**MULTITHREADING** **PROGRAMS**:-

**SUBMITTED TO:- DR. NITIN NAGAR SIR**

BY:- MUSKAN MANDLOI (IT-2K19-34)

1. public class deadlock {

public static void main(String[] args) {

final String resource1 = "ratan jaiswal";

final String resource2 = "vimal jaiswal";

Thread t1 = new Thread() {

public void run() {

synchronized (resource1) {

System.out.println("Thread 1: locked resource 1");

try { Thread.sleep(100);} catch (Exception e) {}

synchronized (resource2) {

System.out.println("Thread 1: locked resource 2");

}

}

}

};

Thread t2 = new Thread() {

public void run() {

synchronized (resource2) {

System.out.println("Thread 2: locked resource 2");

try { Thread.sleep(100);} catch (Exception e) {}

synchronized (resource1) {

System.out.println("Thread 2: locked resource 1");

}

}

}

};

t1.start();

t2.start();

}

}

2. package thread3;

class ThreadId extends Thread {

public void run() {

try {

// Displaying the thread that is running

System.out.println ("Thread " + Thread.currentThread().getId() + " is running");

}

catch (Exception e) {

// Throwing an exception

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

}

}

}

public class getid{

public static void main(String[] args)

{

int n = 8; // Number of threads

for (int i=0; i<8; i++)

{

ThreadId object = new ThreadId();

object.start();

}

}

}

3. import java.lang.\*;

public class isAlive implements Runnable {

public void run()

{

Thread t = Thread.currentThread();

System.out.println("Current thread: "

+ t.getName());

// checks if current thread is alive

System.out.println("Is Alive? " + t.isAlive());

}

public static void main(String args[]) throws Exception

{

Thread t = new Thread(new JoinDemo());

t.start();

// Waits for 1000ms this thread to die.

t.join(1000);

System.out.println("\nJoining after 1000"

+ " milliseconds: \n");

System.out.println("Current thread: "

+ t.getName());

// Checks if this thread is alive

System.out.println("Is alive? " + t.isAlive());

}

}

4. import java.lang.\*;

public class join implements Runnable {

public void run()

{

Thread t = Thread.currentThread();

System.out.println("Current thread: "

+ t.getName());

// checks if current thread is alive

System.out.println("Is Alive? " + t.isAlive());

}

public static void main(String args[]) throws Exception

{

Thread t = new Thread(new JoinDemo());

t.start();

// Waits for 1000ms this thread to die.

t.join(1000);

System.out.println("\nJoining after 1000"

+ " milliseconds: \n");

System.out.println("Current thread: "

+ t.getName());

// Checks if this thread is alive

System.out.println("Is alive? " + t.isAlive());

}

}

5. package thread5;

class Thread1 extends Thread {

public void run( ) {

try{

System.out.println (" First thread starts running" );

sleep(10000);

System.out.println (" First thread finishes running" );

}

catch(Exception e){ }

}

}

class Thread2 extends Thread {

public void run( ) {

try{

System.out.println ( "Second thread starts running");

System.out.println ( "Second thread is suspended itself ");

suspend( );

System.out.println (" Second thread runs again" );

}

catch(Exception e){ }

}

}

class method2{

public static void main (String args[ ] ){

try{

Thread1 first = new Thread1( ); // It is a newborn thread i.e. in Newborn state

Thread2 second= new Thread2( ); // another new born thread

first.start( ); // first is scheduled for running

second.start( ); // second is scheduled for running

System.out.println("Revive the second thread" ); // If it is suspended

second.resume( );

System.out.println ("Second thread went for 10 seconds sleep " );

second.sleep (10000);

System.out.println ("Wake up second thread and finishes running" );

System.out.println ( " Demonstration is finished ");

}

catch(Exception e){ }

}

}

6. package thread5;

import java.util.\*;

class ClassA extends Thread{

public void run() {

System.out.println("Start Thread A ....");

for(int i = 1; i <= 5; i++) {

if (i==1)

yield();

