**Adv. Java Lab Assignment**

**MultiThreading 20 Programs**

**Name:-Milind Bajaj**

**M.Tech(5th Sem)**

**Rollno.:-IT-2k17-35**

**1.Program of Create Thread by implementing Runnable Interface.**

**Solution:-**

import java.util.\*;

class subTh implements Runnable

{

void get() {

System.out.println("WelCome in threading");

}

public void run()

{

System.out.println("Hello world!");

}

}

class FirstThread {

public static void main(String[] args) {

subTh st=new subTh();

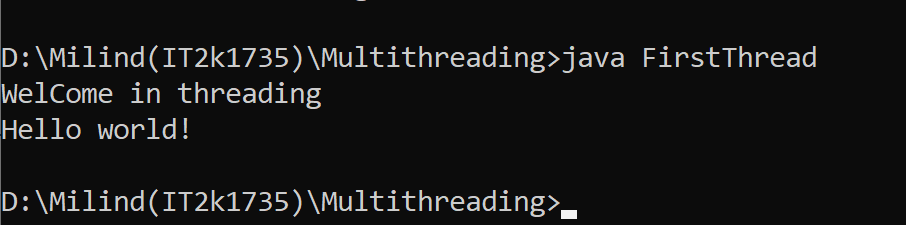
Thread ThreadEx=new Thread(st);

ThreadEx.start();

st.get();

}}

**Output 1:-**

****

**2.Program of Creating Thread by Thread Class.**

**Solution:-**

class ThreadDemo extends Thread {

private Thread t;

private String threadName;

ThreadDemo( String name){

threadName = name;

System .out.println("Creating " + threadName );

}

public ThreadDemo() {

}

public void run() {

System .out.println("Running " + threadName );

try {

for(int i = 4; i > 0; i--) {

System .out.println("Thread: " + threadName + ", " + i);

Thread.sleep(50);

}

} catch (InterruptedException e) {

System .out.println("Thread " + threadName + " interrupted.");

}

System .out.println("Thread " + threadName + " exiting.");

}

public void start ()

{

System .out.println("Starting " + threadName );

if (t == null)

{

t = new Thread (this, threadName);

t.start ();

}}}

class ExtendThread {

public static void main(String args[]) {

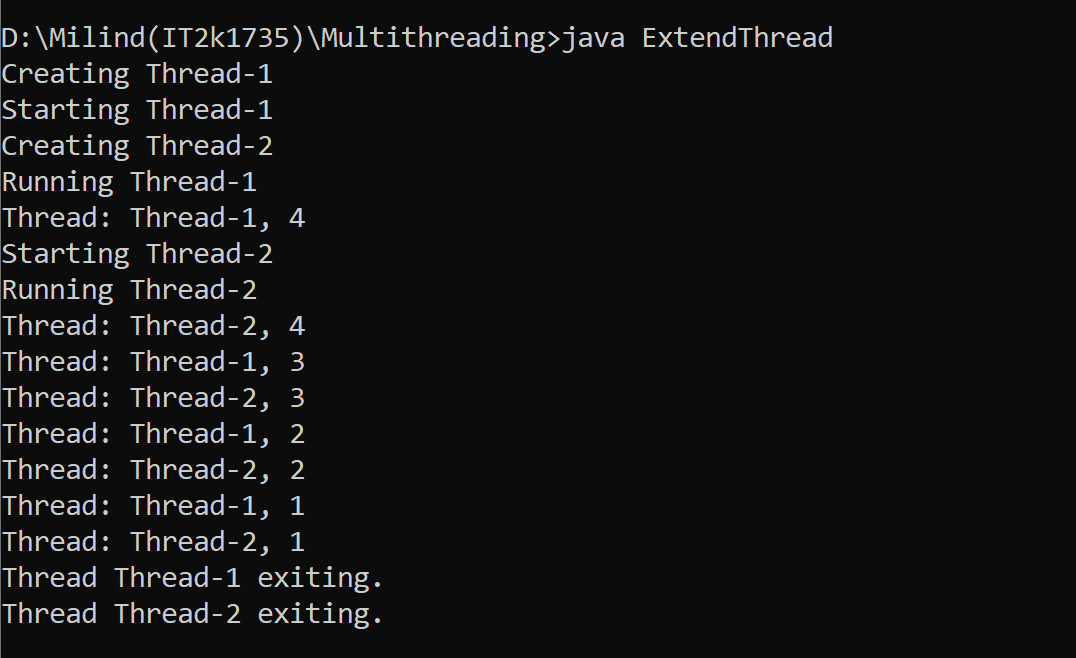
ThreadDemo T1 = new ThreadDemo( "Thread-1");

