Batch: D2 **Roll No.: 16010122323**

Experiment / assignment / tutorial No.07

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

Signature of the Staff In-charge with date

TITLE : User Defined Exception

AIM: Create a user defined exception subclass TriangleException with necessary constructors and overridden toString method. Write a program which accepts three sides of a triangle and throws an object of TriangleException class if the triangle is not constructible otherwise it displays the type of the triangle and area of the triangle formed. On printing, the exception object should display exception name, appropriate message, and the sides responsible for exception.

Expected OUTCOME of Experiment:

CO1: Understand the features of object oriented programming compared with procedural approach with C++ and Java

CO4: Explore the interface, exceptions, multithreading, packages

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:

Exception handling in java is a powerful mechanism or technique that allows us to handle runtime errors in a program so that the normal flow of the program can be maintained. All the exceptions occur only at runtime. A syntax error occurs at compile time.

Exception in Java:

In general, an exception means a problem or an abnormal condition that stops a computer program from processing information in a normal way.

An exception in java is an object representing an error or an abnormal condition that occurs at

runtime execution and interrupts (disrupts) the normal execution flow of the program.

An exception can be identified only at runtime, not at compile time. Therefore, it is also called runtime errors that are thrown as exceptions in Java. They occur while a program is running.
For example:

- If we access an array using an index that is out of bounds, we will get a runtime error named ArrayIndexOutOfBoundsException.
- If we enter a double value while the program expecting an integer value, we will get a runtime error called InputMismatchException.

When JVM faces these kinds of errors or dividing an integer by zero in a program, it creates an exception object and throws it to inform us that an error has occurred.If the exception object is not caught and handled properly, JVM will display an error message and will terminate the rest of the program abnormally.
If we want to continue the execution of remaining code in the program, we will have to handle exception object thrown by error condition and then display a user-friendly message for taking corrective actions. This task is known as exception handling in java.

Types of Exceptions in Java

Basically, there are two types of exceptions in java API. They are:
1. Predefined Exceptions (Built-in-Exceptions)
2. Custom (User defined)Exceptions

Predefined Exceptions:
Predefined exceptions are those exceptions that are already defined by Java system. These exceptions are also called built-in-exceptions.Java API supports exception handling by providing the number of predefined exceptions. These predefined exceptions are represented by classes in java.
When a predefined exception occurs, JVM (Java runtime system) creates an object of predefined exception class. All exceptions are derived from java.lang.Throwable class but not all exception classes are defined in the same package. All the predefined exceptions supported by java are organized as subclasses in a hierarchy under the Throwable class.

All the predefined exceptions are further divided into two groups:

1. Checked Exceptions: Checked exceptions are those exceptions that are checked by the java compiler itself at compilation time and are not under runtime exception class hierarchy. If a method throws a checked exception in a program, the method must either handle the exception or pass it to a caller method.

2. Unchecked Exceptions: Unchecked exceptions in Java are those exceptions that are checked by JVM, not by java compiler. They occur during the runtime of a program. All exceptions under runtime exception class are called unchecked exceptions or runtime exceptions in Java.

Custom exceptions:
Custom exceptions are those exceptions that are created by users or programmers according to their own needs. The custom exceptions are also called user-defined exceptions that are

created by extending the exception class.
So, Java provides the liberty to programmers to throw and handle exceptions while dealing with functional requirements of problems they are solving.\

Exception Handling Mechanism using Try-Catch block:
The general syntax of try-catch block (exception handling block) is as follows:

Syntax:

```
try
{
  // A block of code; // generates an exception
}
catch(exception_class var)
{
  // Code to be executed when an exception is thrown.
}
```

Example:

```
public class TryCatchEx
{
public static void main(String[] args)
{
 System.out.println("11");
 System.out.println("Before divide");
 int x = 1/0;
 System.out.println("After divide");
 System.out.println("22");
          }
}
```

Output:

```
11
Before divide
Exception in thread "main" java.lang.ArithmeticException: / by zero
```

Class Diagram:

TriangleException
+str1: String
+ TriangleException (String): +toString():String

Triangle
+TypeTriangle(double, double, double) : void

Algorithm:

Step 1: Create a class that extends the class Exception and call the constructor of that class. In the constructor initialize the class variables.

Step 2: Use the tostring() method to describe the object of the class by returning value as “cannot form a triangle !”.

Step 3: In the main method of class Triangle input the sides from the user and check if they are valid in the try block.

Step 4: If the sides are invalid then throws an exception .In the catch block print the exception name and the message that needs to be printed along with it.

Step 5: If the sides entered form a valid triangle then call the function TypeTriangle.

Step 6: The TypeTriangle calculates the area of the triangle along with its type.

