**ÖRNEKLER**

**=====================================================================**

**package** deadlock;

**public** **class** Kilitlenme {

**public** **static** **void** main(String[] args) {

// İki kaynak

**final** Object resource1 = "resource1";

**final** Object resource2 = "resource2";

// kaynak1 daha sonra kaynak2

Thread t1 = **new** Thread() {

**public** **void** run() {

// kaynak 1 kitle

**synchronized**(resource1) {

System.*out*.println("Thread 1: kaynak1 kilitlendi");

**try** { Thread.*sleep*(50); } **catch** (InterruptedException e) {}

**synchronized**(resource2) {

System.*out*.println("Thread 1: kaynak 2 kilitlendi");

}

}

}

};

Thread t2 = **new** Thread() {

**public** **void** run() {

**synchronized**(resource2) {

System.*out*.println("Thread 2:kaynak2 kilitlendi");

**try** { Thread.*sleep*(50); } **catch** (InterruptedException e) {}

**synchronized**(resource1) {

System.*out*.println("Thread 2:kaynak1 kilitlendi");

}

}

}

};

t1.start();

t2.start();

}

}

**=====================================================================**

**package** deadlock;

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

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

**public** **class** ThreadBirliktelik {

**private** **static** Account *account* = **new** Account();

**public** **static** **void** main(String[] args) {

// thread havuzu oluştur

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

executor.execute(**new** Depola());

executor.execute(**new** GeriCek());

executor.shutdown();

System.*out*.println("Thread 1\t\tThread 2\t\tDenge");

}

// hesaba thread ekle

**public** **static** **class** Depola **implements** Runnable {

**public** **void** run() {

**try** { // ertele

**while** (**true**) {

*account*.deposit((**int**)(Math.*random*() \* 10) + 1);

Thread.*sleep*(1000);

}

}

**catch** (InterruptedException ex) {

ex.printStackTrace();

}

}

}

// miktardan çıkar

**public** **static** **class** GeriCek **implements** Runnable {

**public** **void** run() {

**while** (**true**) {

*account*.withdraw((**int**)(Math.*random*() \* 10) + 1);

}

}

}

// iç sınıf

**private** **static** **class** Account {

//kilitlenme

**private** **static** Lock *lock* = **new** ReentrantLock();

// durum oluştur

**private** **static** Condition *newDeposit* = *lock*.newCondition();

**private** **int** balance = 0;

**public** **int** getBalance() {

**return** balance;

}

**public** **void** withdraw(**int** amount) {

*lock*.lock(); // kilitle

**try** {

**while** (balance < amount)

*newDeposit*.await();

balance -= amount;

System.*out*.println("\t\t\tCekilen " + amount +

"\t\t" + getBalance());

}

**catch** (InterruptedException ex) {

ex.printStackTrace();

}

**finally** {

*lock*.unlock(); // aç

}

}

**public** **void** deposit(**int** amount) {

*lock*.lock(); // kilitle

**try** {

balance += amount;

System.*out*.println("depolanan " + amount +

"\t\t\t\t\t" + getBalance());

// durumda bekler

*newDeposit*.signalAll();

}

**finally** {

*lock*.unlock(); // serbest bırak

}

}

}

}

depolanan 1 1

Thread 1 Thread 2 Denge

depolanan 4 5

depolanan 9 14

Cekilen 7 7

Cekilen 1 6

depolanan 6 12

Cekilen 9 3

depolanan 5 8

Cekilen 6 2

depolanan 3 5

depolanan 5 10

Cekilen 10 0

depolanan 1 1

**=====================================**

**package** ureticiTuketici;

**public** **class** kubDelik {

**private** **int** contents;

**private** **boolean** available = **false**;

**public** **synchronized** **int** get() {

**while** (available == **false**) {

**try** {

wait();

} **catch** (InterruptedException e) { }

}

available = **false**;

notifyAll();

**return** contents;

}

**public** **synchronized** **void** put(**int** value) {

**while** (available == **true**) {

**try** {

wait();

} **catch** (InterruptedException e) { }

}

contents = value;

available = **true**;

notifyAll();

}

}

**--------------------------------------**

**package** ureticiTuketici;

**public** **class** ureticiTuketici {

**public** **static** **void** main(String[] args) {

kubDelik c = **new** kubDelik();

uretici p1 = **new** uretici(c, 1);

tuketici c1 = **new** tuketici(c, 1);

p1.start();

c1.start();

}

}

**----------------------------------------**

**package** ureticiTuketici;

**public** **class** tuketici **extends** Thread {

**private** kubDelik kubDelik;

**private** **int** number;

**public** tuketici(kubDelik c, **int** number) {

kubDelik = c;

