```java
public class RunExp1 implements Runnable
{
public void run()
{
System.out.println("Thread is running...");
}
public static void main(String args[])jkl;
{
RunExp1 r1=new RunExp1();
Thread t1 =new Thread(r1);
// this will call run() method
t1.start();
}
}
/*Call run() method using start() method*/

public class RunExp2 extends Thread
{
public void run()
{
System.out.println("running...");
}
public static void main(String args[])
{
RunExp2 t1=new RunExp2 ();
// It does not start a separate call stack
t1.run();
}
}
/*Calling the run() method using run() itself*/

public class RunExp3 extends Thread
{
public void run()
{
for(int i=1;i<6;i++)
{
try
{
Thread.sleep(500);
}catch(InterruptedException e){System.out.println(e);}
System.out.println(i);
}
}
public static void main(String args[])
{
RunExp3 t1=new RunExp3();
RunExp3 t2=new RunExp3();
t1.run();
t2.run();
}
}
/*O/P 1 2 3 4 5 1 2 3 4 5*/
/*Call the run() method more than one time*/

class MultithreadingDemo extends Thread{
    public void run(){
    System.out.println("My thread is in running state.");
    }
    public static void main(String args[]){
        MultithreadingDemo obj=new MultithreadingDemo();
        obj.start();
    }
}
/*Java thread example by extending THREAD class*/

class MultiThreadRun implements Runnable{
    public void run(){
        System.out.println("My thread is in running state.");
    }
    public static void main(String args[]){
        MultiThreadRun obj=new MultiThreadRun();
        Thread tobj =new Thread(obj);
        tobj.start();
    }
```

```
}
/*Java Thread Example by implementing Runnable interface*/

class ThreadA extends Thread{
    public void run(){
        for(int i=0;i<=5;i++){
            System.out.println("Thread i"+ -1*i);
        }
        System.out.println("Thread A");
    }
}
class ThreadB extends Thread{
    public void run(){
        for(int j=0;j<=5;j++){
            System.out.println("Thread i"+ -2*j);
        }
        System.out.println("Thread B");
    }
}
class ThreadC extends Thread{
    public void run(){
        for(int k=0;k<=5;k++){
            System.out.println("Thread i"+ -3*k);
        }
        System.out.println("Thread C");
    }
}
class MultiThreadClass{
    public static void main(String args[]){
        ThreadA a=new ThreadA();
        ThreadB b=new ThreadB();
        ThreadC c=new ThreadC();
        a.start();
        b.start();
        c.start();
    }
}
/*Output
Thread i0
Thread i-3
Thread i-6
Thread i-9
Thread i-12
Thread i-15
Thread C
Thread i0
Thread i-2
Thread i-4
Thread i-6
Thread i-8
Thread i-10
Thread B
Thread i0
Thread i-1
Thread i-2
Thread i-3
Thread i-4
Thread i-5
Thread A*/
/*MultiThreading*/

// Create multiple threads.
class NewThread implements Runnable {
String name; // name of thread
Thread t;
NewThread(String threadname) {
name = threadname;
t = new Thread(this, name);
System.out.println("New thread: " + t);
t.start(); // Start the thread
}
// This is the entry point for thread
public void run() {
try {
for(int i = 5; i > 0; i--) {
System.out.println(name + ": " + i);
```

```
Thread.sleep(1000);
} catch (InterruptedException e) {
System.out.println(name + "Interrupted");
}
System.out.println(name + " exiting.");
}
}
class MultiThreadDemo {
public static void main(String args[]) {
new NewThread("One"); // start threads
new NewThread("Two");
new NewThread("Three");
try {
// wait for other threads to end
Thread.sleep(10000);
} catch (InterruptedException e) {
System.out.println("Main thread Interrupted");
}
System.out.println("Main thread exiting.");
}
}
/*OUTPUT
New thread: Thread[One,5,main]
New thread: Thread[Two,5,main]
New thread: Thread[Three,5,main]
Three: 5Two: 5
One: 5
Three: 4
Two: 4
One: 4
Three: 3
Two: 3
One: 3
Three: 2
Two: 2
One: 2
Three: 1
One: 1
Two: 1
One exiting.
Three exiting.
Two exiting.