System.out.println("From Thread A: i = "+ i);

}

System.out.println("... Exit Thread A");

}

}

class ClassB extends Thread{

public void run() {

System.out.println("Start Thread B ....");

for(int j = 1; j <= 5; j++) {

System.out.println("From Thread B: j = "+ j);

if (j==2) stop();

}

System.out.println("... Exit Thread B");

}

}

class ClassC extends Thread{

public void run() {

System.out.println("Start Thread C ....");

for(int k = 1; k <= 5; k++) {

System.out.println("From Thread B: j = "+ k);

if (k==3){

try{

sleep(1000);

}catch(Exception e){}

}

}

System.out.println("... Exit Thread C");

}

}

7. package generic;

class Helper implements Runnable {

public void run() {

try {

System.out.println("thread2 going to sleep for 5000");

Thread.sleep(5000);

}

catch (InterruptedException e) {

System.out.println("Thread2 interrupted");

}

}

}

8. public class MyThread1

{

public static void main(String argvs[])

{

Thread t= new Thread("My first thread");

t.start();

String str = t.getName();

System.out.println(str);

}

}

9. class Main extends Thread {

public void run()

{

synchronized (this)

{

System.out.println(

Thread.currentThread().getName()

+ "...starts");

try {

this.wait();

}

catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println(

Thread.currentThread().getName()

+ "...notified");

}

}

}

class Main2 extends Thread {

Main1 main1;

Main2(Main1 main1){

this.main1 = main1;

}

public void run()

{

synchronized (this.main1)

{

System.out.println(

Thread.currentThread().getName()

+ "...starts");

try {

this.main1.wait();

}

catch (InterruptedException e) {

e.printStackTrace();

}

System.out.println(

Thread.currentThread().getName()

+ "...notified");

}

}

}

class Main3 extends Thread {

Main1 main1;

Main3(Main1 main1) { this.Main1 = main1; }

public void run()

{

synchronized (this.main1)

{

System.out.println(

Thread.currentThread().getName()

+ "...starts");

this.geeks1.notifyAll();

System.out.println(

Thread.currentThread().getName()

+ "...notified");

}

}

}

class NotifyAll {

public static void main(String[] args)

throws InterruptedException

{

Geek1 main1= new Main1();

Geek2 main2 = new Main2(main1);

Geek3 main3 = new Main3(main1);

Thread t1 = new Thread(main1, "Thread-1");

Thread t2 = new Thread(main2, "Thread-2");

Thread t3 = new Thread(main3, "Thread-3");

t1.start();

t2.start();

Thread.sleep(100);

t3.start();

}

}

10. package thread5;

class ClassA extends Thread{

public void run() {

System.out.println("Start Thread A ....");

for(int i = 1; i <= 5; i++) {

System.out.println("From Thread A: i = "+ i);

}

System.out.println("... Exit Thread A");

}

}

class ClassB extends Thread{

public void run() {

System.out.println("Start Thread B ....");

for(int j = 1; j <= 5; j++) {

System.out.println("From Thread B: j = "+ j);

}

System.out.println("... Exit Thread B");

}

}

class ClassC extends Thread{

public void run() {

System.out.println("Start Thread C ....");

for(int k = 1; k <= 5; k++) {

System.out.println("From Thread B: j = "+ k);

}

System.out.println("... Exit Thread C");

}

}

class priority{

public static void main (String args[]) {

ThreadA t1 = new ThreadA();

ThreadB t2 = new ThreadB();

ThreadC t3 = new ThreadC();

t3.setPriority(Thread.MAX\_PRIORITY);

t2.setPriority(t2.getPriority() + 1);

t1.setPriority(Thread.MIN\_PRIORITY);

t1.start(); t2.start(); t3.start();

System.out.println("... End of executuion ");

}

}

11. public class ReentrantExample{

public static void main(String args[]){

final ReentrantExample re=new ReentrantExample();

