Exception:

It is an error but it is a run time error which can occured at run time during (logical error) is known as exception.

If an exception occoured at run time programme excution has bee holded (stoped).

To solve the above problem so that programme will excute smoothly even after an Exception occured is known as exception handling.

There are five winds of Process through which exception can be solved.

1. Try
2. Catch
3. Throw
4. Finally
5. Throws

(1) Try: we have to write down the code on block. whin the try block where we assume that exception can occour. try catch will hold the exception & move forword to catch block.

(2) Catch: After an exception has occured try blocked will forword the exception to the catch block & catch will solve the problem & execution the remaning Portion of the programme.

```
try
{
  Statement 1;
        2;
}
Catch (Exception e)
{
}
```

within the parameter of catch block we have to pass any object of exception class. "Exception" is a super class of all the exception class. which can solve any type of exception, occured in our programme.

NOTE:

There can be more than one catch block in ~~expt~~ respect of one try block. but child or sub class (exception class) must be return above the super class (exception class)

Nested try catch:

If a try block is present inside a try block then it is called nested try.

Note: Inners try will excute with the inners block as well as outers block but outers try. But outers try only will excute with the outers catch block.

Nested try catch:

```
try
{
  try
  {
  }
  catch (Exception e)
  {
  }
}
Catch (Exception e)
{
}
```

If an exception occured with in the outside try block then it will move forward the cursers to the outside catch block only to solve the problem. but if an error occured with ith inners catch (try) block. then first inners catch will try to slove the Problem. If the Problem is not slove then it will move forwared to the outers catch block Solve the problem.

```
import Java.util.*;
Class Ex1
{
Public Static void main (String as[])
{
Scanner ob = new Scanner (System.in);
System.out.print ("Enter the 1st NO.=");
int a = ob.nextInt();
System.out.print ("Enter the 2nd NO.=");
int b = ob.nextInt();

try
{
int c = a/b;
}System.out.print ("Division ="+c);

Catch (Exception e)
{
System.out.print ("Division by zero is not Possible");
}
}
}
```

```
import Java.util.*;
{
Public Static void main (String as[])
{
Scanner ob = new Scanner (System.in);
System.out.print ("Enter the 1st NO.=");
int a = ob.nextInt();
System.out.print ("Enter the 2nd NO.=");
int b = ob.nextInt();
try
{
int c = a/b;
System.out.print ("Division ="+c);
}
Catch (ArithmeticException e)
{
System.out.print ("Division by zero is not Possible");
}
Catch (Exception e)
{
System.out.println ("Parent Exception");
}
}
}
```

```
import Java.util.*;
Class Ex3
{
Public Static void main (String as[])
{
int i;
Scanner ob = new Scanner (System.in);
System.out.print ("Enter the Range =");
int n = ob.nextInt();
int a[] = new int[n];

// input
for (i=0; i<n; i++)
{
```

```
System.out.print("Enter the no :-");
a[i] = ob.nextInt();
}
try
{
a[5] = 52;
System.out.println("m" + a[5];
// printing
System.out.print(" " + a[i]);
}
}
Catch (Arithmetic Exception e)
{
System.out.println("division can't be possible");
}
Catch (ArrayIndex Out of Bounds Exception e)
{
System.out.println("Index is out of range");
}
Catch (Exception e)
{
System.out.Println("Parent Class");
}
}
```

throw:

throw is a keyword through which we can forcefully generate an exception. Generally developer can generate an exception by using throw keyword as per their requirement

throw new name of exception ();

finally:

finally is a block which is used when you want to execute a specific line which we want to display even after an exception is occured. binary block will be written after the catch block. that means at the end of the block finally block. have to be written.