Main thread exiting.*/
/*The following program creates three child threads
*/

// Using join() to wait for threads to finish.
class NewThread implements Runnable {
String name; // name of thread
Thread t;
NewThread(String threadname) {
name = threadname;
t = new Thread(this, name);
System.out.println("New thread: " + t);
t.start(); // Start the thread
}
// This is the entry point for thread.
public void run() {
try {
for(int i = 5; i > 0; i--) {
System.out.println(name + ": " + i);
Thread.sleep(1000);
}
} catch (InterruptedException e) {
System.out.println(name + " interrupted.");
}
System.out.println(name + " exiting.");
}
}
class DemoJoin {
public static void main(String args[]) {
NewThread ob1 = new NewThread("One");
NewThread ob2 = new NewThread("Two");
NewThread ob3 = new NewThread("Three");
System.out.println("Thread One is alive: "
```

```
+ ob1.t.isAlive());
System.out.println("Thread Two is alive: "
+ ob2.t.isAlive());
System.out.println("Thread Three is alive: "
+ ob3.t.isAlive());
// wait for threads to finish
try {
System.out.println("Waiting for threads to finish.");
ob1.t.join();
ob2.t.join();
ob3.t.join();
} catch (InterruptedException e) {
System.out.println("Main thread Interrupted");
}
System.out.println("Thread One is alive: "
+ ob1.t.isAlive());
System.out.println("Thread Two is alive: "
+ ob2.t.isAlive());
System.out.println("Thread Three is alive: "
+ ob3.t.isAlive());
System.out.println("Main thread exiting.");
}
}
/*OUTPUT
New thread: Thread[One,5,main]
New thread: Thread[Two,5,main]
New thread: Thread[Three,5,main]
Thread One is alive: true
Thread Two is alive: true
Thread Three is alive: true
Waiting for threads to finish.
Three: 5
One: 5
Two: 5
Three: 4
One: 4
Two: 4
Three: 3
One: 3
Two: 3
Three: 2
Two: 2
One: 2
Three: 1
One: 1
Two: 1
One exiting.
Three exiting.
Two exiting.
Thread One is alive: false
Thread Two is alive: false
Thread Three is alive: false
Main thread exiting.*/
/*Uses join() to ensure that the main thread is the last to stop. It also demonstrates the isAlive( ) method.
*/

class Table{
 synchronized void printTable(int n){//synchronized method
    for(int i=1;i<=5;i++){
        System.out.println(n*i);
        try{
            Thread.sleep(400);
        }catch(Exception e){System.out.println(e);}
    }
  }
}

class MyThread1 extends Thread{
    Table t;
    MyThread1(Table t){
        this.t=t;
    }
    public void run(){
        t.printTable(5);
    }
}
```

```
304 class MyThread2 extends Thread{
305     Table t;
306     MyThread2(Table t){
307         this.t=t;
308     }
309     public void run(){
310         t.printTable(100);
311     }
312 }
313
314 public class TestSynchronization2{
315     public static void main(String args[]){
316         Table obj = new Table();//only one object
317         MyThread1 t1=new MyThread1(obj);
318         MyThread2 t2=new MyThread2(obj);
319     t1.start();
320     t2.start();
321     }
322 }
323 /*Synchronization in Java*/
324
325 import java.io.*;
326 import java.net.*;
327 public class MyServer {
328     public static void main(String[] args){
329         try{
330             ServerSocket ss=new ServerSocket(6666);
331             Socket s=ss.accept();//establishes connection
332             DataInputStream dis=new DataInputStream(s.getInputStream());
333             String str=(String)dis.readUTF();
334             System.out.println("message= "+str);
335             ss.close();
336         }
337         catch(Exception e){System.out.println(e);}
338     }
339 }
340
341 import java.io.*;
342 import java.net.*;
343 public class MyClient {
344     public static void main(String[] args) {
345         try{
346             Socket s=new Socket("localhost",6666);
347             DataOutputStream dout=new DataOutputStream(s.getOutputStream());
348             dout.writeUTF("Hello Server");
349             dout.flush();
350             dout.close();
351             s.close();
352         }
353         catch(Exception e){System.out.println(e);}
354     }
355 }
356 /*TCP/IP Socket programming*/
357
358 class Q {
359     int n;
360     boolean valueset = false;
361     synchronized int get() {
362         while (!valueset)
363             try {
364                 wait();
365             } catch (InterruptedException e) {
366                 System.out.println("Thread Interrupted");
367             }
368         System.out.println("Got :" + n);
369         valueset = false;
370         notify();
371         return n;
372     }
373     synchronized void put(int n) {
374         while (valueset)
375             try {
