Chapter

7B EXCEPTION HANDLING

- *Exception Introduction*
- *Syntax Errors*
- *Logical and Runtime Error*
- *Handling Exceptions in the Java Way*
- *Keywords – throw and throws*
- *Program demonstrating the application of 'throw' and 'throws'*

Exceptions are erroneous situations in a program. Java has ways to handle errors. The compiler can detect syntax errors during compilation. Logical errors can appear in various forms. As we know that an error in the logic of a program can be the cause of wrong results. There are some other external factors that cause a threat to the robustness of a program.

Java provides ways to handle exceptions in programs.

Syntax Errors

The syntax errors that the compiler can detect during compilation are shown below. They come with an explanatory statement that explains the reason of error. The compilation process makes it easy to detect syntax errors.

Given below are some common syntax errors reasons :

Shown below are some of the most common syntax errors.

- Wrong spelling of keywords.

Note :

- *That Java is case sensitive. One has to be careful while writing long class names where both upper and lower case characters are used.*

For Example : Input Stream Reader

- **Missing semicolon**

Statement terminator indicated by a ; (semicolon) must be present to mark the end of any command. But conditions and loops should not be terminated as then they get disconnected with their block.

- **Missing or mismatching brackets () or {}**

() brackets are used in calculations. Correct spacing saves this error.

{} brackets are used for making blocks. Indentation helps to make the blocks distinct and identifiable. [Indentation is the process of changing the vertical margin of each block.]

- **Data type mismatch** : Care must be taken towards verifying the data types of variables when more than one is involved. The compiler helps out with implicit data type conversions and the programmer should perform explicit data type conversions as per requirement.

- **Unclosed character or string :** Caused when the compiler encounters any character (' ') or string (" ") not closed. Characters are single and strings are multiple characters.

- **Forgetting to import a package** : If one forgets to put the required import statement at the beginning of a program, then the compiler will respond with a message such as:

"Class '...' not found in type declaration"

Note :

- *That java.lang is imported automatically and, hence, does not need an import statement.*

- **Method prototype and method call** : A method prototype should be given return type, acceptable method name, parameter list along with their data types.

During method call, actual parameters should be sent without the data types, with the number and type matching.

- **Accessing a variable out of its scope** : A variable is bound within its block. Variables present inside one method are not accessible in another method.

- **Array index out of bounds** : In case of arrays, the cell numbers begin with 0. In case of string handling also, character positions are numbered from 0 onwards.

Logical Error

These errors cannot be directly shown by the compiler. To detect logical error, the programmer has to verify each step of the execution and verify the values.

Runtime Error

There are external reasons that can cause a program to get unexpectedly terminated. The exception handling statements in Java helps to maintain the robustness of Java programs and can provide ways to handle these situations.

Handling Exceptions in the Java Way

Java provides a class called Exception class that comprises of the various error handling subclasses and methods.

In a method, an exception can be identified and treated with the help of a try .. catch block. The probable error causing code is enclosed in a try .. catch block. In case any error is generated, then it is caught by the catch block.

```
try
{
  //program code
} catch(Exception e1)
{
  //Catch block
}
```

In a method, exceptions can also be handled with the use of "throws" keyword written in the method prototype. The keyword "throws" indicates the compiler to just keep the error aside. The syntax is

```
void fnMethod ( int a, int b) throws Exception
{
  //program code
}
```

Keywords throw and throws

The keyword "throws" is used by the compiler to generate a predefined error. For example, an IOException (Input Output Exception) is generated when an alphabet is input in a variable of integer type.

There is another keyword "throw" that allows the programmer to create a new error situation that is specific to the program problem. For example, in a banking program, negative value in an account might be termed as an InvalidAmountException. [explained in higher programming].

Program Demonstrating the Application of 'Throw' and 'Throws'

```
// File Name FailMarks.java
import java.io.*;
public class clFailMarks extends Exception
{
    private int marks ;
    public clFailMarks(int marks)
    {
        this.marks = marks ;
    }
    public double getMarks()
    {
        return marks;
    }
}
// File Name MarksProcessing.java
import java.io.*;
public class MarksProcessing
{
    private int Roll , marks ;
```

```
    MarksProcessing (int marks)
    {
        this.marks = marks;
    }
    public void verify () throws clFailMarks
    {
        if(marks >= 40 )
        {
            System.out.println("\n Congrats ! U have passed with " + marks + "
                                                                    marks.");
        }
        else
        {
            int needs = 40 - marks;
            throw new clFailMarks(needs);
        }
    }
}
// File Name MarksVerify.java
public class MarksVerify
{
    public static void main(String [] args)
    {
        MarksProcessing mm1 = new MarksProcessing(50);    // first marks = 50
        MarksProcessing mm2 = new MarksProcessing(20);    // second marks = 20
        MarksProcessing mm3 = new MarksProcessing(70);    // third marks = 70
        try
        {
            System.out.println("\n St 1 : Marks Obtained ... ");
            mm1.verify();
            System.out.println("\n St 2 : Marks Obtained ... ");
            mm2.verify();
            System.out.println("\n St 3 : Marks Obtained ... ");
            mm3.verify();
        }catch(clFailMarks e)
        {
            System.out.println("Sorry, but you are short by " + e.getMarks());
        }
    }
}
```

OUTPUT

St 1 : Marks Obtained ...