Thread t1=new Thread(){

public void run(){

re.m();//calling method of Reentrant class

}

};

t1.start();

}}

12. class Multi3 implements Runnable{

public void run(){

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

}

public static void main(String args[]){

Multi3 m1=new Multi3();

Thread t1 =new Thread(m1); // Using the constructor Thread(Runnable r)

t1.start();

}

13. import java.lang.\*;

public class SleepDemo implements Runnable {

Thread t;

public void run()

{

for (int i = 0; i < 4; i++) {

System.out.println(

Thread.currentThread().getName() + " "

+ i);

try {

// thread to sleep for 1000 milliseconds

Thread.sleep(1000);

}

catch (Exception e) {

System.out.println(e);

}

}

}

public static void main(String[] args) throws Exception

{

Thread t = new Thread(new SleepDemo());

// call run() function

t.start();

Thread t2 = new Thread(new SleepDemo());

// call run() function

t2.start();

}

}

14. package synchronization;

import java.util.\*;

class Table2{

void printTable(int n) {

for(int i=1;i<5;i++) {

System.out.println(n\*i);

try {

Thread.sleep(500);

}catch(Exception e) {

System.out.println(e);

}

}

}

}

class MyThread1 extends Thread{

Table2 t;

MyThread1(Table2 t){

this.t=t;

}

public void run() {

t.printTable(5);

}

}

class MyThread2 extends Thread{

Table2 t;

MyThread2(Table2 t){

this.t=t;

}

public void run() {

t.printTable(100);

}

}

public class Table1 {

public static void main(String[] args) {

// TODO Auto-generated method stub

Table2 obj=new Table2();

MyThread1 t1=new MyThread1(obj);

MyThread2 t2 =new MyThread2(obj);

t1.start();

t2.start();

}

}

15. package generic;

class Helper implements Runnable {

public void run() {

try {

System.out.println("thread2 going to sleep for 5000");

Thread.sleep(5000);

}

catch (InterruptedException e) {

System.out.println("Thread2 interrupted");

}

}

}

public class Test implements Runnable {

public void run() {

}

public static void main(String[] args) {

Test obj = new Test();

Helper obj2 = new Helper();

Thread thread1 = new Thread(obj);

Thread thread2 = new Thread(obj2);

thread1.start();

thread2.start();

ClassLoader loader = thread1.getContextClassLoader();

Thread thread3 = new Thread(new Helper());

System.out.println(Thread.activeCount());

thread1.checkAccess();

Thread t = Thread.currentThread();

System.out.println(t.getName());

System.out.println("Thread1 name: " + thread1.getName());

System.out.println("Thread1 ID: " + thread1.getId());

System.out.println("Priority of thread1 = " + thread1.getPriority());

System.out.println(thread1.getState());

thread2 = new Thread(obj2);

thread2.start();

thread2.interrupt();

System.out.println("Is thread2 interrupted? " + thread2.interrupted() );

System.out.println("Is thread2 alive? " + thread2.isAlive());

thread1 = new Thread(obj);

thread1.setDaemon(true);

System.out.println("Is thread1 a daemon thread? " + thread1.isDaemon());

System.out.println("Is thread1 interrupted? " + thread1.isInterrupted());

System.out.println("thread1 waiting for thread2 to join");

try {

thread2.join();

}

catch (InterruptedException e) {

e.printStackTrace();

}

thread1.setName("child thread xyz");

System.out.println("New name set for thread 1" + thread1.getName());

thread1.setPriority(5);

thread2.yield();

System.out.println(thread1.toString());

Thread[] tarray = new Thread[3];

Thread.enumerate(tarray);

System.out.println("List of active threads:");

System.out.printf("[");

for (Thread thread : tarray) {

System.out.println(thread);

}

System.out.printf("]\n");

System.out.println(Thread.getAllStackTraces());

ClassLoader classLoader = thread1.getContextClassLoader();

System.out.println(classLoader.toString());