376                 wait();
377             } catch (InterruptedException e) {
378                 System.out.println("Thread interrupted");
379             }
```

```
380         this.n = n;
381         valueset = true;
382         System.out.println("put " + n);
383         notify();
384     }
385 }
386 class Producer implements Runnable {
387     Q q;
388     Producer(Q q) {
389         this.q = q;
390         new Thread(this, "Producer").start();
391     }
392     public void run() {
393         int i = 0;
394         while (true) {
395             q.put(i++);
396         }
397     }
398 }
399 class Consumer implements Runnable {
400     Q q;
401     Consumer(Q q) {
402         this.q = q;
403         new Thread(this, "Consumer").start();
404     }
405     public void run() {
406         int i = 0;
407         while (true) {
408             q.get();
409         }
410     }
411 }
412 class Demo {
413     public static void main(String args[]) {
414         Q q = new Q();
415         new Producer(q);
416         new Consumer(q);
417         System.out.println("press ctrl+c to exit");
418     }
419 }
420 /*Producer Consumer Problem*/
421
422
423
424 import java.lang.*;
425 public class ThreadPriorityExample extends Thread
426 {
427 public void run()
428 {
429 System.out.println("Inside the run() method");
430 }
431 public static void main(String argvs[])
432 {
433 ThreadPriorityExample th1 = new ThreadPriorityExample();
434 ThreadPriorityExample th2 = new ThreadPriorityExample();
435 ThreadPriorityExample th3 = new ThreadPriorityExample();
436
437 System.out.println("Priority of the thread th1 is : " + th1.getPriority());
438
439 System.out.println("Priority of the thread th2 is : " + th2.getPriority());
440
441 System.out.println("Priority of the thread th2 is : " + th2.getPriority());
442
443 // Setting priorities of above threads by
444
445 th1.setPriority(6);
446 th2.setPriority(3);
447 th3.setPriority(9);
448
449 System.out.println("Priority of the thread th1 is : " + th1.getPriority());
450 System.out.println("Priority of the thread th2 is : " + th2.getPriority());
451 System.out.println("Priority of the thread th3 is : " + th3.getPriority());
452
453 // Main thread
454 System.out.println("Currently Executing The Thread : " + Thread.currentThread().getName());
455 System.out.println("Priority of the main thread is : " + Thread.currentThread().getPriority());
```

```
456
457 // Priority of the main thread is 10 now
458 Thread.currentThread().setPriority(10);
459 System.out.println("Priority of the main thread is : " + Thread.currentThread().getPriority());
460 }
461 }
462 /*
463 Output:
464 Priority of the thread th1 is : 5
465 Priority of the thread th2 is : 5
466 Priority of the thread th2 is : 5
467 Priority of the thread th1 is : 6
468 Priority of the thread th2 is : 3
469 Priority of the thread th3 is : 9
470 Currently Executing The Thread : main
471 Priority of the main thread is : 5
472 Priority of the main thread is : 10
473 */
474 /*Thread Priority*/
475
476 class TestJoinMethod1 extends Thread{
477     public void run(){
478         for(int i=1;i<=5;i++){
479             try{
480                 Thread.sleep(500);
481                }
482                catch(Exception e){System.out.println(e);}
483               System.out.println(i);
484           }
485     }
486     public static void main(String args[]){
487         TestJoinMethod1 t1=new TestJoinMethod1();
488         TestJoinMethod1 t2=new TestJoinMethod1();
489         TestJoinMethod1 t3=new TestJoinMethod1();
490         t1.start();
491         try{
492             t1.join();
493         }
494         catch(Exception e){System.out.println(e);}
495         t2.start();
496         t3.start();
497      }
498 }
499 /*Start Run Sleep Join*/
500
501 public class JavaGetPriorityExp extends Thread
502 {
503     public void run()
504     {
505         System.out.println("running thread name is:"+Thread.currentThread().getName());
506     }
507     public static void main(String args[])
508     {
509         // creating two threads
510         JavaGetPriorityExp t1 = new JavaGetPriorityExp();
511         JavaGetPriorityExp t2 = new JavaGetPriorityExp();
512         // print the default priority value of thread
513         System.out.println("t1 thread priority : " + t1.getPriority());
514         System.out.println("t2 thread priority : " + t2.getPriority());
515         // this will call the run() method
516         t1.start();
517         t2.start();
518     }
519 }
520 /*Get priority Set priority currentThread*/
521
522
523
```