Chapter 12

Exception Handling

Error processing

Two questions to ask about the error processing performed by a program are

1) What does the program do when a run-time error is detected?

2) Where does the program handle errors?

Answer (what it does)

1) The program does nothing in which case the error will typically force the program to terminate.

2) The program executes statements that display an error message and then terminates.

3) The program recovers from the error and continues executing.

Answer (where to do it)

1) The program handles errors where they occur.

2) The program handles the error in a method that is higher in the chain of method calls. Suppose `main` calls `f`, and `f`, in turn, calls `g`. In this case, we have the following chain of method calls:

```
main → f → g
```

If an error occurs during the execution of `g`, the program could handle the error higher up in the chain of method calls.

```
 1 class TestException1  // Illustrates exceptions
 2 {
 3    public static void main (String[] args)
 4    {
 5       System.out.println("Start of main");
 6       f();
 7       System.out.println("End of main"); // no go
 8    }
 9    //-----------------------------------
10    public static void f()
11    {
12       System.out.println("Start of f");
13       g();
14       System.out.println("End of f");// not executed
15    }
16    //-----------------------------------
17    public static void g()
18    {
19       System.out.println("Start of g");
20       int x;
21       x = 5/0;                  // Exception thrown
22       System.out.println("End of g"); // no go
23    }
24 }
```

Results

```
Start of main
Start of f
Start of g


Exception in thread "main"
        java.lang.ArithmeticException: / by zero
        at TestException1.g(TestException1.java:21)
        at TestException1.f(TestException1.java:13)
        at TestException1.main(TestException1.java:6)
```

```
17     public static void g()
18     {
19        System.out.println("Start of g");
20        int x;
21        try
22        {
23           x = 5/0;   // Exception thrown
24        }
25        catch (ArithmeticException e)
26        {
27           System.out.println("In catch block");
28           System.out.println(e.getMessage());
29        }
30        System.out.println("End of g");
31     }
32  }
```

Results

```
/ by zero

Start of main
Start of f
Start of g
In catch block
/ by zero
End of g
End of f
End of main
```

```
 3    public static void main (String[] args)
 4    {
 5       System.out.println("Start of main");
 6       try
 7       {
 8          f();
 9       }
10       catch (ArithmeticException e)
11       {
12          System.out.println(e.getMessage());
13       }
14       System.out.println("End of main");
15    }
```

Results

```
Start of main
Start of f
Start of g
/ by zero
End of main
```

Creating and Throwing Exceptions

```
 1 import java.util.Scanner;
 2 class TestException4
 3 {
 4    public static void main(String[] args)
 5    {
 6       int grade = getGrade();
 7       System.out.println("grade = " + grade);
 8    }
 9    //-----------------------------------
10    public static int getGrade()
11    {
12       Scanner kb = new Scanner(System.in);
13
14       System.out.println("Enter grade");
15       int grade = kb.nextInt();
16
17       // throw the exception that the new
18       // operator creates
19       if (grade < 0)
20          throw new RuntimeException("Invalid grade");
21
22       return grade;
23    }
24 }
```

Results

```
java TestException4
Enter grade
-1
Exception in thread "main"
     java.lang.RuntimeException: Invalid grade at
     TestException4.getGrade(TestException4.java: 20)
     at TestException4.main(TestException4.java: 6)
```

`throws` clause

Indicates exception might propagate to caller

```
public static void f() throws IOException
{
   ...
}
```

Creating your own exception classes

```
 1 import java.util.Scanner;
 2 class TooSmallException extends Exception
 3 {
 4    public TooSmallException()
 5    {
 6       super("Number too small");
 7    }
 8    //-----------------------------------
 9    public TooSmallException(String msg)
10    {
11       super(msg);
12    }
13 }
```

```
15 class TooLargeException extends Exception
16 {
17    public TooLargeException()
18    {
19       super("Number too large");
20    }
21    //-----------------------------------
22    public TooLargeException(String msg)
23    {
24       super(msg);
25    }
26 }
```

```
28 class TestException5
29 {
30      public static void main(String[] args)
31      {
32         Scanner kb = new Scanner(System.in);
33         System.out.println("Enter number");
34         int x = kb.nextInt();
35         try
36         {
37            if (x < 100)
38               throw new TooSmallException();
39            if (x > 200)
40               throw new TooLargeException(x + " too large");
41         }
42         catch (TooSmallException e)
43         {
44            System.out.println(e.getMessage());
45            System.out.println("Enter larger number");
46         }
47         catch (TooLargeException e)
48         {
49            System.out.println(e.getMessage());
50            System.out.println("Enter smaller number");
51         }
52      }
53 }
```

```
Enter number
2
Number too small
Enter larger number

Enter number
500
500 is too large
Enter a smaller number
```