T1.start();

ThreadDemo T2 = new ThreadDemo( "Thread-2");

T2.start(); }}

**Output 2:-**

****

**3.Program of Extending Thread and Creating its Child.**

**Solution:-**

class Count extends Thread

{

Count()

{

super("my extending thread");

System.out.println("my thread created" + this);

start();

}

public void run()

{

try

{

for (int i=0 ;i<10;i++)

{

System.out.println("Printing the count " + i);

Thread.sleep(1000);

}

}

catch(InterruptedException e)

{

System.out.println("my thread interrupted");

}

System.out.println("My thread run is over" );

}

}

class ExtendingExample

{

public static void main(String args[])

{

Count cnt = new Count();

try

{

while(cnt.isAlive())

{

System.out.println("Main thread will be alive till the child thread is live");

Thread.sleep(1500); } }

catch(InterruptedException e)

{

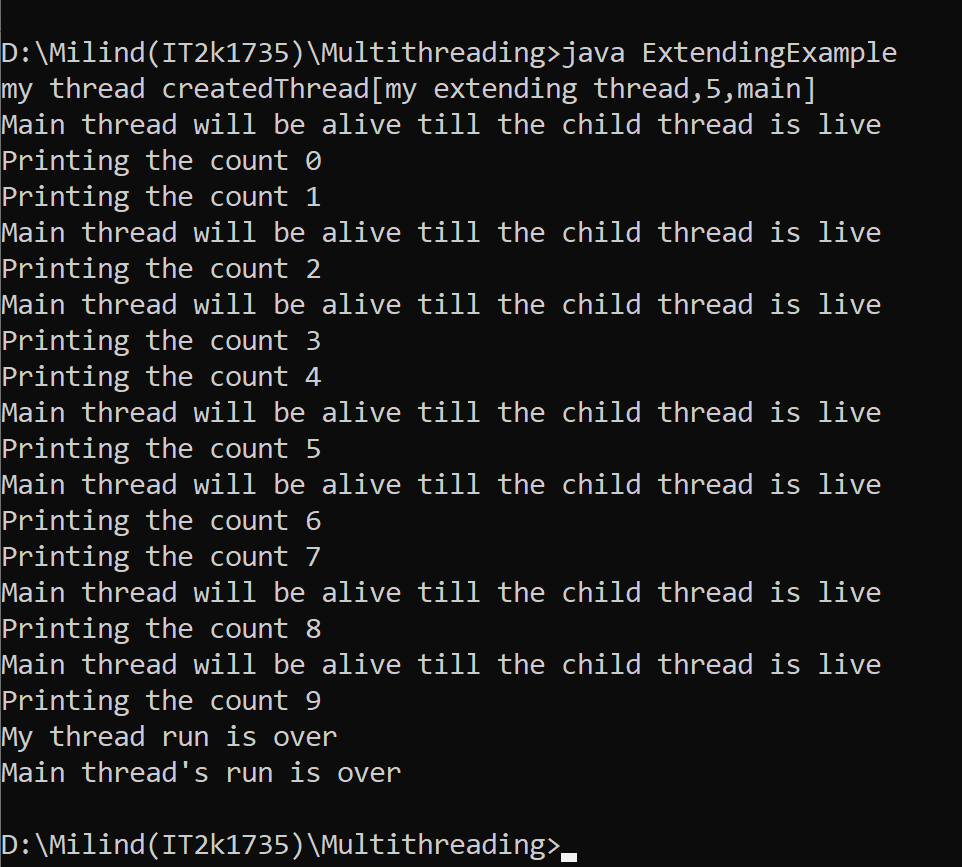
System.out.println("Main thread interrupted");

}

System.out.println("Main thread's run is over" );

}}

**Output 3:-**

****

**4.Program of Creating Thread and Using sleep() method and Terminate it.**

**Solution:-**

public class MainThread {

public static void main(String[] args) {

for (int i = 0; i < 5; i++) {

try {

System.out.println(Thread.currentThread() + " " + i);

Thread.sleep(1010);

