Principles of Computer Science II

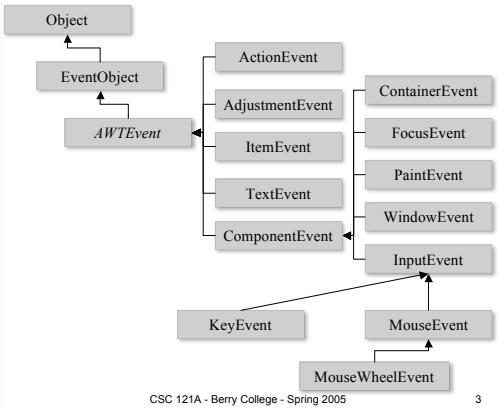
Prof. Nadeem Abdul Hamid

CSC 121A – Spring 2005

Lecture Slides 5 - Event Handling and Exception Control Structures

GUI Events

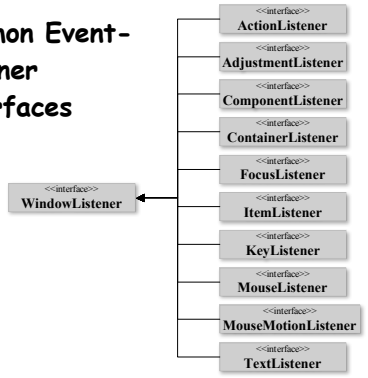
- Many different types of events occur when user interacts with a GUI
- Information about any event is stored in object of a subclass of AWTEvent
- Three parts to event-handling mechanism
  - Event source
    - Component with which user interacts
  - Event object
    - Info about event that occurred: reference to the event source, event-specific information used by the listener to process it
  - Event listener
    - Object that is notified when an event occurs
    - Particular method receives an event object when the listener is notified of the event



Event Handling

- For each event-object type, there is usually a corresponding event-listener interface
- Each interface specifies one or more event-handling methods
- When event occurs, GUI component notifies registered listeners by calling appropriate event-handling method

Common Event-Listener Interfaces



Adapter Classes

- Many listener interfaces specify multiple methods (e.g. `MouseListener`)
- Sometimes application only needs a single handler method and would just define empty bodies for other methods specified in the interface
- Instead of defining empty bodies oneself, a handler class can just extend one of the adapter classes
  - Adapter class implements an interface and provides a default implementation (i.e. empty method body) for methods of the interface

Adapter Classes and Interfaces

`ComponentAdapter`	`ComponentListener`
`ContainerAdapter`	`ContainerListener`
`FocusAdapter`	`FocusListener`
`KeyAdapter`	`KeyListener`
`MouseAdapter`	`MouseListener`
`MouseMotionAdapter`	`MouseMotionListener`
`WindowAdapter`	`WindowListener`

- Example: lec05/MouseDetails.java
  lec05/MouseDetailFrame.java

Subclassing JPanel for Mouse Drawing

- Uses a JPanel as dedicated draw area
- Extends an adapter class for event handling

- lec05/Painter.java
- lec05/PaintPanel.java

Key Event Handling

- Key events generated when keys on keyboard are pressed and released
- KeyListener interface declares methods
  - keyPressed
  - keyReleased
  - keyTyped
  - Each receives a KeyEvent object as argument
- Example:
  - lec05/KeyDemo.java
  - lec05/KeyDemoFrame.java

Component Layout

- Provided to arrange GUI components in a container for presentation
- Three ways to arrange components
  - Absolute positioning: set layout to `null`, specify absolute position of each component w.r.t. upper-left corner of container
  - Layout managers: simpler and faster but not as much control over positioning
  - Visual programming using IDE design tool

Layout Managers

- FlowLayout
  - Default for JPanel
  - Places components left-to-right in order they are added
- BorderLayout
  - Default for JFrame and other windows
  - Arranges components in five areas: NORTH, SOUTH, EAST, WEST, CENTER
  - Limits container to having 5 components
- GridLayout
  - Arranges components in rows and columns

Complex Layouts Using JPanels

- A JPanel is a container and thus may have components, including other JPanels, added to it
- Create complex layouts using nested JPanels and layout managers

- Example
  - lec05/PanelDemo.java
  - lec05/PanelFrame.java

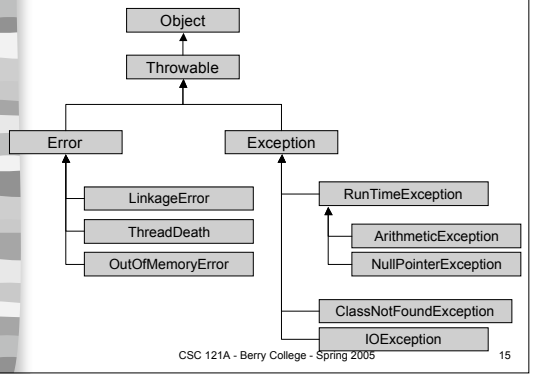
Exception Handling

- Exception: indication of a problem during program's execution
- Exception handling
  - Allows programmers to remove error-handling code from "main line" of program's execution
  - Enables writing *robust* and *fault-tolerant* programs
- Exceptions surface from
  - Explicitly mentioned code in a try block
  - Calls to other methods (may be deeply nested)
  - Java Virtual Machine (JVM) as it executes

Try-Catch Blocks

- A `try` block encloses code that might throw an exception and code that should not execute if the exception occurs
- A `catch` block takes an exception parameter followed by code that catches (i.e. receives) and handles the exception
  - Executes only when an exception arises somewhere in the `try` block
  - At least one `catch` or `finally` block must immediately follow the `try` block
- After exception is handled, control resumes after the last `catch` block
- Uncaught exceptions (i.e. with no matching `catch` block) cause the program to terminate
- Example: lec05/ExceptionTest.java

Java Exception Hierarchy



Types of Exceptions

- Unchecked – derived from RunTimeException
- Checked – must be caught by a method or else must be listed in a throws clause of the method (e.g. IOException)
- Errors are liked unchecked exceptions
  - Usually, should not be caught and do not require listing in a throws clause

Exception Example

- lec05/ExceptionTest2.java
  - Defining and throwing user-defined exceptions
  - Optional `finally` block: executed no matter what happens
  - Multiple `catch` blocks matching exception type