System.out.println(thread1.getDefaultUncaughtExceptionHandler());

thread2.setUncaughtExceptionHandler(thread1.getDefaultUncaughtExceptionHandler());

thread1.setContextClassLoader(thread2.getContextClassLoader());

thread1.setDefaultUncaughtExceptionHandler(thread2.getUncaughtExceptionHandler());

thread1 = new Thread(obj);

StackTraceElement[] trace = thread1.getStackTrace();

System.out.println("Printing stack trace elements for thread1:");

for (StackTraceElement e : trace) {

System.out.println(e);

}

ThreadGroup grp = thread1.getThreadGroup();

System.out.println("ThreadGroup to which thread1 belongs " + grp.toString());

System.out.println(thread1.getUncaughtExceptionHandler());

System.out.println("Does thread1 holds Lock? " + thread1.holdsLock(obj2));

Thread.dumpStack();

}

}

16. class Customer{

int amount=10000;

synchronized void withdraw(int amount){

System.out.println("going to withdraw...");

if(this.amount<amount){

System.out.println("Less balance; waiting for deposit...");

try{wait();}catch(Exception e){}

}

this.amount-=amount;

System.out.println("withdraw completed...");

}

synchronized void deposit(int amount){

System.out.println("going to deposit...");

this.amount+=amount;

System.out.println("deposit completed... ");

notify();

}

}

class TestBank{

public static void main(String args[]){

final Customer c=new Customer();

new Thread(){

public void run(){c.withdraw(15000);}

}.start();

new Thread(){

public void run(){c.deposit(10000);}

}.start();

}}

17. public 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();//creating thread

TestDaemonThread1 t2=new TestDaemonThread1();

TestDaemonThread1 t3=new TestDaemonThread1();

t1.setDaemon(true);//now t1 is daemon thread

t1.start();//starting threads

t2.start();

t3.start();

}

}

18. class TestDaemonThread2 extends Thread{

public void run(){

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

System.out.println("Daemon: "+Thread.currentThread().isDaemon());

}

public static void main(String[] args){

TestDaemonThread2 t1=new TestDaemonThread2();

TestDaemonThread2 t2=new TestDaemonThread2();

t1.start();

t1.setDaemon(true);//will throw exception here

t2.start();

}

}

19. class TestInterruptingThread1 extends Thread{

public void run(){

try{

Thread.sleep(1000);

System.out.println("task");

}catch(InterruptedException e){

throw new RuntimeException("Thread interrupted..."+e);

}

}

public static void main(String args[]){

TestInterruptingThread1 t1=new TestInterruptingThread1();

t1.start();

try{

t1.interrupt();

}catch(Exception e){System.out.println("Exception handled "+e);}

}

}

20. class TestInterruptingThread2 extends Thread{

public void run(){

try{

Thread.sleep(1000);

System.out.println("task");

}catch(InterruptedException e){

System.out.println("Exception handled "+e);

}

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

}

public static void main(String args[]){

TestInterruptingThread2 t1=new TestInterruptingThread2();

t1.start();

t1.interrupt();

}

}

21. class TestInterruptingThread3 extends Thread{

public void run(){

for(int i=1;i<=5;i++)

System.out.println(i);

}

public static void main(String args[]){

TestInterruptingThread3 t1=new TestInterruptingThread3();

t1.start();

t1.interrupt();

}

}

22. public class TestInterruptingThread4 extends Thread{

public void run(){

for(int i=1;i<=2;i++){

if(Thread.interrupted()){

System.out.println("code for interrupted thread");

}

else{

System.out.println("code for normal thread");

}

}//end of for loop

}

public static void main(String args[]){

TestInterruptingThread4 t1=new TestInterruptingThread4();

TestInterruptingThread4 t2=new TestInterruptingThread4();

t1.start();

t1.interrupt();

t2.start();

}

}

23. package thread7;

class Table{

synchronized void printTable(int n) {

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

System.out.println(n\*i);

try {

Thread.sleep(400);

