**OBJECT ORIENTED PROGRAMMING**

**LAB# 13 TASKS**

## **Understanding the concept of Exceptions**

Example of program without exception handling

/\*

\* Program without Exception Handling

\*/

import java.util.Scanner;

public class WithoutException  
{  
    public static void main(String[] args)  
    {  
        Scanner input = new Scanner(System.in);

int donutCount, teaCount;  
        double donutsPerGlass;  
    
        System.out.println("Enter number of donuts:");  
        donutCount = input.nextInt( );  
  
        System.out.println("Enter number of cup of tea:");  
        teaCount = input.nextInt( );  
           
        if (teaCount < 1)  
        {  
            System.out.println("No Tea!");  
            System.out.println("Go buy some tea.");  
        }  
        else  
        {  
            donutsPerGlass = donutCount/(double)teaCount;  
            System.out.println(donutCount + " donuts.");  
            System.out.println(teaCount + " cup of tea.");  
            System.out.println("You have " + donutsPerGlass + " donuts for each cup of tea.");  
        }  
        System.out.println("End of program.");  
    }  
}

Compile and run the program according to the following

Enter number of donuts:  
2  
Enter number of cup of tea:  
0  
No Tea!  
Go buy some tea.  
End of program.

Example of program with exception handling

// An Example of Exception Handling

import java.util.Scanner;

public class ExceptionDemo {

    public static void main(String[] args){  
        Scanner input = new Scanner(System.in);

int donutCount, teaCount;  
        double donutsPerGlass;   
  
        try  //try block  
        {  
            System.out.println("Enter number of donuts:");  
            donutCount = input.nextInt( );  
  
            System.out.println("Enter number of cup of tea:");  
            teaCount = input.nextInt( );  
                  
            if (teaCount < 1)  
              //when the throw statement is executed, the block is stopped  
              throw new Exception("Exception: No Tea!");

            donutsPerGlass = donutCount/(double)teaCount;  
            System.out.println(donutCount + " donuts.");  
            System.out.println(teaCount + " cup of tea.");  
            System.out.println("You have " + donutsPerGlass + " donuts for each cup of tea.");  
        }  
   
        catch(Exception e)  // catch block  
        {  
            System.out.println(e.getMessage( ));  
            System.out.println("Go buy some tea.");  
        }  
  
        System.out.println("End of program.");  
    }  
}

Compile and run the program according to the following

First run:

Enter number of donuts:  
3  
Enter number of cup of tea:  
2  
3 donuts.  
2 cup of tea.  
You have 1.5 donuts for each cup of tea.  
End of program.

Second run:

Enter number of donuts:  
2  
Enter number of cup of tea:  
0  
Exception: No Tea!  
Go buy some tea.  
End of program.

In Java, methods that can fail are declared with a *throws* clause. The *throws IOException* clause means that this method can be invoked only within try clause.

/\*

\* Divide by zero error Exception Handling

\*/

//Example of Exception Handling  program  
import java.util.Scanner;

import java.io.\*;

public class TestExceptions   
{   
  static int getInt() throws IOException   
  {   
     Scanner input =  new Scanner(System.in);   
     System.out.print("Enter an integer: ");   
     int s = input.nextLine();   
     return (s);   
  }

  public static void main(String[] args)   
  { int n1=0, n2=1, n3=0;   
    try {   
      n1 = getInt();   
      n2 = getInt();   
      n3 = n1/n2;   
    }   
    catch (Exception e)   
    { System.out.println("[" + e + "]"); }   
    System.out.println(n1 + "/" + n2 + " = " + n3);   
  }   
}

Compile and run the program according to the following

Sample output (Everything works fine here);   
Enter an integer: 22   
Enter an integer: 7   
22/7 = 3

Sample output (error! - cannot divide by zero);   
Enter an integer: 22   
Enter an integer: 0   
[java.lang.ArithmeticException: / by zero]   
22/0 = 0

Sample output (Incorrect input data type);   
Enter an integer: 22   
Enter an integer: w   
[java.lang.ArithmeticException: w]   
22/1 = 0

1. Write a Fraction class that has a constructor that takes a numerator and a denominator. If the user passes in a denominator of 0, throw an exception of type std::runtime\_error (included in the stdexcept header). In your main program, ask the user to enter two integers. If the Fraction is valid, print the fraction. If the Fraction is invalid, catch a std::exception, and tell the user that they entered an invalid fraction.
2. Write a program which implements Banking System by having all standard functionalities and will be implemented by branches. Try to identify and implement user defined exceptions for the system.

Hint:

* Create Bank Class
* public void Create Account(){}
* public void deposit() throws Exception{

System.out.println(“Enter Amount to be deposited:);

If(deposit>100000)

throw new Exception(“\n you cant deposit this big amount”);

Else

balance=balance+deposit;

}

* Public void withdraw() throws Exception{} (same logic as deposit)