MultiThreading
 =>Utilizing the CPU time

How to create threads in java?
1. Runnable(I)
2. Thread(C)

Which method resposible for register the Thread with Threadscheduler,performing low level activites,invoke run()?
Ans. start() of Thread class, we call it as "Heart of MultiThreading".

Give a syntax of how a Thread can be created using Runnable and Thread class?
a. Using Thread class
```
      eg: MyThread t = new MyThread();
            t.start();
```

b. using Runnable interface
```
      eg: MyRunnable r =new MyRunnable();
          Thread t= new Thread(r);
            t.start();
```

Which approach is best to create threads in java?
Ans. Runnable(I)[To get the benifit of inheritance]

When we create multiple threads, who would decide which thread to execute?
Ans. ThreadScheduler[Alogrithm of JVM].

Can we set priority of a Thread and what is the priority range of the Thread?
Ans. yes[using setPriority(int priority) getPriority():: int]
```
      MIN_PRIORITY    => 1
        MAX_PRIORITY    => 10
      NORMAL_PRIORITY => 5
```

Can we give name for a Thread?
Ans. yes using methods like [setName(String name):: void , getName():: String]

What are the methods avaialble to prevent the execution of Thread?
Ans.
     yield() :: pause the current execution thread and give chance for other thread of the same priority.
     join() :: wait for the other thread to finish the execution.
     sleep() :: If thread doesn't want to perform any operation for a particular period of time then we use sleep().

What is the usage of interrupt() call?
Ans. interrupt() call is used to interrupt the thread, when the thread enters into "Sleeping/Waiting" state.

Which all methods would throw InterruptedException?
Ans. join(), sleep() call would throw InterruptedException, which is fullycheckedException.

What is synchronization and its advantage?
Ans. If multiple threads are operating on a particular resource, then due to multithreading effect the output would
     be "ir-regular", to avoid this problem we use "synchronized" accessmodifier on methods and block.
     Internally synchronization would work with the help of
      a. class level lock ==> it works with "static-synchronized methods".

b. object level lock ==> it works with "instance-synchronized methods".

Which approach is good in multithreading, writing synchronized methods or synchronized blocks?
Ans. synchronized block is good.

How would we achieve InterThread communication in java?
Ans. notifyAll(),notify(),wait().InterThread Communication can happen only in "synchronized" environment, if not
it would result in an Exception called "IllegalMonitorStateException".

wait() ::The thread which needs updation will call wait() on the required object and it enters into waiting state
by releasing the lock immediately.

notify()::The thread which is performing updation on the object will call notify() and it may or maynot release the
lock immediately.Once the notify() is called, the thread which is in waiting state will
get the notification.

InterThreadCommunication(remember postbox example)
==========
Two threads can communicate each other with the help of
a. notify()
b. notifyAll()
c. wait()

notify()=> Thread which is performing updation should call notify(),so the waiting thread will
get notification so it will continue with its execution with the updated items.
wait() => Thread which is expecting notification/updation should call wait(), immediately the
Thread will enter into waiting state.

wait(),notifyAll(),notify() is present in Object class, but not in Thread class why?
=> Thread will call wait(),notify(),notifyall() on any type of objects like Student,Customer,Engineer.

If a thread wants to call wait(),notify()/notifyall() then compulsorily the thread should be
the owner of the object otherwise it would result in "IllegalMonitorStateException".
We say thread to be owner of that object if thread has lock of that object.
It means these methods are part of synchronized block or synchronized method, if we try to use
outside synchronized area then it would result in RunTimeException called "IllegalMonitorStateException".

if a thread calls wait() on any object, then first it immediately releases the lock on that object and it enters into waiting state.
if a thread calls notify() on any object,then he may or may not release the lock on that object immediately.
Except wait(),notify(),notifyAll() lock can't be relased by other methods.

Note::

yield(),sleep(),join() => can't release the lock.
wait(),notify(),notifyAll() => will release the lock,otherwise interthread communication can't happen.

Once a Thread calls wait(), notify(), notifyAll() methods on any object then it releases the lock of that particular object but
not all locks it has.

Method prototype of wait(),notify(),notifyAll()
1. public final void wait()throws InterruptedException
2. public final native void wait(long ms)throws InterruptedException
3. public final void wait(long ms,int ns)throws InterruptedException
4. public final native void notify()
5. public final void notifyAll()

Program
=======
eg#1.

```
class ThreadB extends Thread{
      int total =0;

      @Override
      public void run(){
            for (int i=0;i<=100 ; i++){
                  total+=i;
            }
      }
}
public class Test {
      public static void main(String[] args)throws InterruptedException {
            ThreadB b=new ThreadB();
            b.start();

            stmt-1;
            System.out.println(b.total);
      }
}
```

A. stmt-1
      if i replace with Thread.sleep(10000) then thread will enter into waiting statement
        but within 1ns only the updation value is ready.
        with in 10 sec if the updation is not ready, then we should not use Thread.sleep(10000)

B. stmt-2
       if i replace with b.join(), then main thread will enter into waiting state,then child will
       execute for loop,till then main thread has to wait.
       main thread is waiting for updation result.

```
            for (int i=0;i<=100 ; i++){
                  total+=i;
          }
            //1cr lines of code is available
```

        main thread has to wait till 1 cr lines of code,y main thread should wait for the
        complete code to finish.

eg#2.

```
class ThreadB extends Thread{
	int total =0;

  @Override
  public void run(){

	synchronized(this){
		System.out.println("Child thread started calculation");
		for (int i=0;i<=100 ; i++){
			total+=i;
		}
		System.out.println("Child thread trying to give notification");
		this.notify();
	}
  }
}
public class Test {
	public static void main(String[] args)throws InterruptedException {
		ThreadB b=new ThreadB();
		b.start();

	synchronized(b){
		System.out.println("Main thread is calling wait on B object");
		b.wait(10000);//10sec
		System.out.println("Main thread got notification");
		System.out.println(b.total);
	}

  }
}
```

Output

```
======
Child thread started calculation
Child thread trying to give notification
Main thread is calling wait on B object for 10sec
Main thread got notification
5050
```

eg#3.

```
class ThreadB extends Thread{
	int total =0;

  @Override
  public void run(){

	synchronized(this){
		System.out.println("Child thread started calculation");
		for (int i=0;i<=100 ; i++){
			total+=i;
		}
		System.out.println("Child thread trying to give notification");
		this.notify();
	}
  }
}
public class Test {
```

```
      public static void main(String[] args)throws InterruptedException {
            ThreadB b=new ThreadB();
            b.start();

            Thread.sleep(10000);//10sec

      synchronized(b){
            System.out.println("Main thread is calling wait on B object");
            b.wait();
            System.out.println("Main thread got notification");
            System.out.println(b.total);
      }

   }
}
Output
======
Child thread started calculation
Child thread trying to give notification
Main thread is calling wait on B object
      becoz of Thread.sleep(10000) main thread will never get notification.
```