} catch (InterruptedException e) {

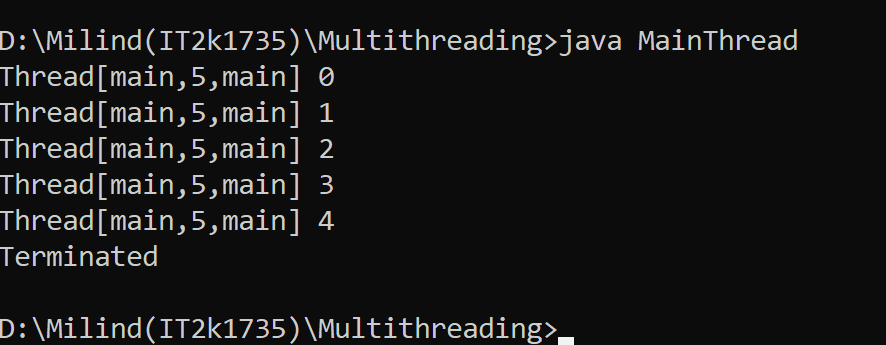
System.out.println("Interrupted");

}

}

System.out.println("Terminated"); }}

**Output 4:-**

****

**5.Program of Creating Thread and Exiting From Main Thread.**

**Solution:-**

public class Main {

static volatile boolean exit = false;

public static void main(String[] args)

{

System.out.println("started main thread..");

new Thread() {

public void run()

{

System.out.println("started inside thread..");

while (!exit) {

}

System.out.println("exiting inside thread..");

}

}

try {

Thread.sleep(500);

}

catch (InterruptedException e) {

System.out.println("Caught :" + e);

}

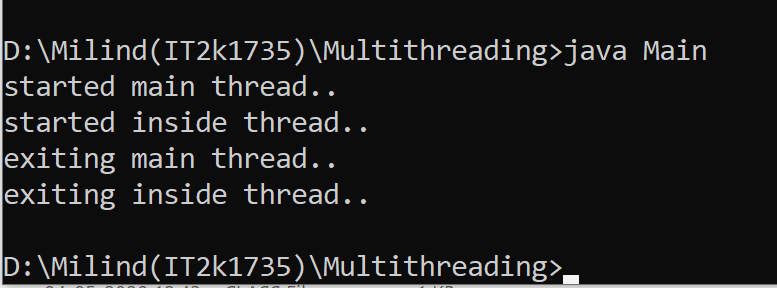
exit = true;

System.out.println("exiting main thread..");

}

}

**Output 5:-**

****

**6.Program of Guessing A Number with the help of Thread.**

**Solution:-**

public class GuessANumber extends Thread {

private int number;

public GuessANumber(int number) {

this.number = number; }

public static void main(String[] args) {

GuessANumber t=new GuessANumber(12);

t.start(); }

public void run() {

int counter = 0;

int guess = 0

do { guess = (int) (Math.random() \* 50 + 1);

System.out.println(this.getName() + " guesses " + guess);

counter++;

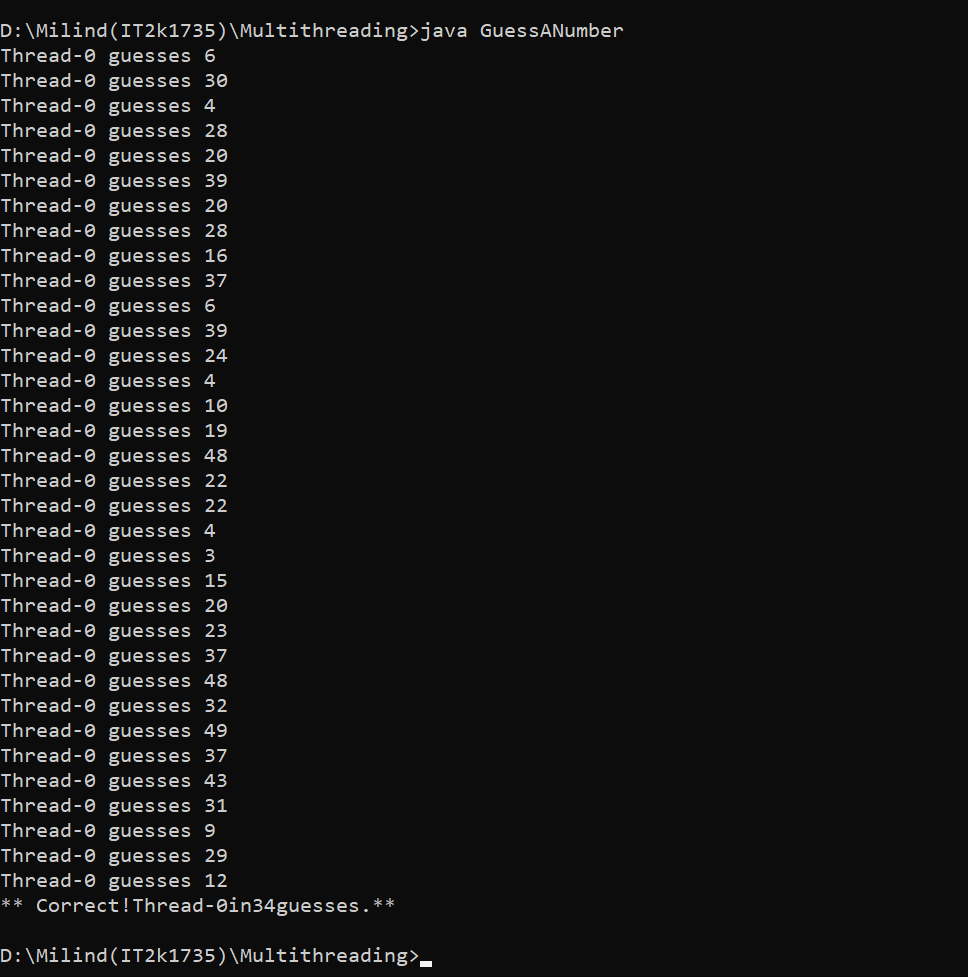
} while(guess != number);

