Exceptions

- Reading: Savitch, Chapter 8

Objectives

- To learn how to handle an exception (exception propagation).
- To learn how to define an exception.

Exception Propagation

- If it is not appropriate to handle the exception where it occurs, it can be handled at a higher level.
- Exceptions can be *propagated* up through the method calling hierarchy until they are caught and handled or until they reach the outermost level (the main() method).

Example

```
//Propagation_Demo.java

public class Propagation_Demo {
    public static void main (String [ ]  args) {
        System.out.println("program  beginning");
        Exception_Scope demo =
                                new Exception_Scope();
       demo.level_1();          // method call
       System.out.println("program  ending");
   }
}
```

```
// Exception_Scope.java  defines three methods
// level_3, level_2, level_1

class Exception_Scope {
   public void level_3 (int adjustment) {
          System.out.println("level_3 beginning");
          int current = 1;
          current = current /adjustment;
                  // don't handle exceptions here
          System.out.println("level_3 ending");
   }
```

```
public void level_2 () {
   System.out.println("level_2 beginning");
   level_3(0);
                    // don't handle exceptions here
   System.out.println("level_2 ending");
}
```

```
 //Exceptions are handled in the method
// when it calls a method
public void level_1 () {
  System.out.println("level_1 beginning");
  try {
        level_2();
  }
  catch (ArithmeticException problem) {
        System.out.println(problem.getMessage());
        problem.printStackTrace();
  }
}  // end of level_1
} // end of class
```

The program execution

% ***java Propagation_Demo***

program beginning

level_1 beginning

level_2 beginning

level_3 beginning

```
/ by zero
java.lang.ArithmeticException: / by zero
        at Exception_Scope.level_3(Propagation_Demo.java:16)
        at Exception_Scope.level_2(Propagation_Demo.java:21)
        at Exception_Scope.level_1(Propagation_Demo.java:28)
at Propagation_Demo.main(Propagation_Demo.java:7)
program ending
```

User Defined Exceptions

- A programmer can define their own exception class by extending the class from an existing API exception class.

- Exceptions are thrown using the throw statement. The syntax of the statement is

```
throw exceptionObject;
```

- When you use a throw-statement in your code you should usually define your own exception class.
- If you use a predefined, more general exception class, then your catch-block will have to be general

Example

```
//Throw_Demo.java
import java.io.IOException;
public class  Throw_Demo {
 public static void main(String[] args) throws Ooops {
    Ooops  problem = new Ooops(“Alert!”);
    throw problem;
  }
}

class Ooops extends IOException {
    Ooops(String message) { super(message); }
}
```

The program execution is

```
% java Throw_Demo
Exception in thread "main" Ooops: Alert!
    at java.lang.Throwable.fillInStackTrace(Native Method)
    at java.lang.Throwable.<init>(Throwable.java:94)
    at java.lang.Exception.<init>(Exception.java:42)
    at java.io.IOException.<init>(IOException.java:47)
    at Ooops.<init>(Throw_Demo.java:12)
    at Throw_Demo.main(Throw_Demo.java:7)
```

- Usually a throw statement is nested inside an if statement that evaluates the condition to see if the exception should be thrown.

Example

```
//ThrowEx.java

import java.io.*;
public class ThrowEx {
    public static void main(String [ ] args) throws IOException
    {
        try {
            System.out.println("Your current age is " + getAge());
        }
```

```
 catch(NumberTooBigException e1)
 {
      System.out.println(e1.getMessage());
 }
 catch(NumberTooSmallException e2)
 {
      System.out.println("Invalid input. Try again.");
 }
} // end of main
```

```
static int getAge() throws IOException,
                           NumberTooBigException,
                           NumberTooSmallException
{
    BufferedReader stdin = new BufferedReader
                (new InputStreamReader (System.in));
    System.out.print ("Enter your age: ");
    int age = Integer.parseInt (stdin.readLine());
```

```
        if (age > 130)
            throw new NumberTooBigException(
                    "Are you sure you are that old?");
        else if (age < 0)
                throw new NumberTooSmallException();
        else
                return age;
    }  // end of getAge
} // end of ThrowEx class
```

```
class NumberTooBigException extends Exception
{

    public NumberTooBigException ()
    {     super();        }
    public NumberTooBigException (String s)
    {     super(s);       }
}
```

```
class NumberTooSmallException extends Exception
{
    public NumberTooSmallException() {super();}
    public NumberTooSmallException (String s) {
    super(s); }
}
```

The program executions

*%**java ThrowEx***

Enter your age: 80

Your current age is 80

*%**java ThrowEx***

*Enter your age: **-4***

Invalid input. Try again.

*%**java ThrowEx***

*Enter your age: **140***

Are you sure you are that old?

Summary

- An exception is an object descended from the `Exception` class
- You can use predefined exception classes or define your own
- Exceptions can be thrown by:
  - certain Java statements
  - methods from class libraries
  - explicit use of the `throw` statement
- An exception can be thrown in either
  - a `try` block, or
  - a method definition without a `try` block, but in this case the call to the method must be placed inside a `try` block

Summary

- An exception is caught in a catch block
- When a method might throw an exception but does not have a `catch` block to catch it, usually the exception class must be listed in the `throws`-clause for the method
- A try block may be followed by more than one catch block
  - more than one catch block may be capable of handling the exception
  - the first catch block that can handle the exception is the only one that executes
  - so put the most specific catch blocks first and the most general last
- Every exception class has a `getMessage` method to retrieve a text message description of the exception caught