**OOP2. Problem Set 5A: Menus and Frames, text fields and labels. Corresponds to**

**Section 5, Unit 13.**

1. The sample program JMenuFrame.java.

(i) This class is a subclass of what class from the Java API?

(ii) What interface does it implement?

(iii) What methods are contained in this interface?

(iv) What kinds of events are handled by the methods in this interface?

(v) What is the difference between a menubar and a menu?

(vi) Can a menu generate an ActionEvent? If not, what kind of menu-related

object can generate such an event?

(vii) Why is it ok to declare ‘menubar’ locally within the constructor, but

‘response’ must be declared as a class attribute? Investigate what happens if

you move the declaration of ‘response’ inside the constructor. Can you

explain the result?

(viii) What method is called when you click on the menu item ‘Cut’?

(ix) Examine the ActionEvent class, and see what methods are available for use

within the actionPerformed() method in connection with ‘event’

2. Write a Frame-based class with a menu bar and one menu called Friends, containing an

item for each of 3 friends. Nothing should happen for the moment when you click on a

friend’s name. You’ll need a main() to create and display the Frame. Set a different

background colour for each friend.

3. Extend your Frame class so that, when you click on a friend’s name, a message saying “I

really like whoever” pops up, where the relevant name appears instead of whoever.

4. You should already have a class called Book, as your solution to problem sheet 2 exercise

2. Write a Frame-based class called BookScreen which will have a menu bar and two

menus. The first menu should be called ‘File’ and carry only one item for the moment:

‘Quit’. The second menu should be called ‘Book’ and carry two items: Add and Display.

The Frame should have the title ‘Booklist”. Include a driver main() to create and display

this Frame. Declare a book object myBook as a class attribute. If you click on ‘Add’, you

should be asked to enter details for a book, then myBook should be created with these

details. If you click on display, the details of myBook should be displayed If you click

on ‘quit’, your program should display a dialog to say the system is finishing, then it

should exit.

5. The sample program TextFieldLabelExample. Compile and run this program.

(i) When you enter a name and press return, where is the name displayed?

(ii) What is displayed if you press ‘OK’?

(iii) What if you press Cancel?

(iv) How many GUI objects are declared as attributes? Can you identify them all

on screen?

(v) What is the main difference between a text field and a label?

(vi) What would happen if you removed the statement

inputLine.addActionListener(this); ? Try it and see.  
(vii) What would happen if you moved all the statements which start with

cPane.add( down to the end of the constructor? Would the program still

run as before? What if you changed the order of these statements? Now

move them all up to the top of the constructor: what happens now? Can you

explain this?

(viii) In the actionPerformed() method, we find the code

if (e.getSource() == okButton) where ‘e’ is an

ActionEvent. Examine the ActionEvent class, find out what class contains

the definition of getSource() and what type of thing it returns. What would

happen if you rewrote the line as

if (e.getSource().equals(okButton))?

6. (i) Copy the class MyJFrame1 and examine it:

 Why is there no dot notation used on the methods: setTitle(), setSize(),

setDefaultCloseOperation(), and getContentPane()?

 Why is the dot notation used with the method setBackground()?

Predict what will happen when this class is compiled and run. Compile and run to see if your

prediction is correct.

(ii) Copy MyJFrame2 and examine it. Predict what will happen when you run this program.

Compile and run it. Without changing the size of the window or the button, position the button so

that it is completely within the window.

(iii) It would be more interesting if something happened when you click on the button. Copy the

MyJFrame3 class. Examine the code. Right now, the actionPerformed() method does nothing

with the button event. Compile and run this program and notice that nothing happens when you

click on the button. Add code to the actionPerformed() method so that each time you click on the

button, the number displayed increments by one. For example, when you click on it the first time

a "1" should appear, the second time you click on it a "2" should appear and so on. Hint: Add a

new private integer instance variable buttonCount that keeps track of how many times the button

has been clicked on. What should you initialize this variable to in the constructor? Every time the

button is clicked, increment this variable and display the new value on the button. In your Java

documentation, look up the setText() method that the JButton class inherits from its parent class

AbstractButton(). Also, look at how the JButton was constructed for some ideas about how to

convert the value of buttonCount to a String. You can pass this string as a parameter to the

button's setText() method to set the label on the button to the new value of your counter variable.

(iv) Copy the MyJFrame4 class and examine it. In this version, MyJFrames are constructed with

two buttons both labeled "0". Notice that a MyJFrame4 object listens for action events from both

buttons. This means that when an action event occurs, you must get the event source and check to

see which button (button1 or button2) was clicked. Once you know which button, the event

handler should increment that button's count (button1Count or button2Count) and redisplay the

new count on the appropriate button.

(v) Copy and examine the MyJFrame5. Compile and run this program. The button works just like

it did in version 3. Notice that whenever you type something into the text field and hit ENTER,

the button increments its number. This is because the actionListener() method doesn't distinguish

between an event from the button or the text box. Modify your actionListener() code so that

when you enter text in the text field and hit ENTER, the handler extracts the String from the text

and turns it into an int (use Integer.parseInt()). This value should then be used to update the

buttonCount value and also the label on the button. For example, suppose the button currently

displays "2" and you enter "77" into the text field and hit ENTER. The button should display

"77". Furthermore, if you follow this with clicking on the button, it should display "78" and so on.

Add a label to this version of MyJFrame, positioned above the text field, that says "Enter a

number to update the button". This is to let the user know what should be entered into the text

field. The label should be centered (as best you can) just above the text field. You may need to

experiment with a few position values to get the label to look right.

(vi) Add a menu to your version of MyJFrame5 that has two menu items, one labeled Clear and

the other labeled Exit.. You will need to add instance variables for the menu and the two menu

items. Use the MyJFrame object as the action listener for each of the two menu items, but do not

change your actionListener() code yet. Compile your program and check to see that the menu

appears and that you can "select" either of the two menu items labeled Clear and Exit, but that

nothing happens when you do. Finally, change the actionListener() code so that if the event

source is clearItem, the value of buttonCount will be reset to zero (and the button text will change

accordingly) and if the event source is exitItem, the program will terminate immediately using

System.exit(0).

7. Write an application which will prompt you (in a label) to enter your age into a text field, and

when you press return in the text field or click the ok button, will display in another label your

age in one year’s time. If you click the ‘quit’ button the system should display a closing dialog,

then shut down. Note that the ‘getText()’ method returns a String: you may need to do some

parsing.