**Exception:**

An Exception is an **unwanted or unexpected condition** which **disturbs our normal flow** of execution.

(OR)

An unexpected statement which stops the execution of a program.

* **Once Exception occurred** remaining part of **program will not be executed**. So, it is our responsibility to handle the exception.
* Exception handling doesn't mean, we are resolving an exception it is just like providing an alternate solution so that even though exception happens our program should work properly
* **Throwable class** is a super class to "Exception" class and "Error" class.
* Exception class is a **super class** to RuntimeException class and other Exception classes
* All the Exception classes belongs to java.lang package.

Depending on Hierarchy, Exceptions are divided into 2 types

1. Checked Exception (Compile time Exceptions)
2. Unchecked Exception (Run time Exceptions)

**1. Checked Exception (Compile time Exceptions):**

Exception which are checked (identified or found out) during compile time by compiler, such type of exception is called as “Checked Exceptions”.

(OR)

Exception classes which are directly inheriting Exception class except RuntimeException class is called as checked exception.

* Checked Exceptions are also called as **Compile time Exception**

Examples (Classes) of Checked Exceptions are:

1. InterruptedException
2. ClassNotFoundException
3. SQLException
4. FileNotFoundException

**2. Unchecked Exception (Run time Exceptions):**

Exception which are checked (identified or found out) during Runtime or execution time, such type of exception is called as “Unchecked Exceptions”.

In case of Unchecked Exception our program will **at least compiles successfully.**

* Unchecked Exceptions are also called as **Runtime Exceptions**
* **RuntimeExceptionclass** is a super class to all UncheckedException classes

Examples (Classes) of Unchecked Exceptions are:

1. ArithmeticException
2. ArrayIndexOutOfBoundsException
3. NullPointerException
4. StringIndexOutofBoundsException
5. ClassCastException
6. NumberFormatException

Once exception occurred remaining part of a program will not be executed and ending up with abnormal termination. So, if we want to execute and have normal termination then, we have to handle the Exception.

**Exception can be handled** by using **try and catch block**

**try block** is used to **keep a code which causes an exception**. Once Exception occurred in try block remaining part of code from try block will not be executed. So, we should keep only statements which causes exception under try block.

Once exception occurred in try block, JVM immediately make a search for **corresponding catch block.**

**Catch block** is where we will **catch the exception.** Under catch block we provide some statements which works as alternate solution for Exception.

**Ex:**

Try

{

//risky statements which causes exception

}

catch (Exception name ref var)

{

//Alternate solution

}

**finally block:**

finally is a block which will get executed irrespective of

1. exception occurred or not

2.exception occurred and handled

3.exception occurred and not handled

Basically, "finally" is used to keep an important code which should not be skipped at any condition like closing of data base connection or closing of opened file etc.

----------------------------------- \*\*\*\*\* ------------------------------------------------------------------------

**Multithreading:**

Programs which contain multiple threads is called as multi-threaded program and such process is called as Multithreading.

**Thread:** It is a flow of execution (OR) it is a small part of an application (OR) it is a light weight process

**Creation of Thread:**

A thread can be created in 2 ways

1. By extending Thread class
2. By implementing Runnable Interface

**Note:**

In every program always there is one default thread i.e., main thread (main ())

**Creating Thread by Extending Thread class:**

* Create our class which **extends** Thread class
* For defining thread, we have to override run () from thread class

public void run ();

* it is a predefine method of thread class
* Make a call to run () by using start ()

public synchronized void start ();

**Note:**

Synchronized is a keyword which indicates that only thread can access method at a time.

Thread is a Predefine class, belongs to java.lang package

**Creating a thread by implementing runnable interface:**

* Create a class and that class should implements Runnable interface
* Override run ()
* Create object of thread class & connect to your class object
* Make a call to start ()

----------------------------------- \*\*\*\*\* ------------------------------------------------------------------------

**Program For Prime Number:**

**public** **class** PrimeNumber

{

**public** **static** **void** main(String[] args)

{

**int** num=7,count=0;

**for**(**int** i=1;i<=num;i++)

{

**if**(num%i == 0)

{

count++; //count=count+1

}

}

**if**(count==2)

{

System.***out***.println(num+" Is a Prime Number");

}

**else**

{

System.***out***.println(num+" Is Not a Prime Number");

}

}

}

**Output:** 7 Is a Prime Number

----------------------------------- \*\*\*\*\* ------------------------------------------------------------------------

**Program For Prime Number in 1 to 100:**

**public** **class** PrimeNumber1to100

{

**public** **static** **void** main(String[] args)

{

**for**(**int** num=1;num<=100;num++)

{

**int** count=0;

**for**(**int** i=1;i<=num;i++)

{

**if**(num%i == 0)

{

count++; //count=count+1

}

}

**if**(count == 2)

{

System.***out***.print(num+" ");

}

}

}

}

**Output:** 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97

----------------------------------- \*\*\*\*\* ------------------------------------------------------------------------