

Title: To work with Exception Handling:

Objectives:

- To be familiar with try, catch, throw block.
- Understand when and where to catch an exception

Using the “if” and “else” loop is not effective when your code has multiple java exceptions to handle.

1. The normal code goes into a **TRY** block.
2. The exception handling code goes into the **CATCH** block

Syntax for using try & catch

```
try{
    statement(s)
}
catch (Exceptiontype name){
    statement(s)
}
```

Syntax of try,catch,finally block:

```
try {
    //code
}
catch (ExceptionType1 e1) {
    // catch block
}
finally {
    // finally block always executes
}
```

Throw block:

- It is used to explicitly throw a single exception.
- When we throw an exception, the flow of the program moves from the try block to the catch block.

Throws:

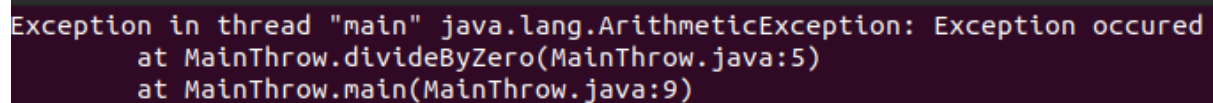
The throws keyword in the method declaration to declare the type of exceptions that might occur within it.

```
accessModifier returnType methodName() throws ExceptionType1, ExceptionType2 ... {  
    // code  
}
```

Program:

```
class MainThrow {  
    public static void divideByZero() {  
  
        // throw an exception  
        throw new ArithmeticException("Exception occurred");  
    }  
  
    public static void main(String[] args) {  
        divideByZero();  
    }  
}
```

Output:



```
Exception in thread "main" java.lang.ArithmeticException: Exception occurred  
    at MainThrow.divideByZero(MainThrow.java:5)  
    at MainThrow.main(MainThrow.java:9)
```

Program 1:

```
public class ExampleOne{  
    public static void main(String args[]){  
        try{  
            int a = 10;  
            int b=a/0;  
            System.out.println(b);  
        }  
    }  
}
```

```

    }
    catch(ArithmeticException e){
        System.out.println("Exception Occurs here");
    }
    finally{
        System.out.println("finally block ");
    }
    System.out.println("Code outside try-catch-finally block");
}
}

```

Output:

```

Exception Occurs here
finally block
Code outside try-catch-finally block
-----

```

Program:

```

public class MainException {
    public static void main(String[] args) {
        //try block containing exception prone code
        try{
            System.out.println("try Block:: Begin");
            int myArray[]=new int[5];
            myArray[6]=10/0;
        }
        //multiple catch blocks
        catch(ArithmeticException e)
        {
            System.out.println("Arithmetic Exception :: Divide by zero!!");
        }
        catch(Exception e)
        {
            System.out.println("Exception :: " + e.getMessage ());
        }
        System.out.println("rest of the code");
    }
}

```

Output:

```
try Block:: Begin
Arithmetic Exception :: Divide by zero!!
rest of the code

-----
(program exited with code: 0)
Press return to continue
```

Program 2:

WAP in java to add the positive numbers taken from the users If the number is not positive ,the program should throw an exception .Use necessary try catch block.

Program 3:

Write a program that creates an Employee class.The class contains two variables as name and salary .Taking the input from the user, throw an exception if the salary is less than 30,000.Make appropriate use of try and catch block.

Program 4:

WAP that creates a calculator class. The class contains two variables of integer type. Design a constructor that accepts two values as parameter and set those values.Make 2 methods one for addition and other for division of two numbers. For addition, two numbers should be positive. So design an exception handler (ArithmeticException) in Add () method. If any negative number is entered then throw an exception in the respective method. .For division,use the exception handler to check if the two numbers are zero. From the main class,perform addition and division through obj1 and obj2.

Program 5:

Write the program to create a class Student which keeps the record of 10 students. Using the concept of multiple try and catch block, construct an exception handler for handling arithmeticException and ArrayOutOfBoundsException.