System.out.println("\*\* Correct!" + this.getName() + "in" + counter + "guesses.\*\*");

}

}

**Output 6:-**

****

**7.Program of GetName and SetName of Thread.**

**Solution:-**

public class GetNameandSetName implements Runnable {

public static void main(String[] args) {

GetNameandSetName s1 = new GetNameandSetName();

Thread t1 = new Thread(s1);

t1.start();

System.out.println(t1.getId());

Thread t2 = new Thread(s1);

System.out.println(t2.getId());

t2.start();

t2.setName("Dhoni");

t1.setName("pant");

}

public void run() {

for (int i = 0; i < 5; i++) {

try {

Thread.sleep(500);

} catch (InterruptedException e) {

System.out.println("error");

}

System.out.println("Name of the thread is " + Thread.currentThread().getName() + " " + i); }}}

**Output 7:-**

****

**8.Program of Creating Daemon Thread.**

**Solution:-**

class DaemonThreadDemo

{

public static void main (String [] args)

{

MyThread5 mt = new MyThread5 ();

mt.setDaemon (true);

mt.start ();

try

{

Thread.sleep (100);

}

catch (InterruptedException e)

{

}

}

}

class MyThread5 extends Thread

{

public void run ()

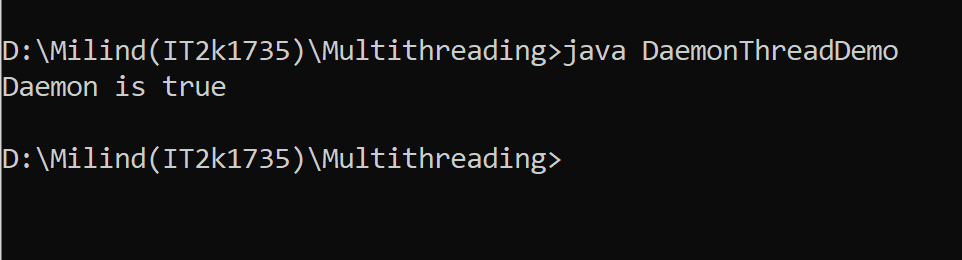
{

System.out.println ("Daemon is " + isDaemon ());

}

}

**Output 8:-**

****

**9.Program of Creating Table with Thread.**

**Solution:-**

class Table {

synchronized void printTable(int n) {

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);

}

}

class MyThread2 extends Thread{

Table t;

MyThread2(Table t){

this.t=t;

}

public void run(){

t.printTable(100);

}}

class Table1{

public static void main(String args[]){

Table obj = new Table();//only one object

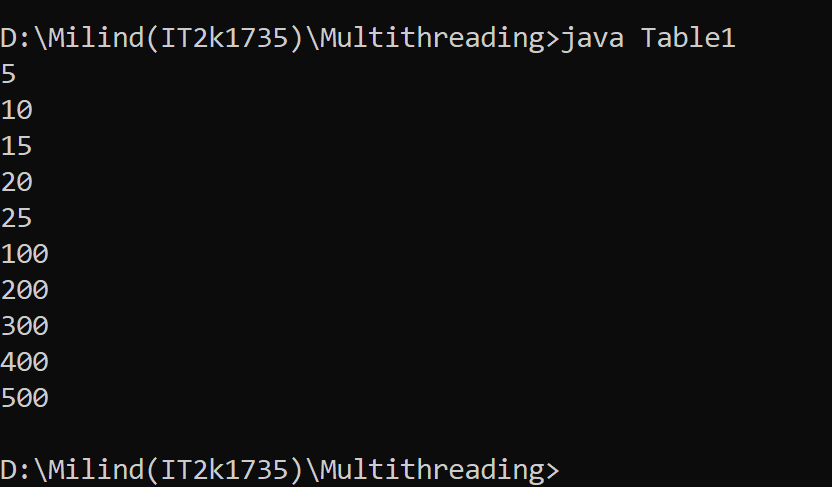
MyThread1 t1=new MyThread1(obj);

