AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

CHAPTER 12
CONCURRENCY

ThreadLocal
sleep() and wait()
fail-safe synchronization
blocking queue
Runnable
CompletionService ConcurrentHashMap
Executor Framework
ConcurrentModificationException
Callable Immutability Atomic classes
HashMap

Concurrency

```
@Override
public Thread newThread
  thread = new Thread();
  return thread;
}
```

SUBJECT:OOP-I CODE:3140705	PREPARED BY: ASST.PROF.NENSI KANSAGARA (CSE DEPARTMENT,ACET)	AMIRAJ COLLEGE OF ENGINEERING & TECHNOLOGY

INTRODUCTION TO THREAD:

A thread is a:

- ➔ Facility to allow multiple activities within a single process
- ➔ Referred as lightweight process
- ➔ A thread is a series of executed statements
- ➔ Each thread has its own program counter, stack and local variables
- ➔ A thread is a nested sequence of method calls
- ➔ Its shares memory, files and per-process state

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

What’s the need of a thread or why we use Threads?

- To perform asynchronous or background processing
- Increases the responsiveness of GUI applications
- Take advantage of multiprocessor systems
- Simplify program logic when there are multiple independent entities

What happens when a thread is invoked?

When a thread is invoked, there will be two paths of execution. One path will execute the thread and the other path will follow the statement after the thread invocation. There will be a separate stack and memory space for each thread.

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

THREAD VS PROCESS

- ➔ Thread is a light weight process
- ➔ Threads do not require separate address space for its exceution.it runs the address space of the process to which it belongs to

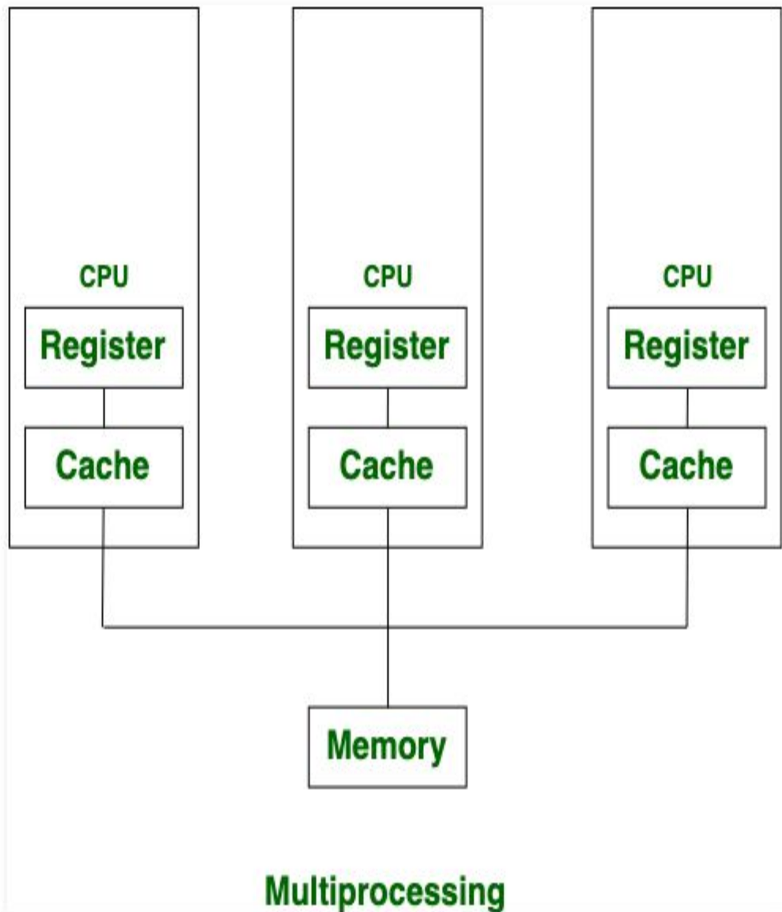
- ➔ Process is heavy weighted process
- ➔ Each process requires separate address space to execute

