Syllabus
========
1.Introduction.
2. The ways to define, instantiate and start a new Thread.
    1. By extending Thread class
    2. By implementing Runnable interface
3. Thread class constructors
4. Thread priority
5. Getting and setting name of a Thread.
6. The methods to prevent(stop) Thread execution.
      1. yield()
      2. join()
      3. sleep()
7. Synchronization.
8. Inter Thread communication.
9. Deadlock
10. Daemon Threads.
11. Various Conclusion
      1. To stop a Thread
      2. Suspend & resume of a thread
      3. Thread group
      4. Green Thread
      5. Thread Local
12. Life cycle of a Thread

MulitTasking
============
  Executing several task simultaneously is the concept of multitasking.
  There are 2 types of Multitasking.
a. Process based multitasking
b. Thread based multitasking.

Process based multitasking
==========================
Executing several tasks simultaneously where each task is a seperate independent process such type of multitasking is called
"process based multitasking".
   eg:: typing a java pgm
           listening to a song
           downloading the file from internet

Process based multitasking is best suited at "os level".

Thread based multitasking
=============================
=>Executing several tasks simulatenously where each task is a seperate independent part of the same Program, is called
    "Thread based MultiTasking".
    Each independent part is called "Thread".

1. This type of multitasking is best suited at "Programatic level".
    The main advantages of multitasking is to reduce the response time of the system and to improve the performance.

2. The main important application areas of multithreading are
     a. To implement mulitmedia graphics
     b. To develop web application servers(will learn in JEE)
     c. To develop video games

d. To develop animations

3. Java provides inbuilt support to work with threads through API called Thread,Runnable,ThreadGroup,ThreadLocal,...

4. To work with multithreading, java developers will code only for 10% remaining 90% java API will take care..

What is thread?
A. Seperate flow of execution is called "Thread".
if there is only one flow then it is called "SingleThread" programming.
For every thread there would be a seperate job.

B. In java we can define a thread in 2 ways
a. Implementing Runnable interface
b. extending Thread class

1. Extending Thread class
=> we can create a Thread by extending a Thread.

```
class MyThread extends Thread{
      @Override
      public void run(){
            for(int i=0;i<10;i++)
                  System.out.println("child thread");
      }
}
```

defining a thread(writing a class and extending a Thread)
job a thread(code written inside run())

```
class ThreadDemo{
      public static void main(String... args){
            MyThread t =new MyThread();//Thread instantiation
            t.start();//starting a thread

            ;;;; // At this line 2 threads are there

            for(int i=1;i<=5;i++)
                  System.out.println("Main Thread");
      }
}
```

Behind the scenes
1. Main thread is created automatically by JVM.
2. Main thread creates child thread and starts the child thread.

ThreadScheduler
==========
If multiple threads are waiting to execute,then which thread will execute 1st is decided by ThreadScheduler which is part of
JVM.
In case of MultiThreading we can't predict the exact output only possible output we can expect.
Since jobs of threads are important,we are not interested in the order of execution it should just execute such that
performance should be improved.

case2: diff b/w t.start() and t.run()
if we call t.start() and seperate thread will be created which is responsible to execute run() method.

if we call t.run(), no seperate thread will be created rather the method will be called just like normal method by main
thread.
if we replace t.start() with t.run() then the output of the program would be

```
child thread
child thread
child thread
child thread
child thread
child thread
child thread
child thread
child thread
child thread
main thread
main thread
main thread
main thread
main thread
main thread
main thread
main thread
main thread
main thread
```

case3:: Importance of Thread class start() method
For every thread, required mandatory activities like registering the thread with Threadscheduler will be taken care by
Thread class start() method and programmer is responsible of just doing the job of the Thread inside run() method.
start() acts like an assistance to programmer.

```
public void start()
{
   register thread with ThreadScheduler
     All other mandatory low level activities
   invoke or calling run() method.
}
```

We can conclude that without executing Thread class start() method there is no chance of starting a new Thread in java.
Due to this start() is considered as heart of MultiThreading.

case4:: If we are not overriding run() method
If we are not Overriding run() method then Thread class run() method will be executed which has empty implementation and
hence we wont get any output.

eg::

```
class MyThread extends Thread{}
class ThreadDemo{
      public static void main(String... args){
            MyThread t=new MyThread();
            t.start();
      }
}
```

It is highly recomended to override run() method, otherwise don't go for MultiThreading concept.