Aim : To learn Exception Handling.

Theory: Java exception handling enables your Java applications to handle errors sensibly.

An exception is a problem that arises during the execution of a program.

It is an abnormal condition that arises in a code sequence at run time. An exception can occur for many different reasons, including the following:

1.A user has entered invalid data.

2.A file that needs to be opened cannot be found.

3.A network connection has been lost in the middle of communications or the JVM has run out of memory.

4.If the input parameters are invalid (negative when expecting positive etc.).

Program1: Exception handling using try and catch block.

class Eg1

{

int a,d;

static void Test()

{

int d=0;

int a=50/d;

}

public static void main(String args[])

{

try

{

Eg1.Test();

}

catch(ArithmeticException e)

{

System.out.println("Divide by zero not allowed");

}

}}

Output:

Picked up JAVA\_TOOL\_OPTIONS: -javaagent:/usr/share/java/jayatanaag.jar

Divide by zero not allowed

Program 2: try block with multiple catch statements.

class Eg2

{

public static void main(String args[])

{

try

{

int d=0;

int a=50/d;

System.out.println("This will never be printed");

}

catch(ArithmeticException e)

{

System.out.println("Divide by zero not allowed");

}

System.out.println("After catch statement");

}}

Output:

Picked up JAVA\_TOOL\_OPTIONS: -javaagent:/usr/share/java/jayatanaag.jar

Divide by zero not allowed

After catch statement

Program 3:

class Multicatch

{

public static void main(String args[])

{

try

{

int a=args.length;

System.out.println("a:"+a);

int b=100/a;

int c[]={10};

c[42]=100;

}

catch(ArithmeticException e)

{

System.out.println("Divide by zero"+e);

}

catch(ArrayIndexOutOfBoundsException b)

{

System.out.println("Array index out of bounds"+b);

}

System.out.println("After catch block");

}}

Output:

Picked up JAVA\_TOOL\_OPTIONS: -javaagent:/usr/share/java/jayatanaag.jar

a:0

Divide by zerojava.lang.ArithmeticException: / by zero

After catch block

Program 4:Using nested try statements.

import java.io.\*;

class nested

{

public static void main(String args[]) throws IOException

{

int a=0,b=0,res;

BufferedReader br = new BufferedReader (new InputStreamReader (System.in)); String str;

System.out.println("Enter two numbers:");

try

{

str=br.readLine();

a=Integer.parseInt(str);

str=br.readLine();

b=Integer.parseInt(str);

try

{

res=a/b;

System.out.println("The Quotient=" + res);

}

catch(ArithmeticException ae)

{

System.out.println("Exception has occurred. You have entered the divisor as zero");

} }

catch (NumberFormatException ne)

{ System.out.println("Invalid number");

}

} }

OUTPUT: C:\Users\Mr.Lopes\Desktop>java nested

Enter two numbers:

5

1

The Quotient=5

C:\Users\Mr.Lopes\Desktop>javac nested.java

C:\Users\Mr.Lopes\Desktop>java nested

Enter two numbers:

8

0

Exception has occurred. You have entered the divisor as zero

Program 5: Using finally keyword for handling an exception.

public class Finallyblock

{

public static void main(String[] a)

{

try

{

int i = 10/0;

}

catch(Exception ex)

{

System.out.println("Inside 1st catch Block");

}

finally

{

System.out.println("Inside 1st finally block");

}

try

{ int i = 10/10;

}

catch(Exception ex)

{

System.out.println("Inside 2nd catch Block");

}

finally

{

System.out.println("Inside 2nd finally block");

} } }

OUTPUT:

C:\Users\Mr.Lopes\Desktop>java Finallyblock

Inside 1st catch Block

Inside 1st finally block

Inside 2nd finally block

Program 6: Write a program to accept and display the month number. ͞Throw͟ a number format exception if improper month number is entered.

import java.io.\*;

class Month

{

public static void main(String args[]) throws IOException

{

int m;

BufferedReader br = new BufferedReader (new InputStreamReader (System.in));

String str;

System.out.println("Enter month number:");

str=br.readLine(); m=Integer.parseInt(str);

try

{

if(m>12 || m<1)

throw new NumberFormatException();

System.out.println("Month number entered is "+m);

}

catch(NumberFormatException ne)

{ System.out.println("Invalid month,Exception Thrown");

}

} }

OUTPUT:

C:\Users\Mr.Lopes\Desktop>java Month

Enter month number:

1

Month number entered is 1

C:\Users\Mr.Lopes\Desktop>java Month

Enter month number:

20

Invalid month,Exception Thrown

Program 7: Throwing user defined exception.

import java.util.Scanner;

public class Client

{

public static void main(String[] args) throws MyException

{

int price ;

Scanner sc=new Scanner(System.in);

System.out.println("Enter your age");

price=sc.nextInt();

if(price < 0)

throw new MyException(price);

else

System.out.println("Your age is :"+price);

} }

class MyException extends Exception

{

private int price;

public MyException(int price)

{

this.price = price;

}

public String toString()

{

return "Price should not be in negative, you are entered" +price;

} }

OUTPUT:

C:\Users\Mr.Lopes\Desktop>javac Client.java

C:\Users\Mr.Lopes\Desktop>java Client

Enter your age

18

Your age is :18

C:\Users\Mr.Lopes\Desktop>java Client

Enter your age

-18

Exception in thread "main" Price should not be in negative, you are entered-18

at Client.main(Client.java:12)

Conclusion: In this experiment we learn about different exception handling methods and about the try catch blocks,throw,throws and finally keywords used for exception handling.