15. Exception Handling

[ECE20016/ITP20003] Java Programming

Agenda

- Basic Exception Handling
- Defining Your Own Exception Classes
- More About Exception Classes

Exceptions in Java

- An exception is ***an object*** that contains an error data and related actions.
  - Signals the occurrence of ***unusual event*** during program execution that **can be handled**. e.g.,
    - NullPointerException, IndexOutOfBoundException,...
- Throwing an exception when it happens
  - We can ***define our own erroneous situation*** by throwing our own exception.
  - Throwing an exception means creating the exception object.
- Handling the exception
  - Code that detects and deals with the exception.

Exceptions in Java

- Consider a program to assure us of a sufficient supply of milk.

```
Enter number of donuts:
2
Enter number of glasses of milk:
0
No milk!
Go buy some milk.
End of program.
```

A Possible Solution

```
Scanner keyboard = new Scanner (System.in);
System.out.println ("Enter number of donuts:");
int donutCount = keyboard.nextInt ();
System.out.println ("Enter number of glasses of milk:");
int milkCount = keyboard.nextInt ();
//Dealing with an unusual event without Java's exception handling features:
if (milkCount < 1)  {
    System.out.println ("No milk!");
    System.out.println ("Go buy some milk.");
} else {
    double donutsPerGlass = donutCount / (double) milkCount;
    System.out.println (donutCount + " donuts.");
    System.out.println (milkCount + " glasses of milk.");
    System.out.println ("You have " + donutsPerGlass +
            " donuts for each glass of milk.");
}
System.out.println ("End of program.");
```

https://github.com/lifove/ExceptionExamples/blob/master/src/main/java/edu/handong/csee/java/exception/examples/SupplierWithoutExceptionHandling.java

Exceptions in Java

- Now we revise the program to use exception-handling

```
try {
    System.out.println ("Enter number of donuts:");
    int donutCount = keyboard.nextInt ();
    System.out.println ("Enter number of glasses of milk:");
    int milkCount = keyboard.nextInt ();
    if (milkCount < 1)
        throw new Exception ("Exception: No milk!");
    double donutsPerGlass = donutCount / (double) milkCount;
    System.out.println (donutCount + " donuts.");
    System.out.println (milkCount + " glasses of milk.");
    System.out.println ("You have " + donutsPerGlass +
            " donuts for each glass of milk.");
} catch (Exception e) {
    System.out.println (e.getMessage ());
    System.out.println ("Go buy some milk.");
}
System.out.println ("End of program.");
```

https://github.com/lifove/ExceptionExamples/blob/master/src/main/java/edu/handong/csee/java/exception/examples/Supplier.java

Exceptions in Java

- Output

```
Enter number of donuts:
3
Enter number of glasses of milk:
2
3 donuts.
2 glasses of milk.
You have 1.5 donuts for each glass of milk.
End of program.
```

```
Enter number of donuts:
2
Enter number of glasses of milk:
0
Exception: No milk!
Go buy some milk.
End of program.
```

Exceptions in Java

- Predefined Exceptions in Java
  - Public method calls or accessing data can throw predefined exceptions.
    - e.g., IndexOutOfBounds esception when using arrays.
- Customized Exceptions by each developer.

Predefined Exception Classes

Ex)

```
SampleClass object = new SampleClass();
try
{
    <Possibly some code>
    object.doStuff(); //may throw IOException
    <Possibly some more code>
}
catch(IOException e)
{
    <Code to deal with the exception, probably including the following:>
    System.out.println(e.getMessage());
}
```

Exceptions in Java

- Note try block
  - Contains code where something could possibly go wrong
  - If it does go wrong, we throw an customized exception or method calls or data access we are using in the try block will throw a predefined exception.

- Note catch block
  - When exception thrown, catch block begins execution
  - Similar to method with parameter
  - Parameter is the thrown Exception object

Predefined Exception Classes

- Java has predefined exception classes within Java Class Library
  - Can place method invocation in try block
  - Follow with catch block for this type of exception

- Example classes
  - NullPointerException
  - ArrayOutOfBoundException
  - BadStringOperationException
  - ClassNotFoundException
  - IOException
  - NoSuchMethodException

Agenda

- Basic Exception Handling
- **Defining Your Own Exception Classes**
- More About Exception Classes
- Graphics Supplement

Defining Your Own Exception Classes

- Must be derived class of some predefined exception class
  - Text uses classes derived from class Exception