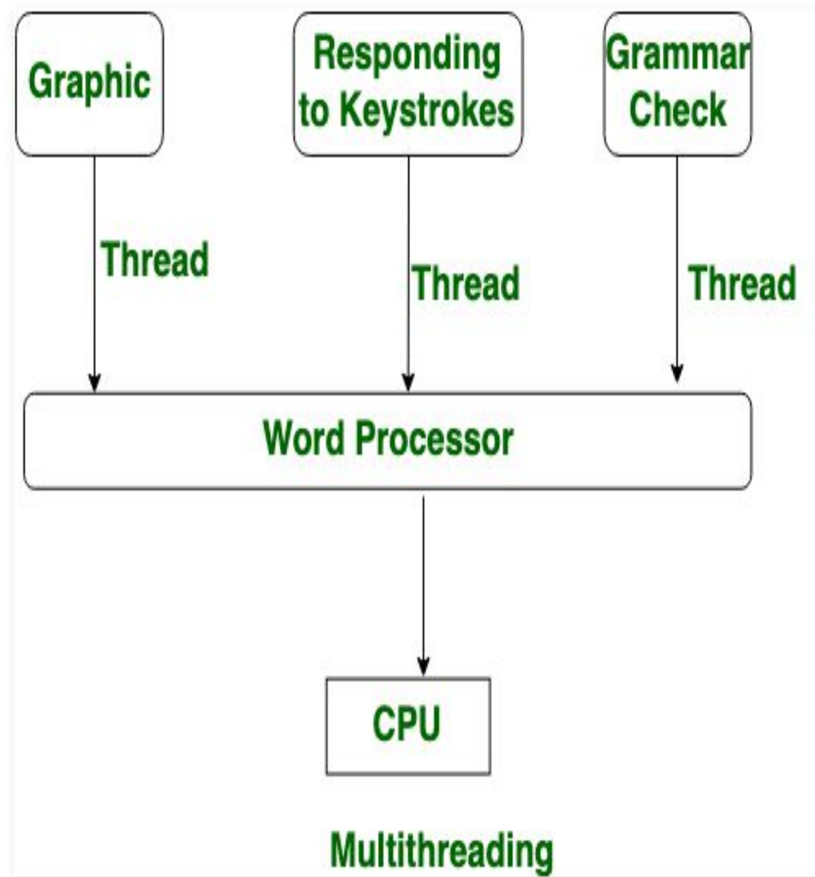
AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

DIFFERENCE BETWEEN MULTITHREADING AND MULTITASKING

BASIS FOR COMPARISON	MULTIPROCESSING	MULTITHREADING
Basic	Multiprocessing adds CPUs to increase computing power.	Multithreading creates multiple threads of a single process to increase computing power.
Execution	Multiple processes are executed concurrently.	Multiple threads of a single process are executed concurrently.
Creation	Creation of a process is time-consuming and resource intensive.	Creation of a thread is economical in both sense time and resource.
Classification	Multiprocessing can be symmetric or asymmetric.	Multithreading is not classified.