MyThread2 t2=new MyThread2(obj);

t1.start();

t2.start(); }}

**Output 9:-**

****

**10.Program of Naming my thread.**

**Solution:-**

class NameMyThread

{

public static void main (String [] args)

{

MyThread4 mt;

if (args.length == 0)

mt = new MyThread4 ();

else

mt = new MyThread4(args [0]);

mt.start ();

}

}

class MyThread4 extends Thread {

MyThread4() {

}

MyThread4(String name) {

setName(name);

}

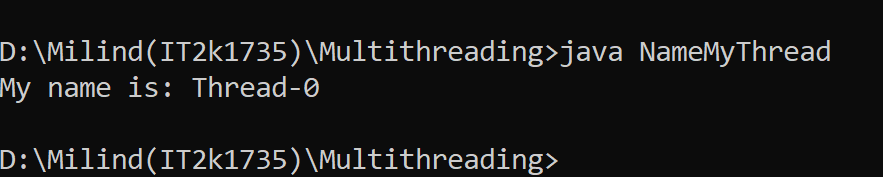
public void run() {

System.out.println("My name is: " + getName());

}

}

**Output 10:-**

****

**11.Program of Testing Daemon Thread.**

**Solution:-**

class TestDaemonThread1 extends Thread{

public void run(){

if(Thread.currentThread().isDaemon()){//checking for daemon thread

System.out.println("daemon thread work");

}

else{

System.out.println("user thread work");

}

}

public static void main(String[] args){

TestDaemonThread1 t1=new TestDaemonThread1();

TestDaemonThread1 t2=new TestDaemonThread1();

TestDaemonThread1 t3=new TestDaemonThread1();

t1.setDaemon(true);

t1.start();

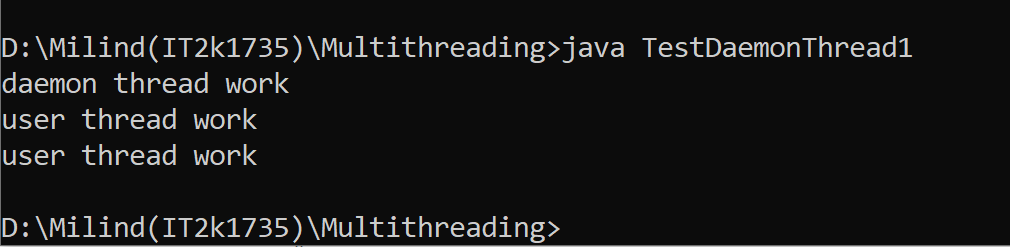
t2.start();

t3.start();

}

}

**Output 11:-**

****

**12.Program of checking Thread Priority and Count Active Thread.**

**Solution:-**

public class ThreadDemo2 {

public static void main(String[] args) {

Thread t = Thread.currentThread();

t.setName("Admin Thread");

t.setPriority(1);

System.out.println("Thread = " + t);

int priority= t.getPriority();

System.out.println("Thread priority= " + priority);

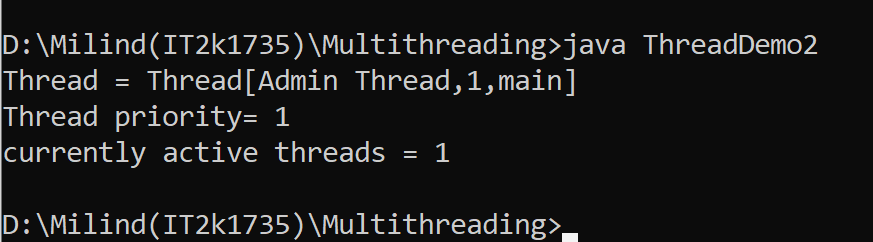
int count = Thread.activeCount();

System.out.println("currently active threads = " + count);

}

}

**Output 12:-**

****

**13.Program of checking Thread whether it is running or not.**

**Solution:-**

class ThreadTest1 extends Thread

{

public void run()

{

System.out.println("In run");

yield();

System.out.println("Leaving run");

}

public static void main(String []argv)

