**Section 1: Define**

Pg. Chapter 12, pg. 435 - Java Programming *A comprehensive Introduction*

Thread-

A *thread* is a thread of execution in a program. The Java Virtual Machine allows an application to have multiple threads of execution running concurrently.

Every thread has a priority. Threads with higher priority are executed in preference to threads with lower priority. Each thread may or may not also be marked as a daemon. When code running in some thread creates a new Thread object, the new thread has its priority initially set equal to the priority of the creating thread, and is a daemon thread if and only if the creating thread is a daemon.

When a Java Virtual Machine starts up, there is usually a single non-daemon thread (which typically calls the method named main of some designated class). The Java Virtual Machine continues to execute threads until either of the following occurs:

* The exit method of class Runtime has been called and the security manager has permitted the exit operation to take place.
* All threads that are not daemon threads have died, either by returning from the call to the run method or by throwing an exception that propagates beyond the run method.

There are two ways to create a new thread of execution. One is to declare a class to be a subclass of Thread. This subclass should override the runmethod of class Thread. An instance of the subclass can then be allocated and started. For example, a thread that computes primes larger than a stated value could be written as follows:

class PrimeThread extends Thread {

long minPrime;

PrimeThread(long minPrime) {

this.minPrime = minPrime;

}

public void run() {

// compute primes larger than minPrime

 . . .

}

}

The following code would then create a thread and start it running:

PrimeThread p = new PrimeThread(143);

p.start();

The other way to create a thread is to declare a class that implements the Runnable interface. That class then implements the run method. An instance of the class can then be allocated, passed as an argument when creating Thread, and started. The same example in this other style looks like the following:

class PrimeRun implements Runnable {

long minPrime;

PrimeRun(long minPrime) {

this.minPrime = minPrime;

}

public void run() {

// compute primes larger than minPrime

 . . .

}

}

The following code would then create a thread and start it running:

PrimeRun p = new PrimeRun(143);

new Thread(p).start();

Every thread has a name for identification purposes. More than one thread may have the same name. If a name is not specified when a thread is created, a new name is generated for it.

Unless otherwise noted, passing a null argument to a constructor or method in this class will cause a [NullPointerException](http://docs.oracle.com/javase/7/docs/api/java/lang/NullPointerException.html" \o "class in java.lang) to be thrown.

start()-

Causes this thread to begin execution; the Java Virtual Machine calls the run method of this thread.

The result is that two threads are running concurrently: the current thread (which returns from the call to the start method) and the other thread (which executes its run method).

It is never legal to start a thread more than once. In particular, a thread may not be restarted once it has completed execution.

run()-

If this thread was constructed using a separate Runnable run object, then that Runnable object's run method is called; otherwise, this method does nothing and returns.

Subclasses of Thread should override this method.

sleep()-

Causes the currently executing thread to sleep (temporarily cease execution) for the specified number of milliseconds plus the specified number of nanoseconds, subject to the precision and accuracy of system timers and schedulers. The thread does not lose ownership of any monitors.

**Parameters:**

millis - the length of time to sleep in milliseconds

nanos - 0-999999 additional nanoseconds to sleep

Task 1:

**USE OBJECT ORIENTATED PROGRAM DESIGN TO SOLVE PROBLEM**

Create basic thread example that executes multiple threads.

The thread example should extend **Thread()** class.

then the program should have the same output

**implement** Runnable

.

