**Q.1 Write a program to create thread and use getPriority and setPriority methods.**

class ThreadMin extends Thread{

public void run(){

System.out.println("Priority of thread t1(MinPriority) is: "+Thread.currentThread().getPriority());

}

}

class ThreadMax extends Thread{

public void run(){

System.out.println("Priority of thread t2 (MaxPriority) is : "+Thread.currentThread().getPriority());

}

}

class ThreadNorm extends Thread{

public void run(){

System.out.println("Priority of thread t3 (NormPriority) is : "+Thread.currentThread().getPriority());

}

}

class ThreadDefault extends Thread{

public void run(){

System.out.println("Priority of thread t4 is (Bydefault priority ) : "+Thread.currentThread().getPriority());

}

}

class JavaPriorities extends Thread{

public static void main(String args[]){

ThreadMin t1=new ThreadMin();

ThreadMax t2=new ThreadMax();

ThreadNorm t3=new ThreadNorm();

ThreadDefault t4=new ThreadDefault();

t1.setPriority(Thread.MIN\_PRIORITY);

t2.setPriority(Thread.MAX\_PRIORITY);

t3.setPriority(Thread.NORM\_PRIORITY);

t1.start();

t2.start();

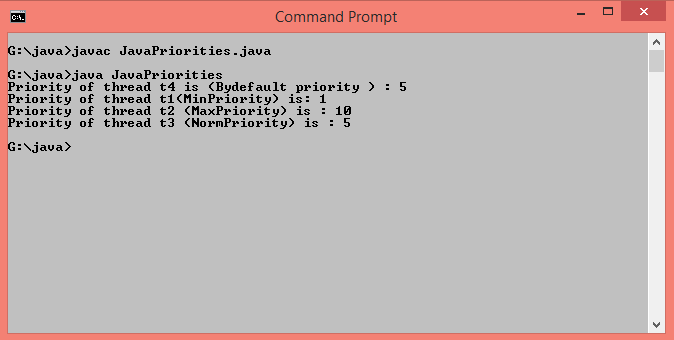
t3.start();

t4.start();

}

}

**Output :**

****

**Q.2 Write a program to implement join method.**

class ThreadJoinMethod extends Thread{

public void run(){

for(int i=1;i<=3;i++){

try{

Thread.sleep(500);

}

catch(Throwable e){

System.out.println(e);

}

System.out.println(i);

}

}

public static void main(String args[]){

ThreadJoinMethod t1=new ThreadJoinMethod();

ThreadJoinMethod t2=new ThreadJoinMethod();

ThreadJoinMethod t3=new ThreadJoinMethod();

t1.start();

try{

t1.join();

}

catch(Throwable e){

System.out.println(e);

}

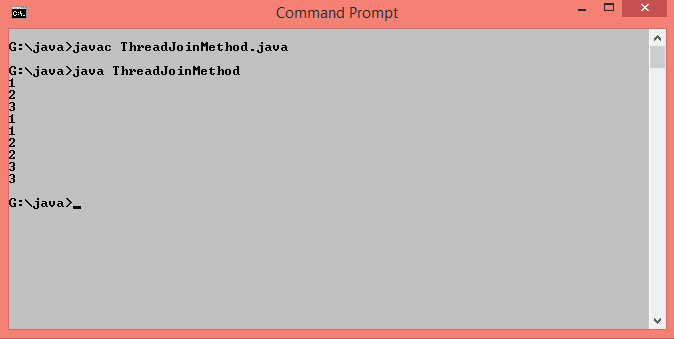
t2.start();

t3.start();

}

}

**Output :**

****

**Q.3 Write a thread class that will repeatedly call the inc() method in an object of type Counter. The object should be a shared global variable. Create several threads, start them all, and wait for all the threads to terminate. Print the final value of the counter, and see whether it is correct.**

import java.util.\*;

public class CounterTest {

static class Counter {

int count;

void inc() {

count = count+1;

}

int getCount() {

return count;

} }

static Counter counter;

static int numberOfIncrements;

static class IncrementerThread extends Thread {

public void run() {

for (int i = 0; i < numberOfIncrements; i++) {

counter.inc();

