What is thread?

A. Separate 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

ThreadScheduler

If multiple threads are waiting to execute,then which thread will execute 1st is decided by ThreadScheduler which is part of JVM.
In the 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.

diff b/w t.start() and t.run()

if we call t.start() and seperate thread will be created which is responsible to execute the run() method.
if we call t.run(), no separate thread will be created, rather the method will be called just like a normal method by main thread.

Importance of Thread class start() method

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

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

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

If we are not overriding run() method

If we are not Overriding run() method then Thread class run() method will be executed which has an empty implementation and hence we won't get any output.

eg::

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