package onlinetestapp;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.ButtonGroup;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JRadioButton;

class OnlineTest extends JFrame implements ActionListener {

private static final long serialVersionUID = 1L;

JLabel label;

JRadioButton radioButton[] = new JRadioButton[5];

JButton btnNext, btnBookmark;

ButtonGroup bg;

int count = 0, current = 0, x = 1, y = 1, now = 0;

int m[] = new int[10];

// create jFrame with radioButton and JButton

OnlineTest(String s) {

super(s);

label = new JLabel();

add(label);

bg = new ButtonGroup();

for (int i = 0; i < 5; i++) {

radioButton[i] = new JRadioButton();

add(radioButton[i]);

bg.add(radioButton[i]);

}

btnNext = new JButton("Next");

btnBookmark = new JButton("Bookmark");

btnNext.addActionListener(this);

btnBookmark.addActionListener(this);

add(btnNext);

add(btnBookmark);

set();

label.setBounds(30, 40, 450, 20);

//radioButton[0].setBounds(50, 80, 200, 20);

radioButton[0].setBounds(50, 80, 450, 20);

radioButton[1].setBounds(50, 110, 200, 20);

radioButton[2].setBounds(50, 140, 200, 20);

radioButton[3].setBounds(50, 170, 200, 20);

btnNext.setBounds(100, 240, 100, 30);

btnBookmark.setBounds(270, 240, 100, 30);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLayout(null);

setLocation(250, 100);

setVisible(true);

setSize(600, 350);

}

// handle all actions based on event

public void actionPerformed(ActionEvent e) {

if (e.getSource() == btnNext) {

if (check())

count = count + 1;

current++;

set();

if (current == 9) {

btnNext.setEnabled(false);

btnBookmark.setText("Result");

}

}

if (e.getActionCommand().equals("Bookmark")) {

JButton bk = new JButton("Bookmark" + x);

bk.setBounds(480, 20 + 30 \* x, 100, 30);

add(bk);

bk.addActionListener(this);

m[x] = current;

x++;

current++;

set();

if (current == 9)

btnBookmark.setText("Result");

setVisible(false);

setVisible(true);

}

for (int i = 0, y = 1; i < x; i++, y++) {

if (e.getActionCommand().equals("Bookmark" + y)) {

if (check())

count = count + 1;

now = current;

current = m[y];

set();

((JButton) e.getSource()).setEnabled(false);

current = now;

}

}

if (e.getActionCommand().equals("Result")) {

if (check())

count = count + 1;

current++;

JOptionPane.showMessageDialog(this, "correct answers= " + count);

System.exit(0);

}

}

// SET Questions with options

void set() {

radioButton[4].setSelected(true);

if (current == 0) {

label.setText("Que1: Which of the following is not introduced with Java 8?");

radioButton[0].setText("Stream API");

radioButton[1].setText("Serialization");

radioButton[2].setText("Spliterator");

radioButton[3].setText("Lambda Expression");

}

if (current == 1) {

label.setText("Que2: Which feature of java 7 allows to not explicitly close IO resource?");

radioButton[0].setText("try catch finally");

radioButton[1].setText("IOException");

radioButton[2].setText("AutoCloseable");

radioButton[3].setText("Streams");

}

if (current == 2) {

label.setText("Que3: SessionFactory is a thread-safe object.");

radioButton[0].setText("true");

radioButton[1].setText("false");

radioButton[2].setText("don't know");

radioButton[3].setText("false");

}

if (current == 3) {

label.setText("Que4: Which is the new method introduced in java 8 to iterate over a collection?");

radioButton[0].setText("for (String i : StringList)");

radioButton[1].setText("foreach (String i : StringList)");

radioButton[2].setText("StringList.forEach()");

radioButton[3].setText("List.for()");

}

if (current == 4) {

label.setText("Que5: What is the substitute of Rhino javascript engine in Java 8?");

radioButton[0].setText(" Nashorn");

radioButton[1].setText("V8");

radioButton[2].setText("Inscript");

radioButton[3].setText("Narcissus");

}

if (current == 5) {

label.setText("Que6: How to read entire file in one line using java 8?");

radioButton[0].setText("Files.readAllLines()");

radioButton[1].setText("Files.read()");

radioButton[2].setText("Files.readFile()");

radioButton[3].setText("Files.lines()");

}

if (current == 6) {

label.setText("Que7: Which feature of java 7 allows to not explicitly close IO resource?");

radioButton[0].setText("try catch finally");

radioButton[1].setText("IOException");

radioButton[2].setText("AutoCloseable");

radioButton[3].setText("Streams");

}

if (current == 7) {

label.setText("Que8: Which of the following is not a core interface of Hibernate?");

radioButton[0].setText("Configuration");

radioButton[1].setText("Criteria");

radioButton[2].setText("SessionManagement");

radioButton[3].setText("Session");

}

if (current == 8) {

label.setText("Que9: SessionFactory is a thread-safe object.");

radioButton[0].setText("true");

radioButton[1].setText("false");

radioButton[2].setText("don't know");

radioButton[3].setText("false");

}

if (current == 9) {

label.setText("Que10: Which of the following is not a state of object in Hibernate?");

radioButton[0].setText("Attached()");

radioButton[1].setText("Detached()");

radioButton[2].setText("Persistent()");

radioButton[3].setText("Transient()");

}

label.setBounds(30, 40, 450, 20);

for (int i = 0, j = 0; i <= 90; i += 30, j++)

radioButton[j].setBounds(50, 80 + i, 200, 20);

}

// declare right answers.

boolean check() {

if (current == 0)

return (radioButton[1].isSelected());

if (current == 1)

return (radioButton[1].isSelected());

if (current == 2)

return (radioButton[0].isSelected());

if (current == 3)

return (radioButton[2].isSelected());

if (current == 4)

return (radioButton[0].isSelected());

if (current == 5)

return (radioButton[0].isSelected());

if (current == 6)

return (radioButton[1].isSelected());

if (current == 7)

return (radioButton[2].isSelected());

if (current == 8)

return (radioButton[0].isSelected());

if (current == 9)

return (radioButton[0].isSelected());

return false;

}

public static void main(String s[]) {

new OnlineTest("Online Test App");

}

}