Exception Handling

Bob Singh

Exception Handling

- An exception is an event, which occurs during the execution of a program that disrupts the normal flow of instructions.
- Transfers control to special exception-handling code, designed in a program, if an exceptional condition occurs.

Why is Exception Handling needed?

- Separating Error Handling Code from Regular Code
- Propagating Errors Up the Call Stack
- Grouping Error Types and Error Differentiation
  - Exception handlers designed to catch exception based on the group of exceptions, or one of the specialized exceptions.

Runtime Exceptions

- **`RuntimeException`** are those exceptions that occur within the Java runtime system. Includes:
  - Arithmetic exceptions (such as division by zero)
  - Pointer exceptions (such as trying to access an object through a null reference)
  - Indexing exceptions (such as attempting to access an array element through an index that is too large or too small).

try Block

- Exception handling is done by using try - catch – finally blocks.

- Syntax for `try` block:

```
try {
   //java statements
}
```

- One or more statements can throw one or more exceptions.
- Each exception needs to be handled.

try Block

- Each statement that throws exceptions can have its own `try` block and provide separate exception handlers for each `try` block.
- Or, multiple exception throwing statements can be put in a single `try` block and provide multiple exception handlers.
- Exception handlers must be placed immediately following their associated `try` block.
- A try block must be accompanied by at least one catch block or one finally block.

catch Block

- Syntax for **`catch`** block:

```
catch (Exception variableName) {
    //java statements
}
```

- The header line in the **`catch`** block requires a single argument .
- The syntax is the same as an argument declaration for a method.

catch Block

- The argument type declares the type of exception object that a particular exception handler can handle
- Also as in a method declaration, name of the exception object is given by the Exception handler to refer to it.

finally Block

- Java's **`finally`** block provides a mechanism that allows method to clean up after itself, regardless of what happens within the **`try`** block.
- The **`finally`** block can be used to close files or release other system resources.

Example (Program1.java)

```
class exceptionJava {
   public static void main (String[] args) {
   int counter = 0;
   try {
        for( int i=0; i <4; i++) {
        counter = 10 / i;
        System.out.println("Counter: " + counter);
        }
   }
    catch (ArithmeticException ae) {
        System.out.println("Divide by zero exception");
   }
   finally {
        System.out.println ("In the finally clause");
   }}
```

Self Healing Software (Example2)

- CustomException extends Exception
- Methods to fix Exceptions caught using try/catch and continuing the program from point of Exception