Congrats ! U have passed with 50 marks.

St 2 : Marks Obtained ...

Sorry, but you are short by 20.0

[Note: The above code can be further modified.]

Program : A group of friends were playing a game in which n numbers were required which should be in ascending order. If anyone entered erroneous data, the process would terminate throwing an error. Apply the above in a program.

Exercise

1. State TRUE or FALSE. Correct the false statements.

(a) Return type of Math.pow() is int.

(b) String is a primitive.

(c) R.length expects R to be an integer array.

(d) void main(void) causes syntax error.

(e) In an array, different types of data can be stored.

2. Observe the code snippet and identify the error, if any. Also write the aim of the code.

(a) int ta = { 12, 34, 56, 78, 90 } ;

(b) String ts = String (qs.length()) ;

(c) System.out.println(m + " x " + i + " = " p);

(d) r = (a != 0 && b != 0) ? a/b : 0 ;

(e) m = (a>b)? (b>c ? a : (a>c ? a : c)) : ((b>c ? b : (c>a ? c : a))) ;

3. Given below are programs that contain some syntax and some logical errors. Debug them.

(a) Verify whether a date is valid or not.

```
//input a date and verify whether it is valid or not
import java.io.*;
classVerifyD
{
   void main()
   {
      intdt, mn, yr;  Scanner sc = new Scanner();
      System.out.println("Enter a date " );
      dt = sc.nextint();
      mn = sc.nextint();
      yr = sc.nextint();
      if(mn>=1 | | mn<= 12)
```

```
{
    if(mn == 2 )
    {
        if ( yr % 4 = 0 &&yr % 100 != 0  || yr % 400 = 0 )
        {
            if(dt>=1 || dt<=29 )
            {
            System.out.print( "Date is valid");
            }
            else
            {
            System.out.print( "Date is NOT valid");
            }
        }
    }
    else
    {
    if ( mn == 1 &&mn == 3 &&mn == 5 &&mn == 7 &&mn == 8 &&mn == 10 &&mn == 12 )
    {
        if(dt>=1 || dt<=31 )
        {
            System.out.print( "Date is valid");
        }
        else
        {
            System.out.print( "Date is NOT valid");
        }
    }
    else
    {
        if(dt>=1 || dt<=30 )
        {
            System.out.print( "Date is valid");
        }
        else
        {
            System.out.print( "Date is NOT valid");
        }
    }
  }
 }
}
```

(b) A Trimorphic number is such that its cube ends with itself. Input a number and verify whether it is Trimorphic or not.

```
import java.io.*;
class TriMorPhic
{
    void digits(int N)
    {
        int c = 0;
        while(N >=0)
        {
            N = N/10;
            c++;
        }
    }
    void main()
    {
        int N; Scanner sc = new Scanner();

        System.out.println("Enter a number " );
        N = sc.nextint();

        int CN = Math.pow(N,3);
        if( N == CN % Math.pow(10, c) )
        {
            System.out.print( "The number is TriMorPhic ");
        }
        else
        {
            System.out.print( "The number is non - TriMorPhic ");
        }
    }
}
```

(c) Print the series 1 2 5 10 17 26 37 ... upto n terms

```
import java.io.*;
class Series1
{
    void main()
    {
        int N; Scanner sc = new Scanner();
        System.out.println("How many terms " );
        N = sc.nextint();
        int V = 1;
        for(int j = 1; j<=N ; j++)
        {
            System.out.print(" " + V);
            V = V + 2;
        }
    }
}
```

(d) **Print the series 3 3 6 9 9 12 15 15 ... upto n terms**

```
import java.io.*;
class Series2
{
    void main()
    {
        int N; Scanner sc = new Scanner();
        System.out.println("How many terms " );
        N = sc.nextint();
        int w1 = 3, w2=3 ;
        for(int j = 1; j<=N ; j++)
        {
            System.out.print(" " + w1 + " " + w2);
            w1 = w1 + 0;
            w2 = w2 + 3;
        }
    }
}
```

❑❑