\

Batch: B1 **Roll No.: 1711072**

Experiment / assignment / tutorial No. 06

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

TITLE :User Defined Exception

AIM: Write a program which accepts marks of a student (between 0 to 100) and checks whether it is within the range or not. If it is within the range then it displays "marks entered successfully", if not then it throws the exception of user defined class "MarksOutOfRangeException". The class should contain appropriate toString method to describe object the class with the out of range marks entered by the user.

Expected OUTCOME of Experiment:

CO4: Demonstrate programs on interface, exceptions, multithreading and applets.

Books/ Journals/ Websites referred:

1. Ralph Bravaco , Shai Simoson , "Java Programing From the Group Up" Tata McGraw-Hill.

2. Grady Booch, Object Oriented Analysis and Design .

Pre Lab/ Prior Concepts:

Exceptions:

An exception (or exceptional event) is a problem that arises during the execution of a program. When an **Exception** occurs, the normal flow of the program is disrupted and the program/Application terminates abnormally, which is not recommended, therefore, these exceptions are to be handled.

An exception can occur for many different reasons. Following are some scenarios where an exception occurs.

- A user has entered an invalid data.
- A file that needs to be opened cannot be found.
- A network connection has been lost in the middle of communications or the JVM has run out of memory.

Some of these exceptions are caused by user error, others by programmer error, and others by physical resources that have failed in some manner.

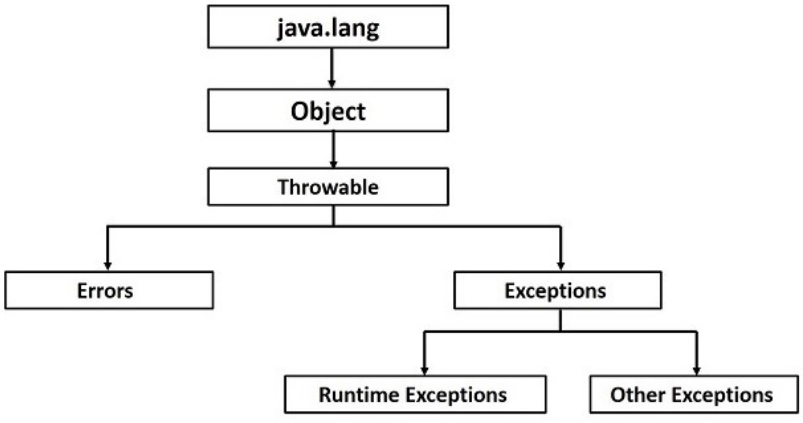
Based on these, we have three categories of Exceptions. You need to understand them to know how exception handling works in Java.

- **Checked exceptions** − A checked exception is an exception that occurs at the compile time, these are also called as compile time exceptions. These exceptions cannot simply be ignored at the time of compilation, the programmer should take care of (handle) these exceptions.

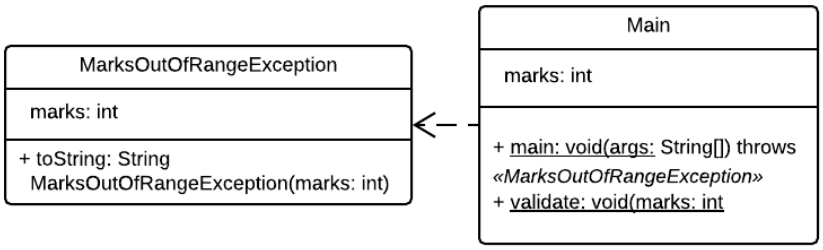
For example, if you use **FileReader** class in your program to read data from a file, if the file specified in its constructor doesn't exist, then a *FileNotFoundException* occurs, and the compiler prompts the programmer to handle the exception.

- Unchecked exceptions − An unchecked exception is an exception that occurs at the time of execution. These are also called as Runtime Exceptions. These include programming bugs, such as logic errors or improper use of an API. Runtime exceptions are ignored at the time of compilation.

