**T323  
ICT104 Program Design and Development**

**Week 8 Tutorial**

**Topic: Object Oriented Concepts**

**Objective: Students learn about the basics of GUI applications using Java**

**Submission: The completed tutorial word file containing student’s answers need to be  
uploaded to Moodle by Monday 08/01/2024 @ 23.59**

**Part A:** Highlight the correct option.

1. In Swing, labels are created with this class:

A) JFCLabel

B) AWTLabel

C) JLabel

D) SwingLabel

2. The GridLayout manager limits each cell to only one component. To put two or more components in a cell, do this:

A) Resize the cells so they can hold more

B) You can nest panels inside the cells, and add other components to the panels

C) The statement is false. The GridLayout manager does not have this restriction

D) Resize the components to fit in the cell

3. If panel references a JPanel object, which of the following statements adds the GridLayout to it?

A) panel.setLayout(new (GridLayout(2,3));

B) panel.addLayout(new (GridLayout(2,3));

C) panel.GridLayout(2,3);

D) panel.attachLayout(GridLayout(2,3));

4. This layout manager arranges components in regions named North, South, East, West, and Center.

A) GridLayout

B) BorderLayout

C) FlowLayout

D) RegionLayout

5. When using the BorderLayout manager, how many components can each region hold?

A) 1

B) 2

C) 5

D) No limit

6. When a component is added to a region in the BorderLayout manager:

A) the component retains its original size

B) it results in a compile time error if it is too large

C) the component is stretched so it fills up the entire region

D) the region is resized to fit the component

7. When adding components to a container that is governed by the GridLayout manager:

A) you cannot specify a cell

B) you specify the cell with the row and column numbers in the add statement

C) you must add them starting with the lower, right cell

D) the components are added automatically by filling up the first column, then the second, etc.

**Part B:** State TRUE or FALSE:

1. The following statement adds the FlowLayout manager to the container, centers the components, and separates the components with a gap of 10 pixels.

setLayout(new FlowLayout());

1. The ActionEvent argument that is passed to an action listener's actionPerformed method is the event object that was generated in response to an event.

**Part C:**

**Question 1:** Revision topics for preparation of Quiz 2. Write short note on:

1. Inheritance
2. Interfaces
3. Abstract methods

**Question 2:** Identify the **output** of the following Java code. **Justify** the syntax.

File Name: ColorCheckBoxWindow.java (it can be downloaded from **Moodle** under **Week 8 Lecture 8 T323 Resources**)

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

/\*\*

The ColorCheckBoxWindow class demonstrates how check boxes

can be used.

\*/

public class ColorCheckBoxWindow extends JFrame

{

private JLabel messageLabel; // A message to the user

private JCheckBox yellowCheckBox; // To select yellow background

private JCheckBox redCheckBox; // To select red foreground

private final int WINDOW\_WIDTH = 300; // Window width

private final int WINDOW\_HEIGHT = 100; // Window height

/\*\*

Constructor

\*/

public ColorCheckBoxWindow()

{

// Set the text for the title bar.

setTitle("Color Check Boxes");

// Set the size of the window.

setSize(WINDOW\_WIDTH, WINDOW\_HEIGHT);

// Specify an action for the close button.

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// Create a label.

messageLabel = new JLabel("Select the check " +

"boxes to change colors.");

// Create the check boxes.

yellowCheckBox = new JCheckBox("Yellow background");

redCheckBox = new JCheckBox("Red foreground");

// Add an item listener to the check boxes.

yellowCheckBox.addItemListener(new CheckBoxListener());

redCheckBox.addItemListener(new CheckBoxListener());

// Add a FlowLayout manager to the content pane.

setLayout(new FlowLayout());

// Add the label and check boxes to the content pane.

add(messageLabel);

add(yellowCheckBox);

add(redCheckBox);

// Display the window.

setVisible(true);

}

/\*\*

Private inner class that handles the event when

the user clicks one of the check boxes.

