11.5.2 Using Throw Keyword

Till now we have been catching exceptions thrown by the Java run-time system. The **throw** keyword is used to explicitly throw an exception. We can throw either checked or unchecked exception by using throw keyword.

Therefore, **throw** keyword is used to manually throw an exception. To throw an exception find the appropriate exception class. Now create an object of that class and use the throw statement. The flow of execution immediately stops after the **throw** statement; any subsequent statements are not executed.

Syntax is :

throw instance ;

where **instance** is an object of type **Throwable.** Commonly, a **new** statement is used to create an instance. We can also create a **Throwable** object using a parameter into a **catch** clause.

Example 11.8

(a) throw new ArithmeticException( );

(b) throw new NullPointerException( );

In example (a), the **throw** statement is manually throwing **ArithmeticException.** In example (b), the **throw** statement is manually throwing **NullPointerException.**

Example 11.9

In this example, we have created the validate method having integer parameter. If the age is less than 18, we are throwing the ArithmeticException otherwise print a message.

```
class Exception5
{
    static void validate(int age)
    {
        if(age<18)
            throw new ArithmeticException("Not Valid To Vote");
        else
            System.out.println("Welcome to Vote");
    }
    public static void main(String agrs[])
    {
        validate(15);
        System.out.println("Last statement");
    }
}
```

Output :

```
Exception in thread "main" java.lang.ArithmeticException: NotValid To Vote
at Exception5.validate(chap117.java:6)
at Exception5.main(chap117.java:12)
```

11.5.3 Using Throws Keyword

The **throws** keyword is used to declare an exception. Throws is an alternate way to indicate that a method may possibly throw an exception. Any exception that is thrown out of a method must be specified as such by a **throws** clause. This is possible by adding the throws keyword after the signature of the method and followed by the name of one or more exceptions.

Syntax is :

```
return type  method_name (parameter_list) throws exception_list
{
    ...
    ...        // body of the method
}
```

Here, **exception_list** is a comma separate list of the exceptions that a method can throw.

Example 11.12

```
import java.io.*;
class Sample1
{
    void method() throws IOException
    {
        throw new IOException("device error");
    }
}
class Testthrows3
{
    public static void main(String args[]) throws IOException //declare exception
    {
        try
        {
            Sample1 obj=new Sample1();
            obj.method();
        }
        catch(Exception e)
        {
            System.out.println("Exception Handled");
        }
        System.out.println("Last Statement");
    }
}
```

Output :

```
Exception Handled
Last Statement
```

5.4 Difference between Throw and Throws

Differences between throw and throws keywords are given in table 11.1.

TABLE 11.1

Throw	Throws
1. Java throw keyword is used to explicitly throw an exception.	1. Java throws keyword is used to declare an exception.
2. Throw is followed by an instance.	2. Throws is followed by class.
3. Throw is used within the method.	3. Throws is used with the method signature.
4. You cannot throw multiple exceptions.	4. You can declare multiple exceptions e.g. public void method( ) throws IOException, SQLException.