For example, if you have declared an array of size 5 in your program and trying to call the 6th element of the array then an *ArrayIndexOutOfBoundsException* occurs.



Class Diagram:



Implementation details:

Variation 1:

```
import java.util.*;
class MarksOutOfRangeException extends Exception{
  int marks;
  MarksOutOfRangeException(int marks){
    this.marks=marks;
  }
  public String toString(){
    return "Marks out of range [0-100] because entered marks is " +
marks;
  }
}
class Main{
  public static void main(String[] args) throws MarksOutOfRangeException{
    int marks;
    Scanner sc=new Scanner(System.in);
    System.out.print("Enter your marks: ");
    marks=sc.nextInt();
    try{
      if(marks>100 || marks<0)
        throw new MarksOutOfRangeException(marks);
      else
        System.out.println("Marks entered successfully");
    }
    catch(Exception m){
      System.out.println("Error has occurred! Check it out: "+m);
    }
  }
```

```
}
```

Variation 2:

```
import java.util.*;
class MarksOutOfRangeException extends Exception{
  int marks;
  MarksOutOfRangeException(int marks){
    this.marks=marks;
  }
  public String toString(){
    return "Marks out of range [0-100] because entered marks is " +
marks;
  }
}
class Main{
  public static void main(String[] args) throws MarksOutOfRangeException{
    int marks;
    Scanner sc=new Scanner(System.in);
    System.out.print("Enter your marks: ");
    marks=sc.nextInt();
    validate(marks);
  }
  public static void validate(int marks){
    try{
      if(marks>100 || marks<0)
        throw new MarksOutOfRangeException(marks);
      else
        System.out.println("Marks entered successfully");
    }
    catch(Exception m){
      System.out.println("Error has occurred! Check it out: "+m);
    }
  }
}
```

Variation 3:

```
import java.util.*;
class Main{
  public static void main(String[] args){
    int marks;
    Scanner sc=new Scanner(System.in);
    System.out.print("Enter your marks: ");
```

```
    marks=sc.nextInt();
      if(marks>100 || marks<0)
        throw new ArithmeticException();
  }
}
```

Output Screens:

Variation 1:

```
Enter your marks:  92
Marks entered successfully
```

```
Enter your marks:  900
Error has occured! Check it out: Marks out of range [0-100]
because entered marks is 900
```

Variation 2:

```
Enter your marks:  92
Marks entered successfully
```

```
Enter your marks:  900
Error has occured! Check it out: Marks out of range [0-100]
because entered marks is 900
```

Variation 3:

```
Enter your marks:  92
Marks entered successfully
```

```
 Enter your marks:  293
 Exception in thread "main" java.lang.ArithmeticException
         at Main.main(Main.java:10)
exit status 1
```

Conclusion: **All the three variations ran successfully and we conclude that:**

1. **Predefined exceptions don't need to be thrown using "throws" or use try-catch. User defined exceptions must be thrown using "throws" keyword.**
2. **If using user defined exceptions, and if you don't use try-catch as well as "throws" keyword, then you get error that exception isn't handled.**
3. **"Throw" and "throws" don't handle the exception, they just throw the exception.**
4. **Try-catch handles the exception well and the code after catch gets executed.**

Post Lab Descriptive Questions (Add questions from examination point view)

1. Compare throw and throws.

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

2. Explain how to create a user define exception and explicitly throwing exception in the program with simple example.

Ans. To create a user-defined exception, create a class with the name of the exception you want to create and this class should be inheriting the **"Exception"** class using **"extends"** keyword. To throw an exception explicitly, we use the **"throw"** keyword.
For eg:

```
class MyException extends Exception{
//few lines of code
}
class Main{
        public static void main(String[] args){
                throw new MyException();
        }
}
```

Date: 13/10/2018 **Signature of faculty in-charge**