Object Oriented Paradigms

College Requirements -Compulsive Courses

CSCR2105

Exception handling

Syntax Errors, Runtime Errors, and Logic Errors

- There are three categories of errors:
  - ***Syntax errors*** arise because the rules of the language have not been followed.
    - They are detected by the compiler.
  - ***Runtime errors*** occur while the program is running if the environment detects an operation that is impossible to carry out.
  - ***Logic errors*** occur when a program doesn't perform the way it was intended to.

Exception

- **Definition**: an occurrence of an undesirable situation that can be detected during program **execution** (e.g. as opposed to at compile-time)
- Examples
  - Division by zero
  - Trying to open an input file that does not exist
  - An array index that goes out of bounds
- When an exception is generated, some code, somewhere must "handle" it. If the exception is never handled, the program will crash.

Exception -Terminology

- When an error is detected, an exception is **thrown**
- Any exception which is thrown, must be caught by and **exception handler**
  - If the programmer hasn't provided one, the exception will be caught by a **catch-all exception handler** provided by the system.
  - The default exception handler may terminate the application.
- Exceptions can be **rethrown** if the exception cannot be handled by the block which caught the exception
- Java has 5 keywords for exception handling:
  - **Try , catch, finally, throw, throws**

Exception handling

- If we wish to handle an exception, the error (exception) generating code is enclosed in something called a **try** block.
- We can handle the exception by following the try block with one or more **catch** blocks.
- Within a **try/catch** block, you have the option of "**handling**" the exception – but you are not required to.
  - You can simply ignore the exception object and allow some other part of the program to "**catch**" it.
- Sometimes we *want* to delegate the handling of an exception to an outer-level. In that case, we can include a '**`throws`**' statement.

`try/catch/finally` Block

- Statements that might generate an exception are placed in a **`try`** block
  - The **`try`** block might also contain statements that should *not* be executed if an exception occurs
  - The **`try`** block is followed by zero or more **`catch`** blocks
    - A catch block is where the "handling" takes place, and for this reason, is also known as an exception **handler**
- A **`catch`** block specifies the type of exception it can catch and contains an exception handler
  - Every **catch block** must specify one (and only one) exception parameter

`try/catch/finally` Block (contd)

- The last **`catch`** block may or may not be followed by a **`finally`** block
  - Another way of stating: Every **try** block must be followed by at least one **catch** block and/or a **finally** block

- Any code contained in a **`finally`** block always executes, regardless of whether an exception actually occurs!
  - One exception to this rule: When the program exits early from a `try` block by calling the method `System.exit();`
- If a **`try`** block has no **`catch`** block, then it *must* have the **`finally`** block

try/catch/finally Block (continued)

```
try
{
    //statements
}
catch (ExceptionClassName1 objRef1)
{
    //exception handler code
}
catch (ExceptionClassName2 objRef2)
{
    //exception handler code
}
...
catch (ExceptionClassNameN objRefN)
{
    //exception handler code
}
finally
{
    //statements
}
```

You can include a catch block for every possible exception that might have been generated inside the try block.

Optional.

NOW:

Waiting for your questions and comments

Lecture 6

Files, Streams and Object Serialization

Some Java Exception Classes

TABLE 12-3 Some of Java's Exception Classes

Exception Class	Description
`ArithmeticException`	Arithmetic errors such as division by zero
`ArrayIndexOutOfBoundsException`	Array index is either less than 0 or greater than or equal to the length of the array.
`FileNotFoundException`	Reference to a file that cannot be found
`IllegalArgumentException`	Calling a method with illegal arguments
`IndexOutOfBoundsException`	An array or a string index is out of bounds.
`NullPointerException`	Reference to an object that has not been instantiated

Some Java Exception Classes (contd)

TABLE 12-3 Some of Java's Exception Classes (continued)

Exception Class	Description
`NumberFormatException`	Use of an illegal number format
`StringIndexOutOfBoundsException`	A string index is either less than 0 or greater than or equal to the length of the string.
`InputMismatchException`	Input (token) retrieved does not match the pattern for the expected type, or the token is out of range for the expected type.

