Example 11.2

Program to illustrate the try-catch block.

```
class exception1
{
        public static void main(String args[])
        {
                int  x=50, y=0;
                int result=0;
                try
                {
                        result=x/y;        // Found Arithmetic Exception
                                           // Exception Object is created & thrown
                        System.out.println ("We are in try block");
                }

              catch (ArithmeticException   e)// catching Arithmetic Exception object
              {
                      System.out.println("Arithmetic exception occurred");
                      System.out.println(e);
              }
              System.out.println("This is the last statement");
        }
}
```

Output :

```
Arithmetic exception occurred
java.lang.ArithmeticException:  /by zero
This is the last statement
```

```
class exception2
{
  public static void main(String args[])
  {
    int  a[ ] = {10, 20, 30};
    int total=0;
    try
    {
      total=a[0]+a[1]+a[2]+a[3];  //Found ArrayIndexOutOfBoundsException so
                                  //An Object is created & thrown

    }

    catch(ArrayIndexOutOfBoundsException e)
    {
      System.out.println("You are trying to use an invalid index");
      System.out.println(e);
    }
    System.out.println("Total Marks : " + total);
    System.out.println ("This is the last statement");
  }
}
```

Output :

```
You are trying to use an invalid index
java.lang.ArrayIndexOutOfBoundsException: 3
Total Marks : 0
This is the last statement
```

11.4.2 Using Multiple Catch Blocks

We can use more than one catch clause in a single try block however every catch block can handle only one type of exception.

In many cases, more than one exception could be raised by a single piece of code. In order to handle such a situation multiple catch clauses, each catching a different exception type can be specified.

Example 11.4

Program to illustrate multiple catch block.

```
class exception2
{
 public static void main(String args[])
 {
  int a[ ] = {10, 20, 30};
  int total=0;


  try
  {
   total=a[0]+a[1]+a[2]+a[3]; //Found ArrayIndexOutOfBoundsException so
                 //An Object is created & thrown
   int result=50/0;      //Arithmetic Exception, but unreachable
  }

  catch (ArithmeticException  e) //catching Arithmetic Exception object
  {
   System.out.println("Arithmetic exception occurred");
   System.out.println(e);
  }

  catch(ArrayIndexOutOfBoundsException e)
  {
   System.out.println("You are trying to use an invalid index");
   System.out.println(e);
  }
  System.out.println("Total Marks : " + total);
  System.out.println ("This is the last statement");
 }
}
```

Output :

```
You are trying to use an invalid index
java.lang.ArrayIndexOutOfBoundsException: 3
Total Marks : 0
This is the last statement
```

11.5 Using Finally Statement

The **finally** statement is used to handle an exception that is not caught by any of the previous catch statements. A **finally** block can be used to handle any exception generated within a try block. The finally block will be executed whenever execution leaves a **try/catch** block, no matter what conditions cause it. That is, whether the **try** block ends normally or because of an exception, the last code executed is that defined by **finally**. Java finally block must be followed by try or catch block. The general from of **try/catch** that includes **finally** is :

```
try
{
        . . .
                        // try code
        . . .
}
catch (. . .)
{
        . . .
                        // catch code
        . . .
}
catch (. . .)
{
        . . .
                        // catch code
        . . .
}
. . .
. . .
finally
{
        . . .
                        // finally code
        . . .
}
```

Since **finally** block is guaranteed to execute. As a result, we can use it to perform certain house-keeping operations such as closing files and releasing system resources.

Example 11.5

Program to illustrate the concept of finally block.

```
class exception1
{
   public static void main(String args[])
   {
      int  x=50, y=0;
      int result=0;
      try
      {
         result=x/y;        //Found Arithmetic Exception
                            //Exception Object is created & thrown
         System.out.println ("We are in try block");
      }

      catch (ArithmeticException   e)//catching Arithmetic Exception object
      {
         System.out.println("Arithmetic exception occurred");
         System.out.println(e);
      }
```

```
            finally
            {
                System.out.println("Finally Block is Always Executed");
            }
            System.out.println("This is the last statement");
        }
    }
```

Output :

```
Arithmetic exception occured
java.lang.ArithmeticException: /by zero
Finally Block is Always Executed
This is the last statement
```

Example 11.6

Let's see the java finally example where no catch statement is used.

```
class exception3
{
    public static void main(String args[])
    {
        int  x=50, y=0;
        int result=0;
        try
        {
            result=x/y;             //Found Arithmetic Exception
                                //Exception Object is created & thrown
            System.out.println ("We are in try block");
        }


        finally
        {
            System.out.println("Finally Block is Always Executed");
        }
        System.out.println("This is the last statement");
    }
}
```

Output :

```
Finally Block is Always Executed
Execption in thread "main" java.lang.ArithmeticException: / by zero
at exception3.main(chap115.java:9)
```