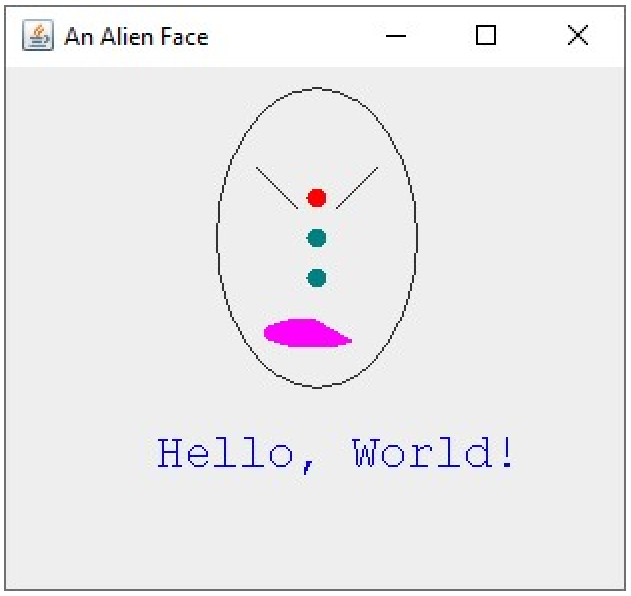
COMP 3609 – Game Programming

Lab #1

1. Download the program EmptyFrame.java from myeLearning.
   - Compile and run the program.
   - Modify the code to reflect the following changes and observe effect on the JFrame.
     Set the size of the JFrame to 500 x 500
   - Change the setVisble attribute to false
   - Update the title on the JFrame to *<Your name>'s JFrame.*
   - Change the setDefaultCloseOperation to one of the following: DO_NOTHING_ON_CLOSE, HIDE_ON_CLOSE or DISPOSE_ON_CLOSE.

2. Create the following:
   - A class, FaceComponent. java, which inherits from JComponent and overrides its paint() method by drawing various shapes using the Graphics object that is passed as a parameter. The shapes look like an alien face. They are drawn using Java's 2D API.
   - An application class, FaceViewer.java which creates an instance of the FaceComponent and adds it to the JFrame, resulting in the face being drawn on the window.

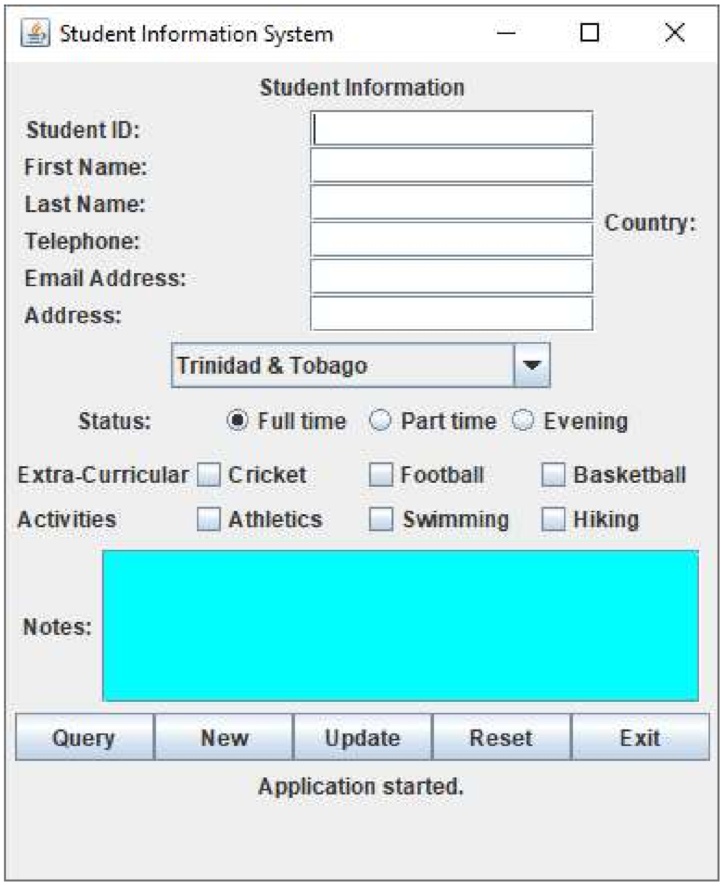


3. Create a class, RectangleComponent.java, is written that inherits from JComponent. It draws a rectangle, translates the rectangle and draws the translated rectangle. It then translates and draws the translated rectangle one more time. The JFrame creates an instance of the RectangleComponent and adds it to the window, resulting in three rectangles being drawn.