Some exceptions from the Scanner class

TABLE 12-4 Exceptions Thrown by the Method `nextInt`

Exception Thrown	Description
`InputMismatchException`	If the next input (token) is not an integer or is out of range
`NoSuchElementException`	If the input is exhausted
`IllegalStateException`	If this scanner is closed

TABLE 12-5 Exceptions Thrown by the Method `nextDouble`

Exception Thrown	Description
`InputMismatchException`	If the next input (token) is not a floating-point number or is out of range
`NoSuchElementException`	If the input is exhausted
`IllegalStateException`	If this scanner is closed

API for nextInt() method

Java Exception Classes (continued)

TABLE 12-9 Exceptions Thrown by the Method `hasNext`

Exception Thrown	Description
`IllegalStateException`	If this scanner is closed

TABLE 12-10 Exceptions Thrown by the Methods of the `class` `Integer`

Method	Exception Thrown	Description
`parseInt(String str)`	`NumberFormatException`	The string `str` does not contain an `int` value.
`valueOf(String str)`	`NumberFormatException`	The string `str` does not contain an `int` value.

Java Exception Classes (continued)

TABLE 12-11 Exceptions Thrown by the Methods of the `class Double`

Method	Exception Thrown	Description
`parseDouble(String str)`	`NumberFormatException`	The string `str` does not contain a `double` value.
`valueOf(String str)`	`NumberFormatException`	The string `str` does not contain a `double` value.

Java Exception Classes (continued)

TABLE 12-12 Exceptions Thrown by the Methods of the `class String`

Method	Exception Thrown	Description
`String(String str)`	`NullPointerException`	`str` is `null`.
`charAt(int a)`	`StringIndexOutOfBoundsException`	The value of `a` is not a valid index.
`indexOf(String str)`	`NullPointerException`	`str` is `null`.
`lastIndexOf(String str)`	`NullPointerException`	`str` is `null`.
`substring(int a)`	`StringIndexOutOfBoundsException`	The value of `a` is not a valid index.
`substring(int a, int b)`	`StringIndexOutOfBoundsException`	The value of `a` and/or `b` is not a valid index.

Event Handling (continued)

TABLE 12-13 Events Generated by a GUI Component, the Listener Interface, and the Name of the Method of the Interface to Handle the Event

GUI Component	Event Generated	Listener Interface	Listener Method
JButtton	ActionEvent	ActionListener	actionPerformed
JCheckBox	ItemEvent	ItemListener	itemStateChanged
JCheckboxMenuItem	ItemEvent	ItemListener	itemStateChanged
JChoice	ItemEvent	ItemListener	itemStateChanged
JComponent	ComponentEvent	ComponentListener	componentHidden
JComponent	ComponentEvent	ComponentListener	componentMoved
JComponent	ComponentEvent	ComponentListener	componentResized
JComponent	ComponentEvent	ComponentListener	componentShown
JComponent	FocusEvent	FocusListener	focusGained
JComponent	FocusEvent	FocusListener	focusLost

Event Handling (continued)

Container	ContainerEvent	ContainerListener	componentAdded
Container	ContainerEvent	ContainerListener	componentRemoved
JList	ActionEvent	ActionListener	actionPerformed
JList	ItemEvent	ItemListener	itemStateChanged
JMenuItem	ActionEvent	ActionListener	actionPerformed
JScrollbar	AdjustmentEvent	AdjustmentListener	adjustmentValueChanged
JTextComponent	TextEvent	TextListener	textValueChanged
JTextField	ActionEvent	ActionListener	actionPerformed
Window	WindowEvent	WindowListener	windowActivated
Window	WindowEvent	WindowListener	windowClosed
Window	WindowEvent	WindowListener	windowClosing
Window	WindowEvent	WindowListener	windowDeactivated
Window	WindowEvent	WindowListener	windowDeiconified
Window	WindowEvent	WindowListener	windowIconified
Window	WindowEvent	WindowListener	windowOpened

Event Handling (continued)

TABLE 12-14 Events Generated by `key` and `mouse` Components

	Event Generated	Listener Interface	Listener Method
key	KeyEvent	KeyListener	keyPressed
key	KeyEvent	KeyListener	keyReleased
key	KeyEvent	KeyListener	keyTyped
mouse	MouseEvent	MouseListener	mouseClicked
mouse	MouseEvent	MouseListener	mouseEntered
mouse	MouseEvent	MouseListener	mouseExited
mouse	MouseEvent	MouseListener	mousePressed
mouse	MouseEvent	MouseListener	mouseReleased
mouse	MouseEvent	MouseMotionListener	mouseDragged
mouse	MouseEvent	MouseMotionListener	mouseMoved