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

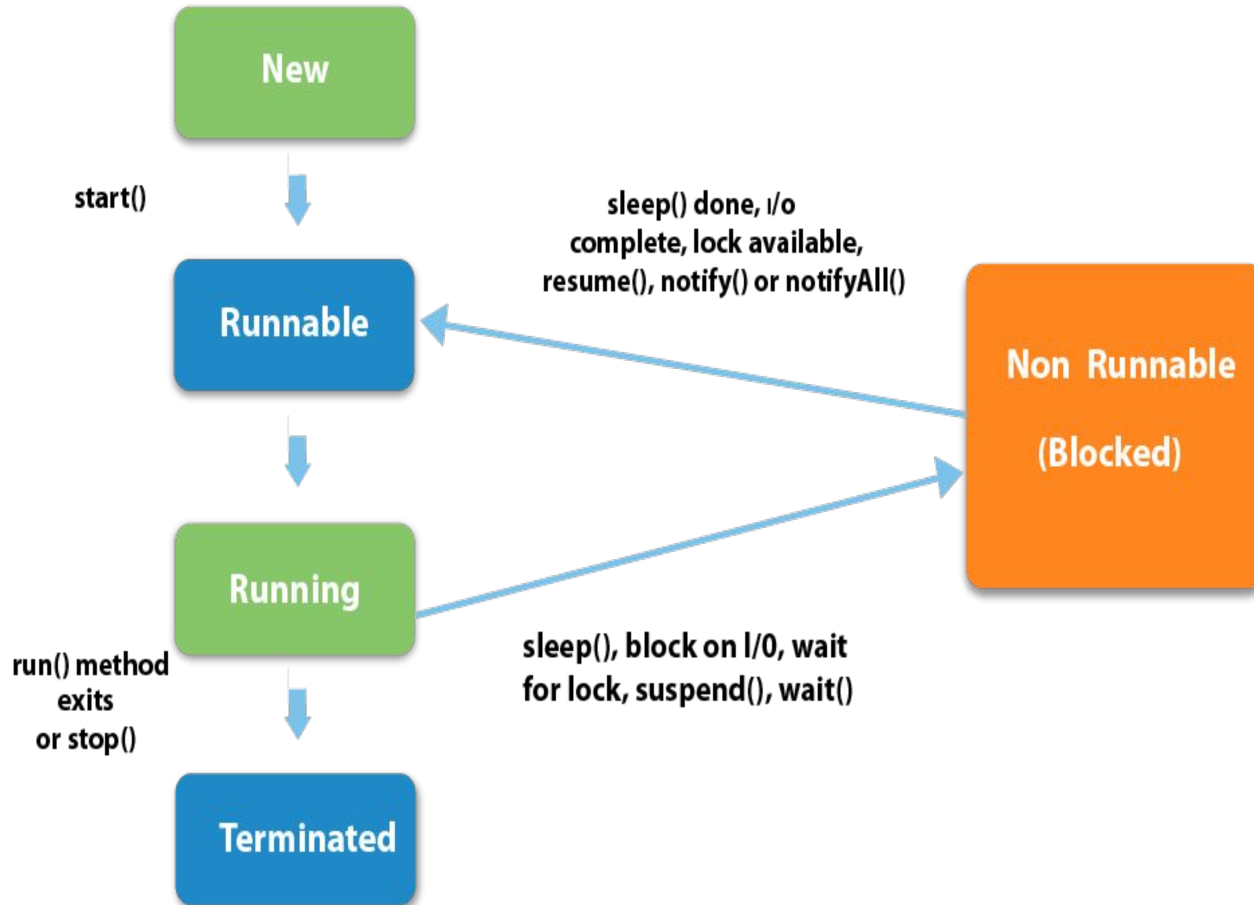

CPU
Register
Cache
CPU
Register
Cache
CPU
Register
Cache
Memory
Multiprocessing


Graphic
Responding to Keystrokes
Grammar Check
Thread
Thread
Thread
Word Processor
CPU
Multithreading