Implementation details :

```
import java.util.*;
import java.lang.Math;

class TriangleException extends Exception {
    String message;

    TriangleException(String message) {
        this.message = message;
    }

    @Override
    public String toString() {
        return message;
    }
}

public class Triangle {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Name: Vedansh Savla");
System.out.println("Roll Number: 16010122323");
System.out.println("Division: D2");
System.out.println("-----------------------------------------------
------------");
System.out.println("OOPM exp 7: Execption handling.");
System.out.println("Implementation details:");
System.out.println("-----------------------------------------------
------------");
        System.out.println("Enter side 1:");
        double a = scanner.nextDouble();
        scanner.nextLine();
```

```
        System.out.println("Enter side 2:");
        double b = scanner.nextDouble();
        scanner.nextLine();
        System.out.println("Enter side 3:");
        double c = scanner.nextDouble();
        scanner.nextLine();

        try {
            if (isValidTriangle(a, b, c)) {
                classifyTriangle(a, b, c);
            } else {
                throw new TriangleException("Cannot form a triangle!");
            }
        } catch (TriangleException e) {
            System.out.println("TriangleException:");
            System.out.println(e.toString());
        }
    }

    public static boolean isValidTriangle(double a, double b, double c) {
        return (a + b > c) && (a + c > b) && (b + c > a);
    }

    public static void classifyTriangle(double a, double b, double c) {
        System.out.println("The following Triangle is of type:");

        if (a == b && b == c) {
            System.out.println("Equilateral");
        } else if (a == b || b == c || a == c) {
            System.out.println("Isosceles");
        } else {
            System.out.println("Scalene");
        }

        System.out.println("Area of Triangle is:");
        double s = (a + b + c) / 2;
        double area = Math.sqrt(s * (s - a) * (s - b) * (s - c));
        System.out.println(area);
    }
}
```

Output:

```
java -cp /tmp/YLn4XYwXaH Triangle
Name: Vedansh Savla
Roll Number: 16010122323
Division: D2
--------------------------------------------------
OOPM exp 7: Exception handling.
Implementation details:
--------------------------------------------------
Enter side 1:
0
Enter side 2:
1
Enter side 3:
2
TriangleException:
Cannot form a triangle!
```

```
java -cp /tmp/YLn4XYwXaH Triangle
Name: Vedansh SavlaRoll Number: 16010122323
Division: D2
--------------------------------------------------
OOPM exp 7: Exception handling.
Implementation details:
--------------------------------------------------
Enter side 1:
3
Enter side 2:
3
Enter side 3:
3
The following Triangle is of type:
Equilateral
Area of Triangle is:
3.897114317029974
```

```
java -cp /tmp/YLn4XYwXaH Triangle
Name: Vedansh Savla
Roll Number: 16010122323
Division: D2
--------------------------------------------------
OOPM exp 7: Exception handling.
Implementation details:
--------------------------------------------------
Enter side 1:
1
Enter side 2:5
Enter side 3:
5
The following Triangle is of type:
Isosceles
Area of Triangle is:
2.48746859276655
```

```
java -cp /tmp/YLn4XYwXaH Triangle
Name: Vedansh Savla
Roll Number: 16010122323
Division: D2
--------------------------------------------------
OOPM exp 7: Exception handling.
Implementation details:
--------------------------------------------------
Enter side 1:
5
Enter side 2:
12
Enter side 3:
13
The following Triangle is of type:
ScaleneArea of Triangle is:
30.0
```

Post Lab Descriptive Questions

1. Compare throw and throws.

Sr. no.	Basis of Differences	throw	throws
1.	Definition	Java throw keyword is used throw an exception explicitly in the code, inside the function or the block of code.	Java throws keyword is used in the method signature to declare an exception which might be thrown by the function while the execution of the code.
2.	Type of exception Using throw keyword, we can only propagate unchecked exception i.e., the checked exception cannot be propagated using throw only.	Using throws keyword, we can declare both checked and unchecked exceptions. However, the throws keyword can be used to propagate checked exceptions only.	
3.	Syntax	The throw keyword is followed by an instance of Exception to be thrown.	The throws keyword is followed by class names of Exceptions to be thrown.
4.	Declaration	throw is used within the method.	throws is used with the method signature.
5.	Internal implementation	We are allowed to throw only one exception at a time i.e. we cannot throw multiple exceptions.	We can declare multiple exceptions using throws keyword that can be thrown by the method. For example, main() throws IOException, SQLException.

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

Java user-defined exception is a custom exception created and throws that exception using a keyword 'throw'. It is done by extending a class 'Exception'. An exception is a problem that arises during the execution of the program. In Object-Oriented Programming language, Java provides a powerful mechanism to handle such exceptions. Java allows to create own exception class, which provides own exception class implementation. Such exceptions are called user-defined exceptions or custom exceptions.
Example:

```
class SampleException{
public static void main(String args[]){
try{
throw new NewException(100);
}
catch(NewException e){
System.out.println(e) ;
```

```
}
}
}
class NewException extends Exception{
int a;
NewException(int b) {
a=b;
}
public String toString(){
return ("Status code = "+ a) ;
}
}
```

3. Suppose the statement2 causes an exception in following try-catch block:

```
try {
        statement1;
statement2;
statement3;
}
catch(Exception1 e1) {
}
catch(Exception2 e2){
}

statement4;
```

Answer the following questions:

- Will statement3 be executed?
  NO, the statement 3 will not be executed.
- If the exception is not caught, will statement4 be executed?
  NO, the statement 4 will not be executed.
- If the exception is caught in the catch block, will statement4 be executed?
  YES, if the exception is caught in the catch block, statement 4 will be executed.
- If the exception is passed to the caller, will the statement4 be executed?
  YES, it the exception is passed to the caller, statement 4 will be executed.

Conclusion:

User defined exceptions were understood and implemented successfully.

Date: 9th October 2023 **Signature of faculty in-charge**