WAP that desribes, whenever an exception is encounter in the program then our program will be terminated abnormally (halt in the middle)

```java
package com.ravi.exception_demo;

import java.util.Scanner;

public class AbnormalTermination {

    public static void main(String[] args)
    {
        System.out.println("Main Method Started...");

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the value of a :");
        int a = sc.nextInt();

        System.out.print("Enter the value of b :");
        int b = sc.nextInt();

        int result = a/b;
        System.out.println("Result is :"+result);

        System.out.println("Main Method Ended...");
        sc.close();
    }

}
```

Note : In the above program, If we provide the value of b as 0 then Program will generate java.lang.ArithmeticException and It will be terminated in the middle, It is called **Abnormal Termination.**
JVM has default exception handler, which will provide the exception message and terminate the program in the middle.

---

* In order to deal with exception and to work with exception, Java software people has provided the following keyword :

1) try block
2) catch block
3) finally block [try with resourses]
4) throw
5) throws

try block :
----
* try block we cannot write independently, It must be followed by either catch block OR finally block OR both.

```
try
{

}
```
Invalid

```
try
{

}
catch(Exception e)
{
}
```
Valid

```
try
{

}
catch(Exception e)
{
}
finally
{
}
```
Valid

```
try
{

}
finally
{
}
catch(Exception e)
{
}
```
Invalid

* In between try and catch, We cannot write any kind of statement.

try-catch flow :
----
* Whenever we have **error suspecting statement** OR **Risky statement** then we should put that statement inside the try block.

```
try
{
  Risky Code OR Error Suspecting statement
}
catch(Exception e)
{
   System.err.println(e);
}
```

```
                          try
                          {
                            int x = 10;
                            int y = 0;
new ArithmeticException() <= int z = x / y;
                            System.out.println(z); X
                          }
                          catch(Exception e)
                          {
                            System.err.println(e);
                          }
```

Work of try block :
----
1) It will trace the code line by line.
2) If any exception encounter in the try block then with the help of **JVM** try block will **automatically create** the appropriate **Exception object**
3) try block will automatically throw the **execption object** to the nearest catch block.
4) After the execption, the remaining code OR line of try block will not be executed because control will transfer from try block to catch block

try block :

Whenever our statement is error suspecting statement OR Risky statement then we should write that statement inside the try block.

try block must be followed either by catch block or finally block or both.

*try block is responsible to trace our code line by line, if any execption is encountered then with the help of JVM, TRY BLOCK WILL CREATE APPROPRIATE EXECPTION OBJECT, AND THROW THIS EXCEPTION OBJECT to the nearest catch block.

After the execption in the try block, the remaining code of try block will not be executed because control will directly transfer to the catch block.

In between try and catch block we cannot write any kind of statement.

catch block :

The main purpose of catch block to handle the exception which is thrown by try block.

catch block will only executed if there is an exception in the try block.

```java
package com.ravi.exception_demo;

import java.util.Scanner;

public class TryDemo
{
    public static void main(String[] args)
    {
        System.out.println("Main Method Started...");
        Scanner sc = null;
        try
        {
            sc = new Scanner(System.in);

            System.out.print("Enter the value of a :");
            int a = sc.nextInt();

            System.out.print("Enter the value of b :");
            int b = sc.nextInt();

            int result = a/b;
            System.out.println("Result is :"+result);
            System.out.println("End of try Block");
        }

        catch(Exception e)
        {
            System.out.println("Inside Catch Block");
            System.err.println(e);
        }

        System.out.println("Main Method Ended...");
        sc.close();

    }

}
```

In the above program if we put the value of b as 0 but still program will be executed normally because we have used try-catch so it is a
normal termination even we have an exception in the program.

The following program explains that With the help of JVM, try block is creating execption object and throwing the exception object to the nearest catch block.

```java
public class ExceptionDemo
{
    public static void main(String[] args)
    {
        try
        {
            //System.out.println(10/0);
           throw new ArithmeticException("Dividing 10 by 0");
        }
        catch (Exception e)
        {
            System.out.println("Catch Block");
            System.err.println(e);
        }
    }
}
```

The following program explains that wehn we use **throw keyword to throw the exception object explicitly then that corresponding object MUST BE THROWABLE TYPE I.E must be sub class of Throwable**

```java
class Foo extends RuntimeException  //Foo is not a throwable Object
{
    Foo(String error)
    {
        super(error);
    }
}
public class ExceptionDemo
{
    public static void main(String[] args)
    {
        try
        {
            throw new Foo("MyError");
            //System.out.println();  Unrechable
        }
        catch (Exception e)
        {
            System.out.println(e);
        }
    }
}
```

Note : The main purpose of exception handling to provide user-friendly message so client can enjoy the services of software/websites.

Exception handlinag = No Abnormal Termination + User-friendly message on wrong input given by the client.

```java
package com.ravi.exception_demo;

import java.util.Scanner;

public class CustomerDemo
{
    public static void main(String[] args)
    {
        System.out.println("Hello Client, Welcome to my Application..");
        Scanner sc = new Scanner(System.in);

        try
        {
            System.out.print("Please Enter the value of x :");
            int x = sc.nextInt();
            System.out.print("Please Enter the value of y :");
            int y = sc.nextInt();
            int z = x /y;
            System.out.println("Result is :"+z);
        }
        catch(Exception e)
        {
            System.err.println("Please don't put zero Here");
        }

        sc.close();
        System.out.println("Thank you for visiting the Application");

    }

}
```