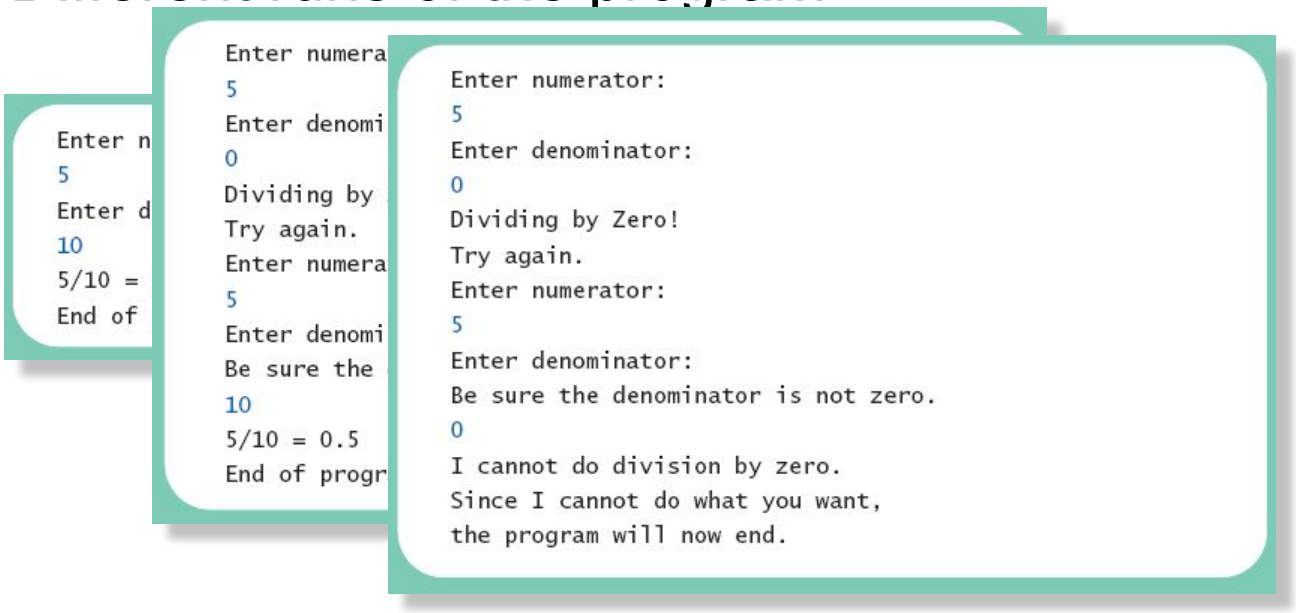
Ex)

```
public class DivideByZeroException extends Exception
{
    public DivideByZeroException ()
    {
        super ("Dividing by Zero!");
    }

    public DivideByZeroException (String message)
    {
        super (message);
    }
}
```

Defining Your Own Exception Classes

- class DivideByZeroDemo
  - https://github.com/lifove/ExceptionExamples/tree/master/src/main/java/edu/handong/csee/java/exception/examples/dividebyzero
- Different runs of the program



Defining Your Own Exception Classes

- Note method getMessage defined in exception classes
  - Returns string passed as argument to constructor
  - If no actual parameter used, default message returned
- The type of an object is the name of the exception class

Defining Your Own Exception Classes

- Guidelines
  - Use the Exception as the base class
  - Define at least two constructors
    - Default, no parameter
    - With String parameter
  - Start constructor definition with call to constructor of base class, using super
  - Do not override inherited getMessage

Agenda

- Basic Exception Handling
- Defining Your Own Exception Classes
- **More About Exception Classes**
- Graphics Supplement

Declaring Exceptions

- Sometimes, it makes sense to delay handling of an exception.
  - A method can pass an exception to its caller.
- Method that does not catch an exception
  - Must notify programmers with throws clause
  - Programmer then given responsibility to handle exception
- Note distinction
  - Keyword throw used to throw exception
  - Keyword throws used in method heading to declare an exception

Declaring Exceptions

- Note syntax for throws clause

```
public Type Method_Name(Parameter_List) throws List_Of_Exceptions
Body_Of_Method
```

  - Indicates that the method may throw exceptions

Ex)

```
public void sampleMethod() throws DivideByZeroException
{
    …
}
```

Declaring Exceptions

- If a method throws exception and exception not caught inside the method
  - Method ends immediately after exception thrown
- A throws clause in overriding method
  - Can declare fewer exceptions than declared
  - But not more
- class DoDivision
  - https://github.com/lifove/ExceptionExamples/blob/master/src/main/java/edu/handong/csee/java/exception/examples/dividebyzero/DoDivision.java