4. Write an application as follows. The application consists of two classes:
   a. StudentsWindow.java which hourse all the graphical user interface (GUI) elements and listeners.
      i. The program creates a JFrame for a student application. It creates and places various GUI elements on the JFrame (as shown below). It determines which key is pressed by implementing the KeyListener interface. In addition, the location where the mouse is clicked is determined by implementing the MouseListener interface.

   StudentApplication.java as indicated below:

```
import javax.swing.*;
public class StudentApplication{
        public static void main (String[] args) {
                JFrame studentWindow = new StudentWindow();
        }
}
```



A JFrame can hold several components in it. Some of these components include:

Component	Description
JLabel	Text used to identify something in the frame
JTextField	A field that can be used to type text into
JRadioButton	A button/Selector that can be used to select from multiple options
ButtonGroup	Used to group a set of buttons
JCheckBox	A checkbox used to select an option
JButton	A button that can be pressed to complete some task
JTextArea	Like a text field, just much larger
JScrollPane	Allows for a component to be scrolled
JComboBox	A drop down list used for selecting an option from said list

What is an Event?

When an object changes its state this is known as an event. Events are generated as a result of user interaction with the graphical user interface components. For example, clicking on a button, moving the mouse, entering a character through keyboard, selecting an item from the list, and scrolling the page are the activities that causes an event to occur.

Types of Event

The events can be broadly classified into two categories −

- Foreground Events – These type of events requires *direct* interaction of the user. They are generated as a consequence of a person interacting with the graphical components in the GUI. For example, clicking on a button, moving the mouse, entering a character through keyboard, selecting an item from list, scrolling the page, etc.

- Background Events − These type of events requires the interaction of the end user. Operating system interrupts, hardware or software failure, timer expiration, and operation completion are some examples of background events.

What is Event Handling?

Event Handling is the mechanism that controls the event and decides what should happen if an event occurs. This mechanism has a code which is known as an *event handler*, that is executed when an event occurs.

Java uses the Delegation Event Model to handle the events. This model defines the standard mechanism to generate and handle the events. The Delegation Event Model has the following key participants.

- Source − The source is an object on which the event occurs. Source is responsible for providing information of the occurred event to its handler. Java provide us with classes for the source object.

- Listener − It is also known as event handler. The listener is responsible for generating a response to an event. From the point of view of Java implementation, the listener is also an object. The listener waits till it receives an event. Once the event is received, the listener processes the event and then returns.

The benefit of this approach is that the user interface logic is completely separated from the logic that generates the event. The user interface element is able to delegate the processing of an event to a separate piece of code.

In this model, the listener needs to be registered with the source object so that the listener can receive the event notification. This is an efficient way of handling the event because the event notifications are sent only to those listeners who want to receive them.

Steps Involved in Event Handling

Step 1: The user clicks the button and the event is generated.

Step 2: The object of concerned event class is created automatically and information about the source and the event get populated within the same object.

Step 3: Event object is forwarded to the method of the registered listener class.

Step 4: The method is gets executed and returns.

In order to design a listener class, you have to develop some listener interfaces. These Listener interfaces forecast some public abstract callback methods, which must be implemented by the listener class.

SWING Event Classes

Some popular event classes are:

Events	Description
ActionEvent	The ActionEvent is generated when the button is clicked or the item of a list is double-clicked.
InputEvent	The InputEvent class is the root event class for all component-level input events.
KeyEvent	On entering the character the Key event is generated.
MouseEvent	This event indicates a mouse action occurred in a component.
MouseMotionEvent	The object of this class represents the change in the state of a window.
WindowEvent	The object of this class represents the change in the state of a window.

SWING - Event Listeners

Event listeners represent the interfaces responsible to handle events. Java provides various Event listener classes, however, only those which are more frequently used will be discussed. Every method of an event listener method has a single argument as an object which is the subclass of EventObject class. For example, mouse event listener methods will accept instance of MouseEvent, where MouseEvent derives from EventObject.

Action Listener:	These are listeners that are used to respond to an action (button being pressed etc). There is one Action Listener: actionPerformed: Responds to any action that occurs within the frame
Key Listeners:	These are listeners that respond to using the keyboard. There are three types of Key Listeners. • keyPressed: Responds to when a key is pushed down • keyReleased: Responds to when a key is no longer being pressed • keyTyped: Responds to when a key is pressed and released
Mouse Listeners:	These are listeners that respond to using the mouse. There are 5 types of Mouse Listeners. • mouseClicked: Responds to when a mouse button is clicked and released • mouseEntered: Responds to when the mouse enters a specified component.mouseExited: Responds to when the mouse exists a specified component • mousePressed: Responds to when the mouse button is pressed down • mouseReleased: Responds to when the mouse button is released from being pressed

References

1. https://www.tutorialspoint.com/swing/swing_event_handling.htm

2. The application for Question 4, contains copyrighted material from the book, Fundamentals of Object-Oriented Programming in Java. It is used with permission.