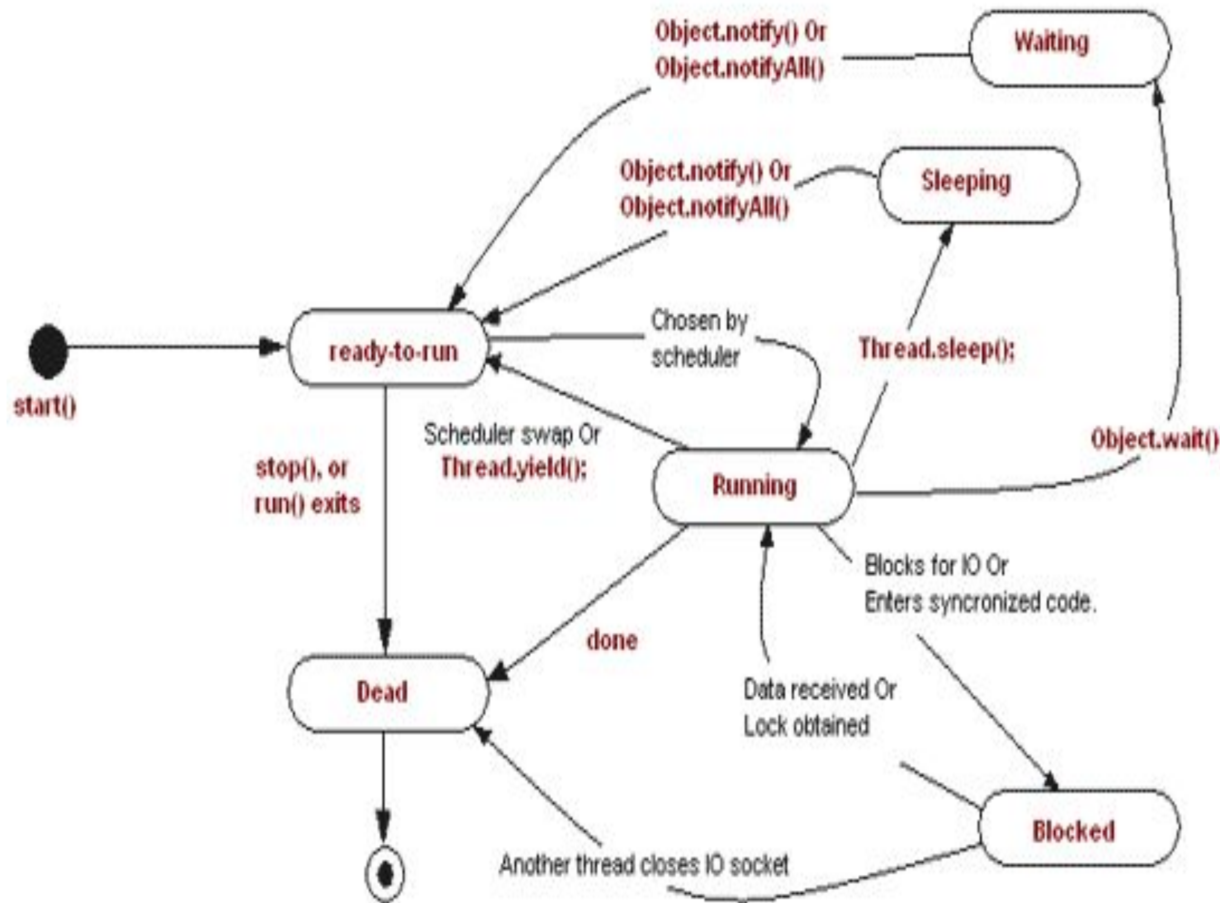
THREAD STATES AND LIFE CYCLE

- The start method creates the system resources, necessary to run the thread, schedules the thread to run, and calls the thread's run method.
- A thread becomes "Not Runnable" when one of these events occurs:
  - If sleep method is invoked.
  - The thread calls the wait method.
  - The thread is blocking on I/O.
- A thread dies naturally when the run method exits.

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY



AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY



AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

1) New

The thread is in new state if you create an instance of Thread class but before the invocation of start() method.

2) Runnable

The thread is in runnable state after invocation of start() method, but the thread scheduler has not selected it to be the running thread.

3) Running

The thread is in running state if the thread scheduler has selected it.

4) Non-Runnable (Blocked)

This is the state when the thread is still alive, but is currently not eligible to run.

5) Terminated

A thread is in terminated or dead state when its run() method exits.

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

CREATION OF THREAD

There are two ways to create a thread:

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

```
Public void run()
{
//statements to implements thread
}
```

AMIRAJ COLLEGE OF ENGINEERING & TECHNOLOGY

EXTENDING THREAD CLASS

- The class should extend Java Thread class.
- The class should override the run() method.
- The functionality that is expected by the Thread to be executed is written in the run() method.

void start(): Creates a new thread and makes it runnable.

void run(): The new thread begins its life inside this method.

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

CONSTRUCTORS USED IN THREAD CLASS

- Thread()
- Thread(String name)
- Thread(Runnable r)
- Thread(Runnable r,String name)

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

METHOD NAMES:

1. **public void run():** is used to perform action for a thread.
2. **public void start():** starts the execution of the thread.JVM calls the run() method on the thread.
3. **public void sleep(long miliseconds):** Causes the currently executing thread to sleep (temporarily cease execution) for the specified number of milliseconds.
4. **public void join():** waits for a thread to die.
5. **public void join(long miliseconds):** waits for a thread to die for the specified miliseconds.
6. **public int getPriority():** returns the priority of the thread.
7. **public int setPriority(int priority):** changes the priority of the thread.
8. **public String getName():** returns the name of the thread.
9. **public void setName(String name):** changes the name of the thread.