\*/

private class CheckBoxListener implements ItemListener

{

public void itemStateChanged(ItemEvent e)

{

// Determine which check box was clicked.

if (e.getSource() == yellowCheckBox)

{

// Is the yellow check box selected? If so, we

// want to set the background color to yellow.

if (yellowCheckBox.isSelected())

{

// The yellow check box was selected. Set

// the background color for the content

// pane and the two check boxes to yellow.

getContentPane().setBackground(Color.yellow);

yellowCheckBox.setBackground(Color.yellow);

redCheckBox.setBackground(Color.yellow);

}

else

{

// The yellow check box was deselected. Set

// the background color for the content

// pane and the two check boxes to light gray.

getContentPane().setBackground(Color.lightGray);

yellowCheckBox.setBackground(Color.lightGray);

redCheckBox.setBackground(Color.lightGray);

}

}

else if (e.getSource() == redCheckBox)

{

// Is the red check box selected? If so, we want

// to set the forground color to red.

if (redCheckBox.isSelected())

{

// The red check box was selected. Set the

// foreground color for the label and the

// two check boxes to red.

messageLabel.setForeground(Color.red);

yellowCheckBox.setForeground(Color.red);

redCheckBox.setForeground(Color.red);

}

else

{

// The red check box was deselected. Set the

// foreground color for the label and the

// two check boxes to black.

messageLabel.setForeground(Color.black);

yellowCheckBox.setForeground(Color.black);

redCheckBox.setForeground(Color.black);

}

}

}

}

/\*\*

The main method creates an instance of the

ColorCheckBoxWindow class, displaying its window.

\*/

public static void main(String[] args)

{

new ColorCheckBoxWindow();

}

}

File Name:ColorWindow.java (it can be downloaded from **Moodle** under **Week 7 Lecture 7 T323 Resources**)

import javax.swing.\*; // Needed for Swing classes

import java.awt.\*; // Needed for Color class

import java.awt.event.\*; // Needed for event listener interface

/\*\*

This class demonstrates how to set the background color of

a panel and the foreground color of a label.

\*/

public class ColorWindow extends JFrame

{

private JLabel messageLabel; // To display a message

private JButton redButton; // Changes color to red

private JButton blueButton; // Changes color to blue

private JButton yellowButton; // Changes color to yellow

private JPanel panel; // A panel to hold components

private final int WINDOW\_WIDTH = 200; // Window width

private final int WINDOW\_HEIGHT = 125; // Window height

/\*\*

Constructor

\*/

public ColorWindow()

{

// Set the title bar text.

setTitle("Colors");

// Set the size of the window.

setSize(WINDOW\_WIDTH, WINDOW\_HEIGHT);

// Specify an action for the close button.

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// Create a label.

messageLabel = new JLabel("Click a button to " +

"select a color.");

// Create the three buttons.

redButton = new JButton("Red");

blueButton = new JButton("Blue");

yellowButton = new JButton("Yellow");

// Register an event listener with all 3 buttons.

redButton.addActionListener(new RedButtonListener());

blueButton.addActionListener(new BlueButtonListener());

yellowButton.addActionListener(new YellowButtonListener());

// Create a panel and add the components to it.

panel = new JPanel();

panel.add(messageLabel);

panel.add(redButton);

panel.add(blueButton);

panel.add(yellowButton);

// Add the panel to the content pane.

add(panel);

// Display the window.

setVisible(true);

}

/\*\*

Private inner class that handles the event when

the user clicks the Red button.

\*/

private class RedButtonListener implements ActionListener

{

public void actionPerformed(ActionEvent e)

{

// Set the panel's background to red.

panel.setBackground(Color.RED);

// Set the label's text to blue.

messageLabel.setForeground(Color.BLUE);

}

}

/\*\*

Private inner class that handles the event when

the user clicks the Blue button.

\*/

private class BlueButtonListener implements ActionListener

{

public void actionPerformed(ActionEvent e)

{

// Set the panel's background to blue.

panel.setBackground(Color.BLUE);

// Set the label's text to yellow.

messageLabel.setForeground(Color.YELLOW);

}

}

/\*\*

Private inner class that handles the event when

the user clicks the Yellow button.

\*/

private class YellowButtonListener implements ActionListener

{

public void actionPerformed(ActionEvent e)

{

// Set the panel's background to yellow.

panel.setBackground(Color.YELLOW);

// Set the label's text to black.

messageLabel.setForeground(Color.BLACK);

}

}

/\*\*

main method

\*/

public static void main(String[] args)

{

new ColorWindow();

}

}

You will have **MCQ Quiz B** test in this week's (**Week 8**) **tutorial**.

It will be conducted during your **Week 8 tutorial class** in presence of **your tutor**.

It is worth **15%**. It has **15 MCQs** worth **1 mark each** for a duration of **30 minutes**. It covers topics from **Weeks 1 to 7 inclusive**.

It is a **single attempt** and students **cannot do it in their own time**. Please make sure to attend your **enrolled tutorial** class to do this quiz this week.

Wishing you all the best.