St Francis Institute of Technology, Mumbai-400 103

Class: SE-ITA/ITB Semester: III; A.Y. 2020-2021
Subject: Java Labs

Title-6: Java Program to implement Exception Handling (Inbuilt and User Defined Exceptions).

1. **Aim:**
   i. Write a program to demonstrate checked Exception Handling using nested try, catch statements.
   ii. Write a Java Program to Create Account with 1000 Rs Minimum Balance, Deposit Amount, Withdraw Amount and Also Throws LessBalanceException. It has a Class Called LessBalanceException Which returns the Statement that Says WithDraw Amount(_Rs) is Not Valid. It has a Class Which Creates 2 Accounts, Both Account Deposite Money and One Account Tries to WithDraw more Money Which Generates a LessBalanceException. Take Appropriate Action for the Same.

2. **Prerequisite:** Knowledge of Exception Handling in Java.

3. **Requirements:** Personal Computer (PC), Windows Operating System, Net beans 8.0.

4. **Pre-Experiment Exercise:**
   Theory:

   a. **Exception Keyword:**
      i. **Throw**- used to throw an exception explicitly. Only object of Throwable class or its sub classes can be thrown. Program execution stops on encountering **throw** statement, and the closest catch statement is checked for matching type of exception.
      ii. **Throws**- Any method that is capable of causing exceptions must list all the exceptions possible during its execution, so that anyone calling that method gets a prior knowledge about which exceptions are to be handled. A method can do so by using the **throws** keyword.
      iii. **Finally**- A finally keyword is used to create a block of code that follows a try block. A finally block of code is always executed whether an exception has occurred or not. Using a finally block, it lets you run any cleanup type statements that you want to execute, no matter what happens in the protected code. A finally block appears at the end of catch block.
      iv. **Try**- The try block contains set of statements where an exception can occur. A try block is always followed by a catch block, which handles the exception that occurs in associated try block. A try block must be followed by catch blocks or finally block or both.
      v. **Catch**- A catch block is where you handle the exceptions; this block must follow the try block. A single try block can have several catch

blocks associated with it. You can catch different exceptions in different catch blocks.

5. Laboratory Exercise

A. Procedure

i. Open Net beans for Java.
ii. Open File and Create New Java Project.
iii. Inside the Java Project rename give name to your Java Class.
iv. Click on Finish.
v. Type the Java Code in the opened class.
vi. Save the code by pressing Ctrl+S.
vii. Run the code by pressing Shift+F6.

B. Program code with comments:

Write and execute your program code to achieve the given aim and attach it **with your own comments with neat indentation**.

6. Post-Experiments Exercise

A. Extended Theory:

1. Explain the hierarchy of Exception Handling Classes with the help of a diagram.
2. List and explain runtime errors.

B. Results/Observations/Program output:

Present the program input/output results and comment on the same.

C. Questions/Programs:

1. Write a Java Program to calculate the Result. Result should consist of name, seatno, date, centre number and marks of sem-2 examination. Create a user defined exception class MarksOutOfBoundsException, If Entered marks of any subject is greater than 100 or less than 0, and then program should create a user defined Exception of type MarksOutOfBoundsException and must have a provision to handle it.

D. Conclusion:

1. Write what was performed in the experiment/program.
2. What is the significance of experiment/program?
3. Mention few applications of what was studied.

E. References

1. Balguruswamy, “Programming with java A primer”, Fifth edition, Tata McGraw Hill Publication.
2. Let Us Java-Yashwant Kanetkar.
3. Learn to Master JAVA, from Star EDU solutions , by ScriptDemics.
4. Java 8 Programming-Black Book,by-Dreamtech Publications.
5. www.programmingsimplified.com
6. www.javatpoint.com

Program 1:

```
//Write a program to demonstrate checked Exception Handling using nested try,
//catch statements.

class Exp6_1 {

    // main method
    public static void main(String args[])
    {
        // Main try block
        try {

            // initializing array
            int a[] = { 1, 2, 3, 4, 5 };

            // trying to print element at index 5
            System.out.println(a[5]);

            // try-block2 inside another try block
            try {

                // performing division by zero
                int x = a[2] / 0;
            }
            catch (ArithmeticException e2) {
                System.out.println("division by zero is not possible");
            }
        }
        catch (ArrayIndexOutOfBoundsException e1) {
            System.out.println("ArrayIndexOutOfBoundsException");
            System.out.println("Element at such index does not exists");
        }
    }
    // end of main method
}
```

Output:

```
Microsoft Windows [Version 10.0.19042.662]
(c) 2020 Microsoft Corporation. All rights reserved.

D:\College\JAVA\Experiments\Exp6>javac Exp6_1.java

D:\College\JAVA\Experiments\Exp6>java Exp6_1
ArrayIndexOutOfBoundsException
Element at such index does not exists

D:\College\JAVA\Experiments\Exp6>
```