}catch(Exception e){

System.out.println(e);

}

}

}

}

public class TestSynchronization {

public static void main(String[] args) {

// TODO Auto-generated method stub

final Table obj=new Table();

Thread t1=new Thread() {

public void run() {

obj.printTable(5);

}

};

Thread t2=new Thread() {

public void run() {

obj.printTable(100);

}

};

t1.start();

t2.start();

}

}

24. package thread1;

import java.util.\*;

class ThreadA extends Thread{

public void run( ) {

for(int i = 1; i <= 5; i++) {

System.out.println("From Thread A with i = "+ -1\*i);

}

System.out.println("Exiting from Thread A ...");

}

}

class ThreadB extends Thread {

public void run( ) {

for(int j = 1; j <= 5; j++) {

System.out.println("From Thread B with j= "+2\* j);

}

System.out.println("Exiting from Thread B ...");

}

}

class ThreadC extends Thread{

public void run( ) {

for(int k = 1; k <= 5; k++) {

System.out.println("From Thread C with k = "+ (2\*k-1));

}

System.out.println("Exiting from Thread C ...");

}

}

public class thread {

public static void main(String args[]) {

ThreadA a = new ThreadA();

ThreadB b = new ThreadB();

ThreadC c = new ThreadC();

a.start();

b.start();

c.start();

System.out.println("... Multithreading is over ");

}

}

25. package thread2;

import java.util.\*;

class ThreadY implements Runnable {

public void run(){/\* Creating three threads using the Runnable interface and then running them concurrently. \*/

class ThreadX implements Runnable{

public void run( ) {

for(int i = 1; i <= 5; i++) {

System.out.println("Thread X with i = "+ -1\*i);

}

System.out.println("Exiting Thread X ...");

}

{

for(int j = 1; j <= 5; j++) {

System.out.println("Thread Y with j = "+ 2\*j);

}

System.out.println("Exiting Thread Y ...");

}

}

class ThreadZ implements Runnable{

public void run( ) {

for(int k = 1; k <= 5; k++) {

System.out.println("Thread Z with k = "+ (2\*k-1));

}

System.out.println("Exiting Thread Z ...");

}

}

public class thread2 {

public static void main(String []args) {

ThreadX x = new ThreadX();

Thread t1 = new Thread(x);

ThreadY y = new ThreadY();

Thread t2 = new Thread(y);

//ThreadZ z = new ThreadZ();

//Thread t3 = new Thread(z);

Thread t3 = new Thread(new ThreadZ());

t1.start();

t2.start();

t3.start();

System.out.println("... Multithreading is over ");

}

}

26. class Bank {

int total = 100;

void withdrawn(String name, int withdrawal)

{

if (total >= withdrawal) {

System.out.println(name + " withdrawn "

+ withdrawal);

total = total - withdrawal;

System.out.println("Balance after withdrawal: "

+ total);

try {

// Making thread t osleep for 1 second

Thread.sleep(1000);

}

catch (InterruptedException e) {

e.printStackTrace();

}

}

else {

// Print statement

System.out.println(name+ " you can not withdraw "+ withdrawal);

System.out.println("your balance is: " + total);

try {

Thread.sleep(1000);

}

catch (InterruptedException e) {

e.printStackTrace();

}

}

}

void deposit(String name, int deposit)

{

System.out.println(name + " deposited " + deposit);

total = total + deposit;

System.out.println("Balance after deposit: "

+ total);

try {

Thread.sleep(1000);

}

catch (InterruptedException e) {

e.printStackTrace();

}

}

}

class ThreadBank

{

public static void main(String[] args)

{

Bank obj = new Bank();

// Custom input - Transactions

obj.withdrawn("muskan", 20);

obj.withdrawn("namarta", 40);

obj.deposit("pallavi", 35);

obj.withdrawn("Riya", 80);

obj.withdrawn("shubhi", 40);

}

}

27. import java.io.\*;