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

```java
public class MyThread extends Thread {
   public void run(){
    System.out.println("thread is running...");
  }
   public static void main(String[] args) {
     MyThread obj = new MyThread();
     obj.start();
}
```

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

IMPLEMENTING RUNNABLE INTERFACE

- The class should implement the Runnable interface
- The class should implement the run() method in the Runnable interface
- The functionality that is expected by the Thread to be executed is put in the run() method

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

```
public class MyThread implements Runnable {
   public void run(){
     System.out.println("thread is running..");
   }
   public static void main(String[] args) {
     Thread t = new Thread(new MyThread());
     t.start();
}
```

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

CREATING MULTIPLE THREADS

- The multiple threads can be created both by extending thread class and by implementing the runnable interface.

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

```java
package javaapplicationthreadprog;
class A extends Thread
{
    public void run()
    {
        for(int i=0;i<5;i++)
        {
            System.out.println(i);
        }
    }
}
class B extends Thread
{
    public void run()
    {
        for(int i=10;i>=5;i--)
        {
            System.out.println(i);
        }
    }
}

public class JavaApplicationThreadProg {

    public static void main(String[] args) {
       A t1 = new A();
       B t2 = new B();
       t1.start();
       t2.start();
```

Output - JavaApplicationThreadProg (run) X

```
run:
0
1
2
3
4
10
9
8
7
6
5
BUILD SUCCESSFUL (total time: 9 seconds)
```

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

```java
package javaapplicationthreadprog;
class A implements Runnable
{
    public void run()
    {
        for(int i=0;i<5;i++)
        {
            System.out.println(i);
        }
    }
}
class B implements Runnable
{
    public void run()
    {
        for(int i=10;i>=5;i--)
        {
            System.out.println(i);
        }
    }
}
public class JavaApplicationThreadProg {

    public static void main(String[] args) {
       A obj1 = new A();
       B obj2 = new B();
       Thread t1 = new Thread(obj1);
       Thread t2 = new Thread(obj2);
       t1.start();
       t2.start();
    }
```

Output - JavaApplicationThreadProg (run) X

```
run:
0
1
2
3
4
10
9
8
7
6
5
BUILD SUCCESSFUL (total time: 9 seconds)
```

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

CREATING AND EXECUTING THREADS WITH THE EXECUTOR FRAMEWORK

- **Java thread pool** manages the pool of worker threads. It contains a queue that keeps tasks waiting to get executed. We can use ThreadPoolExecutor to create thread pool in Java.
- Java thread pool manages the collection of Runnable threads. The worker threads execute Runnable threads from the queue. **java.util.concurrent.Executors** provide factory and support methods for **java.util.concurrent.Executor** interface to create the thread pool in java.
- Executors is a utility class that also provides useful methods to work with ExecutorService, ScheduledExecutorService, ThreadFactory, and Callable classes through various factory methods.