Program 2:

```
/**
Write a Java Program to Create Account with 1000 Rs Minimum Balance,
Deposit Amount, Withdraw Amount and Also Throws LessBalanceException.
It has a Class Called LessBalanceException Which returns the Statement that
Says WithDraw Amount(_Rs) is Not Valid. It has a Class Which Creates 2
Accounts, Both Account Deposit Money and One Account Tries to
WithDraw more Money Which Generates a LessBalanceException. Take
Appropriate Action for the Same.
 */

import java.util.Scanner;

class LessBalanceException extends Exception{
    String msg;
    LessBalanceException(String msg){
        super(msg);
    }
}
public class Exp6_2 {
    public static void main(String[] rags)throws LessBalanceException{
       double deposit_amount,withdraw_amount;
       Scanner sc=new Scanner(System.in);
```

```
        try{
        System.out.println("Enter deposit amount");
            deposit_amount=sc.nextDouble();
            System.out.println("Enter Amount to be withdrawn");
            withdraw_amount=sc.nextDouble();
                if(deposit_amount<=1000)
                        throw new LessBalanceException("Amount cannot be
withdrawn");
                else
                    {
                            System.out.println("Amount"+withdraw_amount+"\t
withdrawn successfully");
                    }
        }
       catch(LessBalanceException ex){
                    System.out.println(ex.getMessage());
                    }

        }
    }
```

Output:

```
D:\College\JAVA\Experiments\Exp6>javac Exp6_2.java

D:\College\JAVA\Experiments\Exp6>java Exp6_2
Enter deposit amount
2000
Enter Amount to be withdrawn
500
Amount500.0      withdrawn successfully

D:\College\JAVA\Experiments\Exp6>java Exp6_2
Enter deposit amount
1000
Enter Amount to be withdrawn
200
Amount cannot be withdrawn
```

Questions:

Question 1:

```
import java.util.Scanner;

class MarksOutOfBoundsException extends Exception{
    String msg;
    MarksOutOfBoundsException(String msg){
        super(msg);
    }
}
public class Questions {
    public static void main(String[] args)throws
        MarksOutOfBoundsException{
        double seatno, centre_num, marks;
        Scanner scanner=new Scanner(System.in);

        try{
            System.out.println("Enter your name:");
```

```
            String name=scanner.nextLine();

            System.out.println("Enter the seatno:");
            seatno=scanner.nextDouble();

            System.out.println("Enter the date:");
            String date=scanner.next();

            System.out.println("Enter the centre_num:");
            centre_num=scanner.nextDouble();

            System.out.println("Enter the marks:");
            marks=scanner.nextDouble();

            if(marks<0 || marks>100)
                throw new MarksOutOfBoundsException("Marks cannot be less than
zero or even cannot exceed 100");
            else{
                System.out.println("Details of Sem2");
                System.out.println("Name:"+name);
                System.out.println("Seatno:"+seatno);
                System.out.println("Date:"+date);
                System.out.println("Centre number:"+centre_num);
                System.out.println("Marks of sem 2:"+marks);

                }
        }
        catch(MarksOutOfBoundsException ex){
            System.out.println(ex.getMessage());
        }

        scanner.close();
    }
}
```

Output:

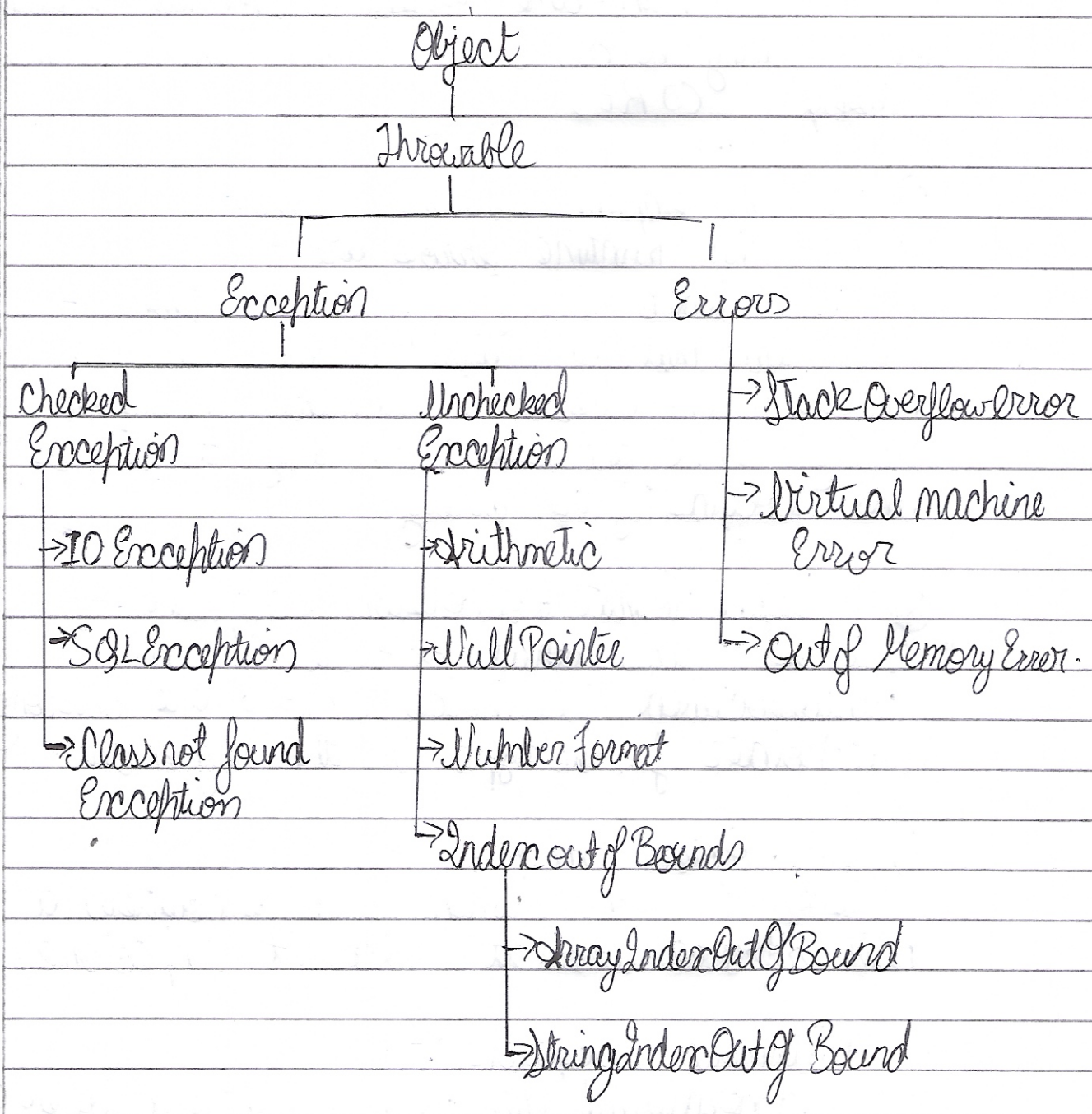
```
D:\College\JAVA\Experiments\Exp6>javac Questions.java

D:\College\JAVA\Experiments\Exp6>java Questions
Enter your name:
Yash
Enter the seatno:
1234
Enter the date:
01-01-2020
Enter the centre_num:
789
Enter the marks:
90
Details of Sem2
Name:Yash
Seatno:1234.0
Date:01-01-2020
Centre number:789.0
Marks of sem 2:90.0

D:\College\JAVA\Experiments\Exp6>java Questions
Enter your name:
Yash
Enter the seatno:
1234
Enter the date:
01-01-2020
Enter the centre_num:
789
Enter the marks:
-12
Marks cannot be less than zero or even cannot exceed 100

D:\College\JAVA\Experiments\Exp6>
```

Yash Mahajan SE IT B 04

6. Post Experiment Exercise.

A. Extended Theory:-

1) Explain the hierarchy of exception handling with the help of diagram



All exceptions and errors are sub classes of class Throwable, which is base class of hierarchy. One branch is headed by Exception. This class is used for exceptional conditions that user programs should catch, Null pointer is an example of such an exception. Another branch, Error are used by Java run-time system JVM to indicate errors having to do with the run-time environment itself (JRE)

2) List and explain runtime errors
The main runtime errors are :-
(1) Array index out of bounds Exception :-
This means that an attempt was made to access a non existential element. The element trying to be accessed is generally one outside the bounds of the array.

(2) Illegal Format Conversion Exception :-
This means that there is an unassociated placeholder or it's the wrong placeholder for the type in the argument.

4) Null Pointer Exception :-
This is thrown when Java encounters a null reference when it doesn't expect one.

5) Arithmetic Exception :-
This generally happens when you try to divide by 0.

Yash Mahajan SE IT B 04

6) Class cast Exception :-
This happens when you try to wrongly cast an object to a particular class. This means that your object is not an instance or a subclass of this class.

7) Stack Overflow Exception :-
This happens when ~~you~~ Java runs out of it's available memory.

D] Conclusion :-

In this experiment we have studied the important concept of ~~Ex~~ exception handling in Java. Exceptions are divided into two categories i.e. compiletime and runtime exceptions.

Exception handeling helps us maintain the desired output of the program even if unexpected events happen. If exceptions are handelled incorrectly the program may stop running.

Exception handling prevents a program from terminating ~~abrut~~ abruptly. It provides a proper method to handele all the compiletime and run time exceptions.