Kinds of Exceptions

- In most cases, exception is caught …
  - In a catch block … or
  - Be declared in throws clause
- But, Java has exceptions you do not need to account for

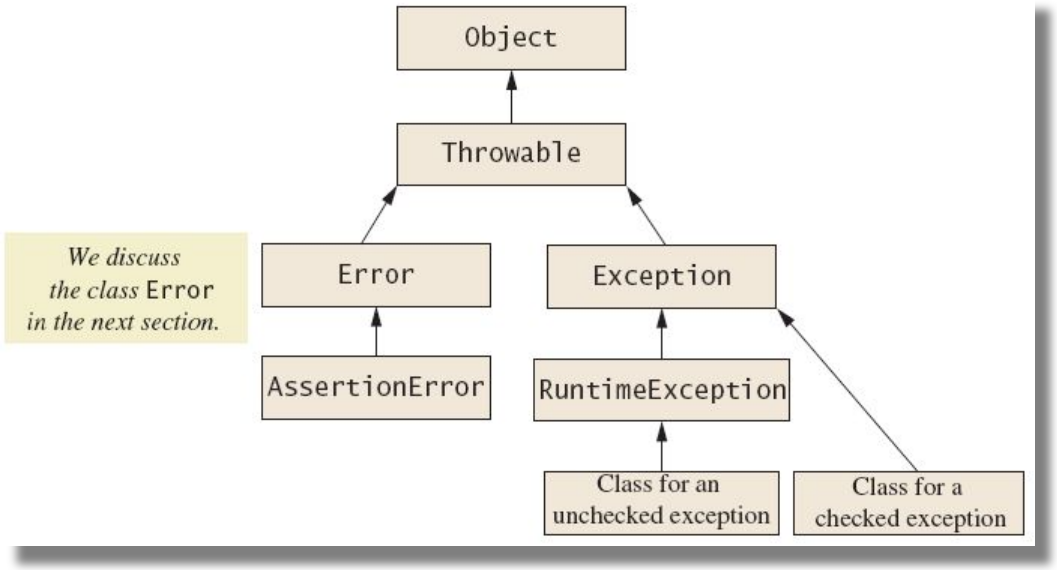
Kinds of Exceptions

- Checked exceptions
  - Must be caught in catch block or declared in throws clause
- Unchecked exceptions (run-time exception)
  - Need not be caught in catch block or declared in throws
  - Exceptions that indicates coding problems exist, should be fixed

  Ex) Attempt to use array index out of bounds, Division by zero
- Uncaught runtime exception terminates program execution

Kinds of Exceptions

- Hierarchy of the predefined exception classes



Errors

- An error is an object of class Error
  - Similar to an unchecked exception
  - Need not catch or declare in throws clause
  - Object of class Error generated when abnormal conditions occur

- Errors are more or less beyond your control

  Ex) OutOfMemoryError
  - Require change of program to resolve

Multiple Throws and Catches

- A try block can throw any number of exceptions of different types
- Each catch block can catch exceptions of only one type
  - Order of catch blocks matter
- class TwoCatchesDemo
  - main() can throw NegativeNumberException
  - exceptionalDivision() can throw DivideByZeroException
  - https://github.com/lifove/ExceptionExamples/blob/master/src/main/java/edu/handong/csee/java/exception/examples/dividebyzero/TwoCatchesDemo.java
- class NegativeNumberException
  - https://github.com/lifove/ExceptionExamples/blob/master/src/main/java/edu/handong/csee/java/exception/examples/dividebyzero/NegativeNumberException.java

Multiple Throws and Catches

- Note multiple sample runs

```
Enter number of widgets produced:
1000
How many were defective?
500
One in every 2.0 widgets is defective.
End of program.
```

```
Enter number of widgets produced:
-10
Cannot have a negative number of widgets
End of program.
```

```
Enter number of widgets produced:
1000
How many were defective?
0
Congratulations! A perfect record!
End of program.
```

Multiple Throws and Catches

- Exceptions can deal with invalid user input
- To handle an exception thrown by a method
  - It does not matter where in the method the throw occurs
- Use of throw statement be should be reserved for cases where it is unavoidable
- Text suggests separate methods for throwing and catching of exceptions
- Nested try-catch blocks rarely useful

The *finally* Block

- Possible to add a finally block after sequence of catch blocks

```
try {
        ...
} catch(Exception e) {
        ...
} finally {
        ...
}
```

- Code in finally block executed
  - Whether or not exception thrown
  - Whether or not required catch exists

Rethrowing an Exception

- Legal to throw an exception within a catch block
- Possible to use contents of String parameter to throw same or different type exception