} } }

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

while (true) {

System.out.print("How many threads do you want to run (Enter 0 to end)? ");

int numberOfThreads = in.nextInt();

if (numberOfThreads <= 0)

break;

do {

System.out.println();

System.out.println("How many times should each thread increment the counter? ");

numberOfIncrements = in.nextInt();

if (numberOfIncrements < 1) {

System.out.println("Number of increments must be positive.");

numberOfIncrements = 1;

}

} while (numberOfIncrements <= 0);

System.out.println();

System.out.println("Using " + numberOfThreads + " threads.");

System.out.println("Each thread increments the counter "

+ numberOfIncrements + " times.");

IncrementerThread[] workers = new IncrementerThread[numberOfThreads];

counter = new Counter();

for (int i = 0; i < numberOfThreads; i++)

workers[i] = new IncrementerThread();

for (int i = 0; i < numberOfThreads; i++)

workers[i].start();

for (int i = 0; i < numberOfThreads; i++) {

try {

workers[i].join();

}

catch (InterruptedException e) {

} }

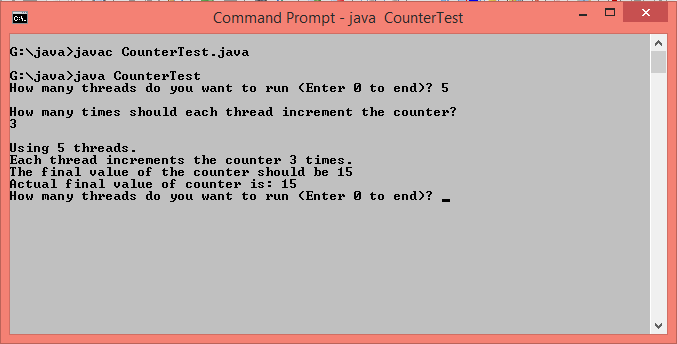
System.out.println("The final value of the counter should be "

+ (numberOfIncrements\*numberOfThreads));

System.out.println("Actual final value of counter is: " + counter.getCount());

} } }

**Output :**

****

**Q.4 Implement three classes: Storage, Counter, and Printer. The Storage class should store an integer. The Counter class should create a thread that starts counting from 0 (0, 1, 2, 3 ...) and stores each value in the Storage class. The Printer class should create a thread that keeps reading the value in the Storage class and printing it. Write a program that creates an instance of the Storage class and sets up a Counter and a Printer object to operate on it.**

**Modify the program from the previous exercise to ensure that each number is printed exactly once, by adding suitable synchronization**

public class Counter implements Runnable{

public static void main(String[] args) {

Storage store = new Storage();

Counter c1 = new Counter(store);

Printer p1 = new Printer(store);

}

Storage st;

public Counter(Storage store){

st = store;

new Thread(this, "Counter").start();

}

@Override

public void run() {

for(int i = 0 ; i < 3; i++){

st.setValue(i);

}

}

}

class Printer implements Runnable{

Storage st;

public Printer(Storage st){

this.st = st;

new Thread(this, "Printer").start();

}

@Override

public void run() {

for(int i = 0; i < 3; i++)

System.out.println(Thread.currentThread().getName() + " " + st.getValue());

}

}

class Storage{

int i;

public synchronized void setValue(int i){

this.i = i;

}

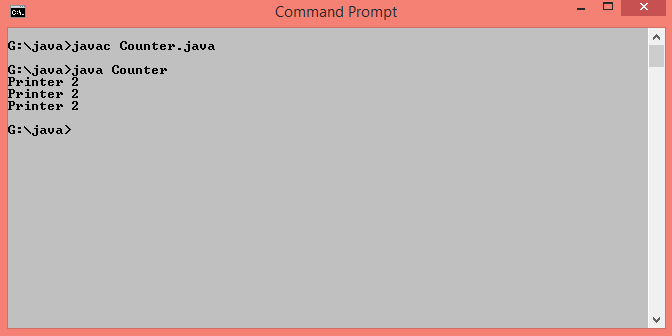
public synchronized int getValue(){

return this.i;

}

}

**Output :**

****