EECS1720
Worksheet 6 – Java AWT/Swing Event Handling

SOLUTIONS

1) What is an Event?

   An Event is an object that is created (instantiated) when something changes within a graphical user interface.

   It notifies the program that an action has been taken by a user, allowing the program to respond to, and process that action.

2) How do Events differ from Exceptions?

   An Exception is an object that is created during the execution of a program when something unexpected occurs that disrupts the normal flow of a program's instructions.

   An Event defines a specific user interaction with the GUI that can send flow of program execution directly to an appropriate handler to process the event before continuing.

   An Event is triggered by a user's interaction, while an Exception is triggered by a potential problem that has arisen in the program. The routing of an Event vs. Exception is also different. Exceptions are routed back through the calling stack, while Events are routed directly to listener methods to which have been registered against GUI objects or JComponents that act as the source of those events.

3) How are Events dispatched and handled?

   The mechanism for capturing and handling Events vs Exceptions is different.

   Exceptions use try/catch blocks to capture and handle events (and Exceptions will be delegated up the call stack if unhandled, ultimately to the JVM if not handled in the program).

   Events are delegated to Event Listeners that have been explicitly configured to respond to specific events occurring on registered JComponents

4) What is an Inner Class (why are they useful)?

   A class that is defined within another (main) class [i.e. in the same .java file]

Useful for defining event handlers that are specific for actions to be taken from events triggered in a particular application (main class).

Both classes have direct access to any class variables defined in the enclosing (application) class, which can make it simpler to modify other GUI objects defined in the application

5) What types of Events can we typically detect through interactions with a (GUI) interface?

ActionEvent, MouseEvent, MouseMotionEvent, KeyEvent (main ones)

6) What does the keyword **`instanceOf`** signify in a java program?

For testing whether one object is an instance of a particular type: e.g. testing whether a source of an event is an instance of a particular class.

E.g. JButton, or some other kind of class relating to a particular JComponent that might have been configured to be the source of a particular kind of event: e.g. a JPanel or JLabel as a source of a KeyEvent

7) In the context of Swing, How do you create a

// see lecture slides and sample code for examples

i. Button
ii. CheckBox
iii. Radio Button
iv. Image

And link it to a

v. ActionEvent
use addActionListener(..) to register an instance of an object that implements ActionListener interface, and has overridden the actionPerformed method to appropriately handle a desired event

vi. MouseEvent
Similar to vi) use addMouseListener(…). Must register with an instance of a class that implements MouseListener interface

vii. MouseMotionEvent
Similar to vii) use addMouseMotionListener(…) , must register with an instance of a class that implements MouseMotionListener interface

viii. KeyEvent
Use addKeyListener(…) to a container (preferably the JFrame), and register an instance of a class that implements KeyListener interface. Ensure the container has the focus (by invoking requestFocus() on that container).

8) What is an Adaptor? Give an example of how an Adaptor can be used.. what is the advantage of using an Adaptor over a Listener?

An adaptor is a class that implements a known interface, however the implementations of the interface's methods are empty. By extending an adaptor class, one only needs to override the methods that they intend to use (as opposed to declaring and overriding all methods required by the listener interface).

9) In the context of using a set of JRadioButtons, explain how they would first be constructed, then how you would setup a Listener to respond to when they are changed. Assume that when a new JRadioButton is selected, a different message is outputted to the System console (i.e. standard output).

Create several JRadioButton objects (individually).
Register each JRadioButton object to an ActionListener instance (can be the same listener).
Make a new ButtonGroup object, and add each of the JRadioButtons created to it. This ensures that selection of one JRadioButton will de-select the others in the group.

Create an inner class that implements ActionListener (this will be the listener registered above). Override the actionPerformed(..) method. In this method, go through each of the JRadioButtons in the ButtonGroup and process them (or check which is selected and for the one that is selected, process it… in this question, the selected one would be used to print a message to the console)

10) What does it mean to get the "focus" in a Java Swing application? How does one assign focus to a particular JFrame or a specific JComponent?

Focus means the control or window in the OS that has currently been selected and will be the target for events to be routed through

See L17, slide 21

11) Outline at least two different approaches to setting up a Listener for a KeyEvent

i. Create a class that implements KeyListener interface, then override the 3 methods keyPressed(..), keyTyped(..), keyReleased(..)
ii. Create a class that extends KeyAdapter class, then override only the methods needed

12) In the context of KeyEvents, what are "modifiers" ? How can they be accessed?

See L17, slide 30

Modifiers are the list of other keys that are currently being pressed while the current key is being interacted with (pressed/released)

13) How do MouseMotionEvents differ from MouseEvents? How do we process these?

MouseMotionEvents relate to events that are continuously triggered as the mouse is in motion, MouseEvents are static, and are only triggered once when a mouse enters/leaves a region/gui object's bounds within the user space, or a mouse button is interacted with (pressed/released).

We need a class that implements MouseMotionListener for the former, or MouseListener for the latter, with the associated methods overridden. This can be a single class that implements both interfaces.

Event Listeners

User Action	Event Triggered	Event Listener interface
Click a `Button`, `JButton`	`ActionEvent`	`ActionListener`
Open, iconify, close `Frame`, `JFrame`	`WindowEvent`	`WindowListener`
Click a `Component`, `JComponent`	**`MouseEvent`**	**`MouseListener`**
Change texts in a `TextField`, `JTextField`	`TextEvent`	`TextListener`
Type a key	**`KeyEvent`**	**`KeyListener`**
Click/Select an item in a `Choice`, `JCheckbox`, `JRadioButton`, `JComboBox`	`ItemEvent`, `ActionEvent`	`ItemListener`, `ActionListener`