// The ThreadJoin class is the child class of the class Thread

class ThreadJoin extends Thread

{

// overriding the run method

public void run()

{

for (int j = 0; j < 2; j++)

{

try

{

// sleeping the thread for 300 milli seconds

Thread.sleep(300);

System.out.println("The current thread name is: " + Thread.currentThread().getName());

}

// catch block for catching the raised exception

catch(Exception e)

{

System.out.println("The exception has been caught: " + e);

}

System.out.println( j );

}

}

}

public class ThreadJoinExample

{

// main method

public static void main (String argvs[])

{

// creating 3 threads

ThreadJoin th1 = new ThreadJoin();

ThreadJoin th2 = new ThreadJoin();

ThreadJoin th3 = new ThreadJoin();

th1.start();

try

{

System.out.println("The current thread name is: "+ Thread.currentThread().getName());

th1.join();

}

catch(Exception e)

{

System.out.println("The exception has been caught " + e);

}

th2.start();

try

{

System.out.println("The current thread name is: " + Thread.currentThread().getName());

th2.join();

}

catch(Exception e)

{

System.out.println("The exception has been caught " + e);

}

th3.start();

}

}

28. class ABC implements Runnable

{

public void run()

{

try

{

Thread.sleep(100);

}

catch (InterruptedException ie)

{

ie.printStackTrace();

}

System.out.println("The state of thread t1 while it invoked the method join() on thread t2 -"+ ThreadState.t1.getState());

try

{

Thread.sleep(200);

}

catch (InterruptedException ie)

{

ie.printStackTrace();

}

}

}

public class ThreadState implements Runnable

{

public static Thread t1;

public static ThreadState obj;

public static void main(String argvs[])

{

obj = new ThreadState();

t1 = new Thread(obj);

System.out.println("The state of thread t1 after spawning it - " + t1.getState());

t1.start();

System.out.println("The state of thread t1 after invoking the method start() on it - " + t1.getState());

}

public void run()

{

ABC myObj = new ABC();

Thread t2 = new Thread(myObj);

System.out.println("The state of thread t2 after spawning it - "+ t2.getState());

t2.start();

System.out.println("the state of thread t2 after calling the method start() on it - " + t2.getState());

try

{

Thread.sleep(200);

}

catch (InterruptedException ie)

{

ie.printStackTrace();

}

System.out.println("The state of thread t2 after invoking the method sleep() on it - "+ t2.getState() );

try

{

t2.join();

}

catch (InterruptedException ie)

{

ie.printStackTrace();

}

System.out.println("The state of thread t2 when it has completed it's execution - " + t2.getState());

}

}

29. public class WaitNotifyTest {

private static final long SLEEP\_INTERVAL = 3000;

private boolean running = true;

private Thread thread;

public void start() {

print("Inside start() method");

thread = new Thread(new Runnable() {

public void run() {

print("Inside run() method");

try {

Thread.sleep(SLEEP\_INTERVAL);

} catch(InterruptedException e) {

Thread.currentThread().interrupt();

}

synchronized(WaitNotifyTest.this) {

running = false;

WaitNotifyTest.this.notify();

}

}

});

thread.start();

}

public void join() throws InterruptedException {

print("Inside join() method");

synchronized(this) {

while(running) {

print("Waiting for the peer thread to finish.");

wait(); //waiting, not running

}

print("Peer thread finished.");

}

}

private void print(String s) {

System.out.println(s);

}

public static void main(String[] args) throws InterruptedException {

WaitNotifyTest test = new WaitNotifyTest();

test.start();

test.join();

}

}

30. class MultithreadingDemo extends Thread {

public void run()

{

try {

// Displaying the thread that is running

System.out.println(

"Thread " + Thread.currentThread().getId()

+ " is running");

}

catch (Exception e) {

// Throwing an exception

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

}

}

}

public class Multithread {

public static void main(String[] args)

{

int n = 8; // Number of threads

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

MultithreadingDemo object

= new MultithreadingDemo();

object.start();

}

}

}