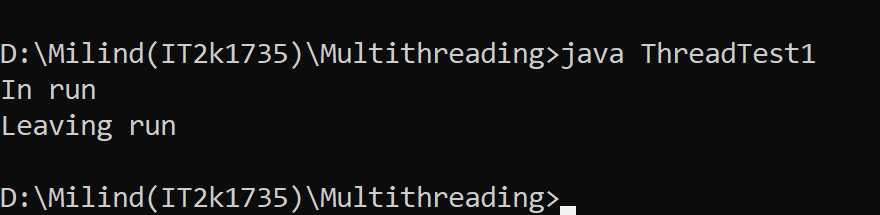
{

(new ThreadTest1()).start();

}

}

**Output 13:-**

****

**14.Program of Multithreading.**

**Solution:-**

class MultithreadingDemo extends Thread

{

public void run()

{

try{

System.out.println("Thread:"+Thread.currentThread().getId()+"\tis running");

}

catch(Exception e)

{

System.out.println("Exception is caught");

}

}

}

public class Mthread1 {

public static void main(String[] args) {

int n = 5;

for (int i = 0; i <= 5; i++) {

MultithreadingDemo object = new MultithreadingDemo();

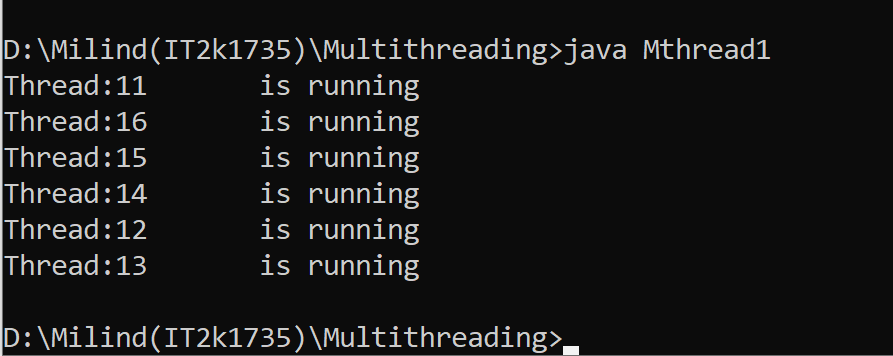
object.start();

}

}

}

**Output 14:-**

****

**15.Program of Multithreading Using Runnable.**

**Solution:-**

public class MultiThreadingusingRunnable {

public static void main(String[] args) {

StrtThread st = new StrtThread();

Thread thread = new Thread(st);

thread.start();

}

}

class StrtThread implements Runnable {

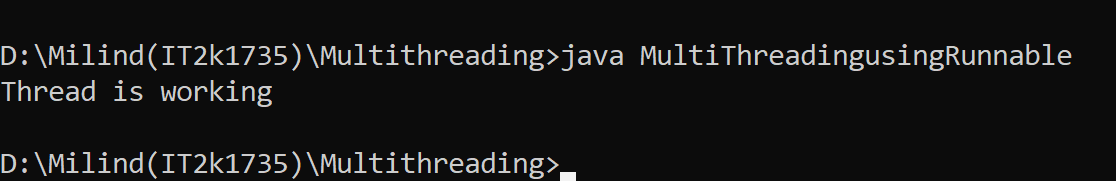
public void run() {

System.out.println("Thread is working");

}

}

**Output 15:-**

****

**16.Program of Multithreading Using Thread Class.**

**Solution:-**

public class MultiThreadingusingThread {

public static void main(String[] args) {

StartThread st1 = new StartThread();

st1.start();

}

}

class StartThread extends Thread {

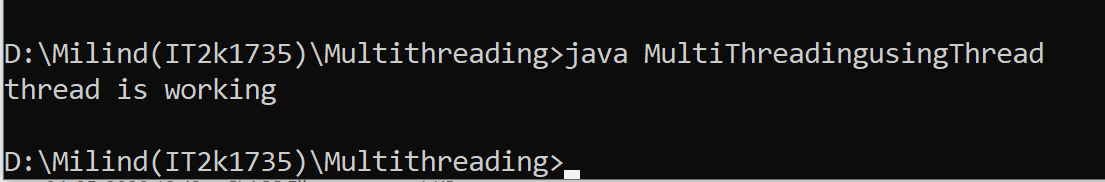
public void run() {

System.out.println("thread is working");

}

}

**Output 16:-**

****

**17.Program of Getting Priority of Multithreading.**

**Solution:-**

public class MultithreadgetPriority extends Thread {

public static void main(String[] args) {

MultithreadgetPriority m1 = new MultithreadgetPriority();

MultithreadgetPriority m2 = new MultithreadgetPriority();

m1.start();

m2.start();

m1.setPriority(MIN\_PRIORITY);

System.out.println(m1.getPriority());

