Assignment 1

create a multi-threaded application by using Thread pool.create 2 threads. each thread should display characters from A to J. [ make sure while one thread executes , other thread should not interfere ]

import java.util.concurrent.\*;

public class Demo implements Runnable{

synchronized public void disp() {

for(char ch='A';ch<='J';ch++) {

System.***out***.println(ch);

}

try {

Thread.*sleep*(200);

} catch (InterruptedException e) {

e.printStackTrace();

}

System.***out***.println();

}

public void run() {

disp();

}

public static void main(String[] args) {

ExecutorService ex = Executors.*newFixedThreadPool*(2);

System.***out***.println("Submitting tasks");

Demo d = new Demo();

ex.submit(d);

ex.submit(d);

System.***out***.println("Shutting down executor service");

ex.shutdown();

}

}

Assignment 2

create a multi-threaded application by using Thread pool.create 2 threads. each thread should display numbers from 1 to 10. [ make sure while one thread executes , other thread should not interfere ]

Also when first thread displays nos. from 1 to 5 it should released the lock and allow other thread to display nos. from 1 to 5 and then previous thread will display nos. from 6 to 10 and so on.

import java.util.concurrent.\*;

public class Demo implements Runnable{

static Class *c*;

void disp() {

synchronized(*c*) {

for(int i=1;i<11;i++) {

if(i==6) {

try {

*c*.wait();

}

catch (InterruptedException e) {

e.printStackTrace();

}

}

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

*c*.notifyAll();

}

}

}

public void run() {

disp();

}

public static void main(String[] args) {

*c* = Demo.class;

ExecutorService ex = Executors.*newFixedThreadPool*(2);

System.***out***.println("Submitting tasks");

ex.submit(new Demo());

ex.submit(new Demo());

System.***out***.println("Shutting down the executor service");

ex.shutdown();

}

}

Assignment 4

create a multi-threaded application by using Thread pool and ReentrantLock (explicit locking) .create 2 threads. each thread should display characters from A to J. [ make sure while one thread executes , other thread should not interfere ]

import java.util.concurrent.\*;

import java.util.concurrent.locks.\*;

public class Demo implements Runnable{

ReentrantLock mylock = new ReentrantLock();

public void disp() {

mylock.lock();

for(char ch = 'A';ch<='J';ch++) {

System.***out***.print(ch+" ");

try {

Thread.*sleep*(100);

}

catch(Exception e) {}

}

mylock.unlock();

System.***out***.println();

}

public void run() {

disp();

}

public static void main(String[] args) {

ExecutorService ex = Executors.*newFixedThreadPool*(2);

Demo d = new Demo();

ex.submit(d);

ex.submit(d);

ex.shutdown();

}

}

Assignment 5

create a multi-threaded application by using Thread pool and ReentrantLock. create 2 threads. each thread should display numbers from 1 to 10. [ make sure while one thread executes, other thread should not interfere]

Also, when first thread displays nos. from 1 to 5 it should release the lock and allow other thread to display nos. from 1 to 5 and then previous thread will display nos. from 6 to 10 and so on.

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

import java.util.concurrent.locks.Condition;

import java.util.concurrent.locks.ReentrantLock;

class NumberPrinter implements Runnable {

private final String threadName;

private final ReentrantLock lock;

private final Condition condition;

private final int threadId; // 0 or 1

private static int *turn* = 0; // shared state

public NumberPrinter(String threadName, int threadId,

ReentrantLock lock, Condition condition) {

this.threadName = threadName;

this.threadId = threadId;

this.lock = lock;

this.condition = condition;

}

*@Override*

public void run() {

try {

// First half: 1–5

lock.lock();

while (*turn* != threadId) {

condition.await();

}

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

System.***out***.println(threadName + " -> " + i);

Thread.*sleep*(200);

}

*turn* = 1 - threadId; // give turn to other thread

condition.signalAll();

lock.unlock();

// Second half: 6–10

lock.lock();

while (*turn* != threadId) {

condition.await();

}

for (int i = 6; i <= 10; i++) {

System.***out***.println(threadName + " -> " + i);

Thread.*sleep*(200);

}

*turn* = 1 - threadId;

condition.signalAll();

} catch (InterruptedException e) {

Thread.*currentThread*().interrupt();

} finally {

lock.unlock();

}

}

}

public class ThreadPoolReentrantLockDemo {

public static void main(String[] args) {

ReentrantLock lock = new ReentrantLock();

Condition condition = lock.newCondition();

ExecutorService executor = Executors.*newFixedThreadPool*(2);

Runnable task1 = new NumberPrinter("Thread-1", 0, lock, condition);

Runnable task2 = new NumberPrinter("Thread-2", 1, lock, condition);

executor.submit(task1);

executor.submit(task2);

executor.shutdown();

}

}