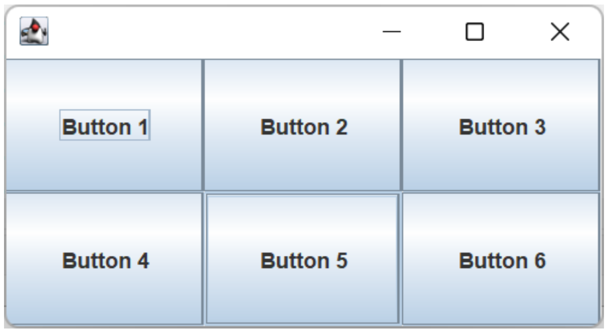
6. Consider the Java program given below.

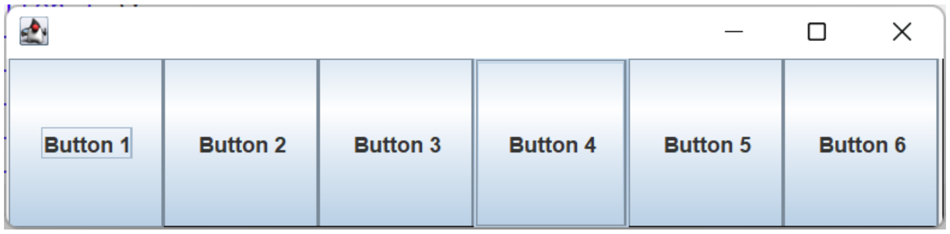
```
import javax.swing.*;
import java.awt.*;
public class GUITest extends JFrame{
    JButton b1,b2,b3,b4,b5,b6;
    LayoutManager manager;
    public GUITest(){
        manager=new GridLayout(3,2);
        setLayout(manager);
        b1=new JButton("Button 1");
        b2=new JButton("Button 2");
        b3=new JButton("Button 3");
        b4=new JButton("Button 4");
        b5=new JButton("Button 5");
        b6=new JButton("Button 6");
        add(b1);
        add(b2);
        add(b3);
        add(b4);
        add(b5);
        add(b6);
        setVisible(true);
        setSize(400,400);
    }
    public static void main(String[] args){
        new GUITest();
    }
}
```

What will the output be?

○ 

✓ 

○ 

○ 

Solution: The program uses GridLayout with 2 rows and 2 columns to arrange the buttons.