System.out.println(m2.getPriority());

System.out.println(m1.getId());

}

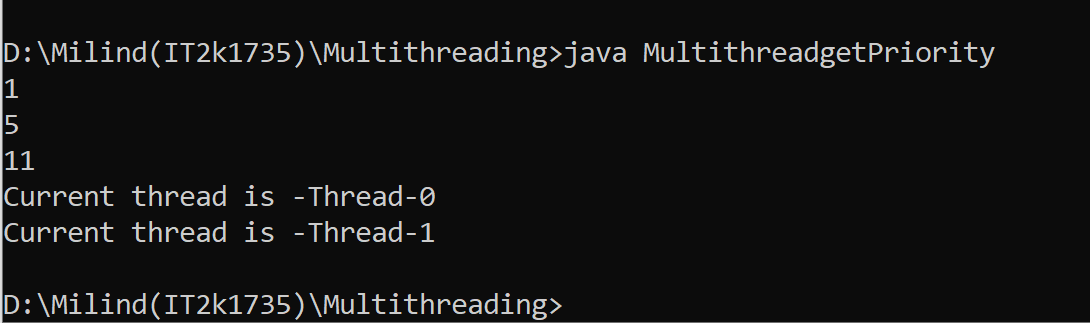
public void run() {

System.out.println("Current thread is -" + Thread.currentThread().getName());

}

}

**Output 17:-**

****

**18.Program of Multihtreading Using sleep Method.**

**Solution:-**

class Starthread implements Runnable {

public void run() {

for (int i = 1; i < 6; i++) {

try {

Thread.sleep(5000);

} catch (InterruptedException e) {

System.out.println(e);

}

System.out.print(i + " ");

System.out.println(Thread.currentThread().getName());

}

}

}

class MultiThreadSleepMethod {

public static void main(String[] args) {

Starthread t1 = new Starthread();

Starthread t2 = new Starthread();

Thread th1 = new Thread(t1);

Thread th2 = new Thread(t2);

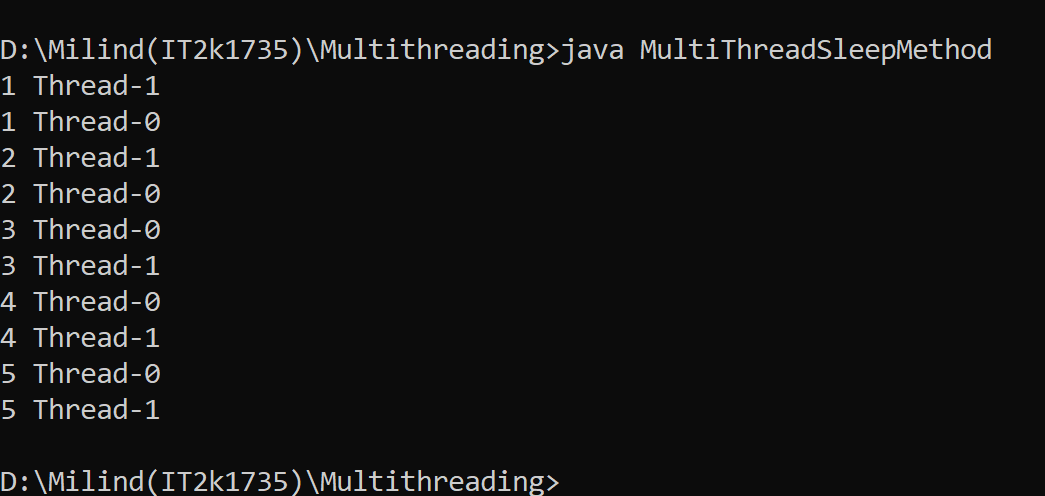
th1.start();

th2.start();

}

}

**Output 18:-**

****

**19.Program of creating thread Using join Method.**

**Solution:-**

class UsingjoinMethod {

public static void main(String[] args) {

strt t1 = new strt();

strt t2 = new strt();

strt t3 = new strt();

t1.start();

try {

t1.join(1500);

} catch (Exception e) {

System.out.println(e);

}

t2.start();

t3.start();

}

}

class strt extends Thread {

public void run() {

for (int i = 1; i < 6; i++) {

try {

Thread.sleep(500);

} catch (InterruptedException e) {

System.out.println(e);