**this**.number = number;

}

**public** **void** run() {

**int** value = 0;

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

value = kubDelik.get();

System.*out*.println("tuketici #" + **this**.number

+ " got: " + value);

}

}

}

**------------------------------------**

**package** ureticiTuketici;

**public** **class** uretici **extends** Thread {

**private** kubDelik kubDelik;

**private** **int** number;

**public** uretici(kubDelik c, **int** number) {

kubDelik = c;

**this**.number = number;

}

**public** **void** run() {

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

kubDelik.put(i);

System.*out*.println("uretici #" + **this**.number

+ " put: " + i);

**try** {

*sleep*((**int**)(Math.*random*() \* 100));

} **catch** (InterruptedException e) { }

}

}

}

uretici #1 put: 0

tuketici #1 got: 0

uretici #1 put: 1

tuketici #1 got: 1

uretici #1 put: 2

tuketici #1 got: 2

uretici #1 put: 3

tuketici #1 got: 3

uretici #1 put: 4

tuketici #1 got: 4

uretici #1 put: 5

tuketici #1 got: 5

uretici #1 put: 6

tuketici #1 got: 6

uretici #1 put: 7

tuketici #1 got: 7

uretici #1 put: 8

tuketici #1 got: 8

uretici #1 put: 9

tuketici #1 got: 9

**=====================================**

In a multithreaded Java process , it is often needed a communication between different threads. This inter-thread  communication is performing using **wait()**, **notify()** , and **notifyAll(**) methods of **Java.Lang.Object** class. The idea is going to be clear once we do an example which has two child  threads  and those  two child  threads are communicating with each other.

**package** processIletisim2;

**public** **class** ciftIslem **implements** Runnable{

**private** **int** number = 2;

**private** Object shared = **null**;

**public** ciftIslem(Object object) {

shared = object;

}

**public** **void** run() {

**while** (number < 50) {

**synchronized** (shared) {

System.*out*.println("Cift sayi = " + number);

number = number + 2;

**try** {

Thread.*sleep*(500); //only to view sequence of execution

shared.notify();

shared.wait();

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

}

}

}

**package** processIletisim2;

**public** **class** tekIslem **implements** Runnable {

**int** oddNumber = 1;

**private** Object shared = **null**;

**public** tekIslem(Object object) {

shared = object;

}

**public** **static** **void** main(String[] args) {

Object shared = **new** Object();

ciftIslem nesne1 = **new** ciftIslem(shared);

tekIslem nesne2 = **new** tekIslem(shared);

Thread tekThread = **new** Thread(nesne1, "tekThread");

Thread ciftThread = **new** Thread(nesne2, "ciftThread");

tekThread.start();

ciftThread.start();

}

**public** **void** run() {

**while** (oddNumber < 50) {

**synchronized** (shared) {

System.*out*.println("Tek sayi = " + oddNumber);

oddNumber = oddNumber + 2;

**try** {

Thread.*sleep*(500); // only to view the sequence of execution

shared.notify();

shared.wait();

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

}

}

}

================================================

**================================================**

**===========================================**

**package processIletisim3;**

**import java.io.\*;**

**import java.util.concurrent.LinkedBlockingQueue;**

**class ClientMessageHandler implements Runnable {**

**public void run() {**

**while (Test.keepRunning) {**

**try {**

**String msg = Test.msgBuf.take();**

**System.out.println("Incelenen " + msg);**

**} catch (InterruptedException ie) {**

**}**

**}**

**}**

**}**

**=**

**package processIletisim3;**

**import java.io.\*;**

**import java.util.concurrent.LinkedBlockingQueue;**

**class ServerMessageHandler implements Runnable {**

**public void run() {**

**BufferedReader br = new BufferedReader(new InputStreamReader(System.in));**

**String in;**

**try {**

**while (!(in = br.readLine()).equals("quit")) {**

**System.out.println("Gelen " + in);**

**Test.msgBuf.offer(in);**

**}**

**} catch (IOException e) {**

**}**

**Test.keepRunning = false;**

**Test.thread2.interrupt();**

**}**

**}**

**package processIletisim3;**

**import java.io.\*;**

**import java.util.concurrent.LinkedBlockingQueue;**

**public class Test {**

**static LinkedBlockingQueue<String> msgBuf = new LinkedBlockingQueue<String>();**

**static volatile boolean keepRunning = true;**

**static Thread thread1, thread2;**

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

**ClientMessageHandler clientMessagehandler = new ClientMessageHandler();**

**ServerMessageHandler serverMessagehandler = new ServerMessageHandler();**

**thread1 = new Thread(serverMessagehandler);**

**thread2 = new Thread(clientMessagehandler);**

**thread2.start();**

**thread1.start();**

**}**

**}**