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

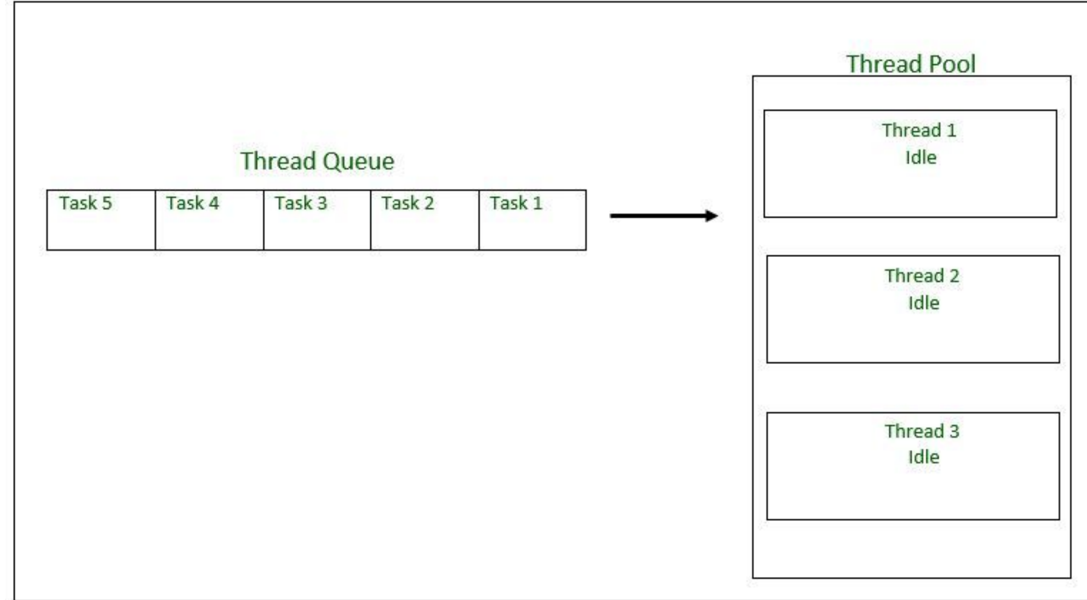
Steps to be followed

1. Create a task(Runnable Object) to execute

2. Create Executor Pool using Executors

3. Pass tasks to Executor Pool

4. Shutdown the Executor Pool



AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

```
package javaapplicationexceutordemo;

import java.util.concurrent.ThreadPoolExecutor;
import java.util.logging.Level;
import java.util.logging.Logger;

class Task implements Runnable
{
    String Name;
    public Task(String Name)
    {
        this.Name=Name;
    }
    public String getName()
    {
        return Name;
    }
    public void run()
    {
        try{
        System.out.println("Executing "+Name);
            try {
                Thread.sleep(10000);
            } catch (InterruptedException ex) {
                Logger.getLogger(Task.class.getName()).log(Level.SEVERE, null, ex);
            }
    }

    }
    }
```

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

```java
public class JavaApplicationExceutorDemo {

    public static void main(String[] args) {
        ThreadPoolExecutor executor = (ThreadPoolExecutor)Exceutors.newFixedThreadPool(2);
        for(int i=1;i<=5;i++)
        {
            Task task = new Task("Task#"+i);
            System.out.println("Created:"+task.getName());
            executor.execute(task);
        }
        executor.shutdown();
    }

    private static class Exceutors {

        public Exceutors() {
        }
    }
```

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

THREAD SYNCHRONIZATION

- ➔ In many cases concurrently running threads share data and two threads try to do operations on the same variables at the same time. This often results in corrupt data as two threads try to operate on the same data.
- ➔ A popular solution is to provide some kind of lock primitive. Only one thread can acquire a particular lock at any particular time. This can be achieved by using a keyword “synchronized” .
- ➔ By using the synchronize only one thread can access the method at a time and a second call will be blocked until the first call returns or wait() is called inside the synchronized method.

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY

Using Synchronized Method

```
package javaapplicationmysynthread;
class Test
{
    synchronized void display(int num)
    {
        System.out.println("\nTable for"+num);
        for(int i=1;i<=10;i++)
        {
            System.out.println("\nend of table");
        }
        try
        {Thread.sleep(1000);}
        catch(Exception e)
        {}
    }
}
class A extends Thread
{
    Test th1;
    A(Test t)
    {
        th1=t;
    }
    public void run()
    {th1.display(2);}
}
```

```
class B extends Thread
{
    Test th2;
    B(Test t)
    {
        th2=t;
    }
    public void run()
    {th2.display(100);}
}
public class JavaApplicationMySynThread {

    public static void main(String[] args) {
        Test obj = new Test();
        A t1 = new A(obj);
        B t2 = new B(obj);
        t1.start();
        t2.start();
    }

}
```

Using Synchronized Block

```
package javaapplicationmysynthread;
class Test
{
    void display(int num)
    {
    synchronized(this)
    {
        System.out.println("\nTable for"+num);
        for(int i=1;i<=10;i++)
        {
            System.out.println(""+num*i);
        }
        System.out.println("\nEnd of table");
        try
        {Thread.sleep(1000);}
        catch(Exception e)
        {}
    }
    }
}
class A extends Thread
{
    Test th1;
    A(Test t)
    {
        th1=t;
    }
    public void run()
    {th1.display(2);}
}
```

```
class B extends Thread
{
    Test th2;
    B(Test t)
    {
        th2=t;
    }
    public void run()
    {th2.display(100);}
}
public class JavaApplicationMySynThread {

    public static void main(String[] args) {
        Test obj = new Test();
        A t1 = new A(obj);
        B t2 = new B(obj);
        t1.start();
        t2.start();
    }

}
```

Thank you!

AMIRAJ
COLLEGE OF ENGINEERING & TECHNOLOGY