}

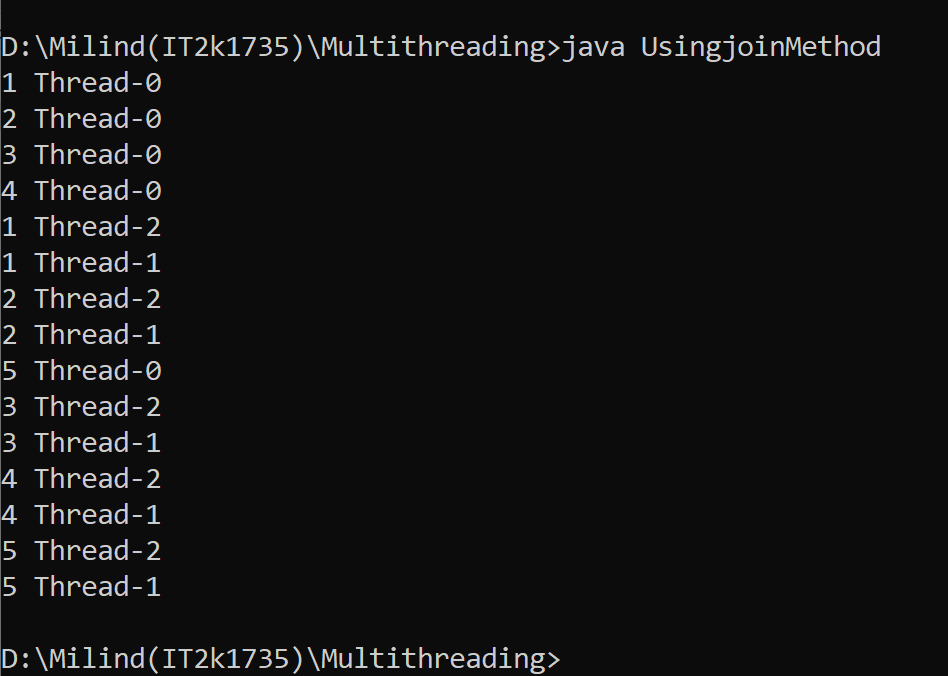
System.out.println(i + " " + Thread.currentThread().getName());

}

}

}

**Output 19:-**

****

**20.Program of creating thread using Yield Method.**

**Solution:-**

public class YieldMethod extends Thread {

public static void main(String[] args) {

YieldMethod m1 = new YieldMethod();

YieldMethod m2 = new YieldMethod();

YieldMethod m3 = new YieldMethod();

m1.setName("Rohit");

m2.setName("kohli");

m3.setName("pant");

m1.start();

m2.start();

m2.suspend();

m3.start();

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

System.out.println(Thread.currentThread().getName() + " in control");

}

}

public void run() {

for (int i = 0; i < 5; i++) {

try {

Thread.sleep(500);

} catch (InterruptedException e) {

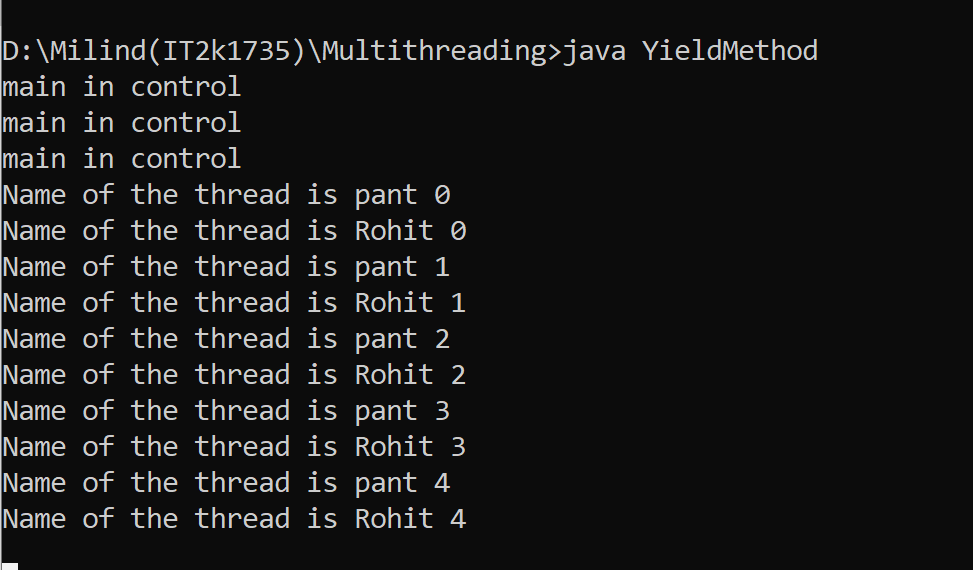
System.out.println("error");

}

System.out.println("Name of the thread is " + Thread.currentThread().getName() + " " + i);

}}}

**Output 20:-**

****

**END**