Module: Mobile Application development (Android)

Session 53: Assessment

Trainer can conduct online/offline assessment using following problem statements

**Problem Statement 1 - Tip Calculator**

Implement a simple *Tip Calculator* using an interface similar to the layout suggested below. The calculator uses *EditText* boxes to enter: the check’s amount, tip percent (using a default of 15%), and the number of people

to pay.

Pushing the GO *button* calculates the details (exposed in two side‐by‐side *TexView* boxes). The first box shows:Total Tip, and Total to Pay. The second box displays: Total per person and Tip per person.

The input is supplied from the phone’s hardware keyboard (as we learn more about Android’s features, this crude design will be improved).

**HINT**: Use nested Linear Layouts (as discussed in class) to produce the UI for this assignment.



**Problem Statement 2 - Tic-Tac-Toe game**

In this assessment you are asked to re-engineer a Tic-Tac-Toe game in such a way that it can be deployed in your Android phone (or emulator). The assessment presents two challenges: (1) you need to produce an Android UI that resembles the common board for the game, and (2) you need to produce new code or adapt existing Java solutions of the problem. Your UI should include two additional buttons: “QUIT” and “NEW GAME”.

The following public domain Java implementation of the game was found as the first output of a Google search on the “Tic-tac-toe” subject. You may use it as starting point for your assignment. You may also write your own or choose another version of the code (if so, please include the references).

<http://forum.codecall.net/java-tutorials/2141-java-tutorial-tic-tac-toe.html>

|  |
| --- |
| package mytictactoe;  import java.awt.\*;  import java.awt.event.\*;  import javax.swing.\*;  public class TicTacToeV1 implements ActionListener {  /\*Instance Variables\*/  private JFrame window = new JFrame("Tic-Tac-Toe");  private JButton button1 = new JButton("");  private JButton button2 = new JButton("");  private JButton button3 = new JButton("");  private JButton button4 = new JButton("");  private JButton button5 = new JButton("");  private JButton button6 = new JButton("");  private JButton button7 = new JButton("");  private JButton button8 = new JButton("");  private JButton button9 = new JButton("");  private String letter = "";  private int count = 0;  private boolean win = false;  public TicTacToeV1(){  /\*Create Window\*/  window.setSize(300,300);  window.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  window.setLayout(new GridLayout(3,3));  /\*Add Buttons To The Window\*/  window.add(button1);  window.add(button2);  window.add(button3);  window.add(button4);  window.add(button5);  window.add(button6);  window.add(button7);  window.add(button8);  window.add(button9);  /\*Add The Action Listener To The Buttons\*/  button1.addActionListener(this);  button2.addActionListener(this);  button3.addActionListener(this);  button4.addActionListener(this);  button5.addActionListener(this);  button6.addActionListener(this);  button7.addActionListener(this);  button8.addActionListener(this);  button9.addActionListener(this);  /\*Make The Window Visible\*/  window.setVisible(true);  }  public void actionPerformed(ActionEvent a) {  count++;  /\*Calculate Who's Turn It Is\*/  if(count == 1 || count == 3 || count == 5 || count == 7 || count ==  9){  letter = "X";  } else if(count == 2 || count == 4 || count == 6 || count == 8 ||  count == 10){  letter = "O";  }  /\*Display X's or O's on the buttons\*/  if(a.getSource() == button1){  button1.setText(letter);  button1.setEnabled(false);  } else if(a.getSource() == button2){  button2.setText(letter);  button2.setEnabled(false);  } else if(a.getSource() == button3){  button3.setText(letter);  button3.setEnabled(false);  } else if(a.getSource() == button4){  button4.setText(letter);  button4.setEnabled(false);  } else if(a.getSource() == button5){  button5.setText(letter);  button5.setEnabled(false);  } else if(a.getSource() == button6){  button6.setText(letter);  button6.setEnabled(false);  } else if(a.getSource() == button7){  button7.setText(letter);  button7.setEnabled(false);  } else if(a.getSource() == button8){  button8.setText(letter);  button8.setEnabled(false);  } else if(a.getSource() == button9){  button9.setText(letter);  button9.setEnabled(false);  }  /\*Determine who won\*/  //horizontal wins  if( button1.getText() == button2.getText() && button2.getText() ==  button3.getText() && button1.getText() != ""){  win = true;  }  else if(button4.getText() == button5.getText() && button5.getText()  == button6.getText() && button4.getText() != ""){  win = true;  }  else if(button7.getText() == button8.getText() && button8.getText()  == button9.getText() && button7.getText() != ""){  win = true;  }  //virticle wins  else if(button1.getText() == button4.getText() && button4.getText()  == button7.getText() && button1.getText() != ""){  win = true;  }  else if(button2.getText() == button5.getText() && button5.getText()  == button8.getText() && button2.getText() != ""){  win = true;  }  else if(button3.getText() == button6.getText() && button6.getText()  == button9.getText() && button3.getText() != ""){  win = true;  }  //diagonal wins  else if(button1.getText() == button5.getText() && button5.getText()  == button9.getText() && button1.getText() != ""){  win = true;  }  else if(button3.getText() == button5.getText() && button5.getText()  == button7.getText() && button3.getText() != ""){  win = true;  }  else {  win = false;  }  /\*Show a dialog if someone wins or the game is tie\*/  if(win == true){  JOptionPane.showMessageDialog(null, letter + " WINS!");  } else if(count == 9 && win == false){  JOptionPane.showMessageDialog(null, "Tie Game!");  }  }  public static void main(String[] args){  new TicTacToeV1();  }  } |

**Problem Statement 3 - TODO LIST Application**

Write an Android App to implement a simple TODO list. The app displays a *ListView* showing a numbered sequence of lines representing tasks that you need to perform.

The app is controlled by an OPTION MENU offering the following possibilities:

1. Add new entry to the list

2. Delete a task from the list (ask for number and remove)

3. Update the text of a selected task.

4. Save TODO list (fake operation – we still have not done file IO)

5. Close app

The main layout of the TODO list should be similar to the following:



**Functionality**:

1. Type a new task in the top EditText box. Press **Menu** button, click on **Add entry.**

2. Click on a line. The text should be displayed on the EditText box.

a. Write over the text, click **Menu** > **Update entry** to modify the TODO list value (clear EditText)

b. Click on **Menu** > **Delete entry** to remove the task from the list (clear EditText).

3. Click on **Save List** to *fake* the writing of the list to disk

4. Click on **Close** to save list and finish application.

**Problem Statement 4 - Multithreading Problem**

In this assessment you will create an Android application that relies on two background threads to accomplish its goals. The main activity waits for the threads to return an agreeable ‘magic’ number. A magic number is a four digits value that either (1) is a multiple of seven or (2) is a multiple of four and ‘2’ is its last digit. The main steps in the application are

1. The main activity controls the UI presentation, while waiting it shows a ‘rotating’ progress bar to suggest delay.

2. Each background thread does the following: (a) sleep for 1 second, (b) generate a random four-digits number, (3) if the number is ‘magic’ send a message to the main thread with the calculated value, otherwise repeat the cycle (sleep, …)

3. When the main activity receives the message containing a magic number it stops both background threads and displays the value of the magic number.

**Note.** Your implementation should use a message-passing Handler mechanism for the threads to communicate with the UI activity (request token, send message).