|  |  |
| --- | --- |
| H:\LOGO-NXV\Hai__090908__02_1_den.jpg | Faculty of Information Technology  H A N O I U N I V E R S I T Y |

61FIT3NPR – Network Programming   
Tutorial week 4

Java Threads

1. **Part 1:**
2. Exercise 1: Create a thread by extending Thread

**package** tut4;

**public** **class** CountDownThread **extends** Thread {

@Override

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

**int** count = 10;

**for** (**int** i = count; i > 0; i--) {

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

**try** {

Thread.*sleep*(1000);

} **catch** (InterruptedException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

System.***out***.println("Time is up");

}

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

CountDownThread countDownThread = **new** CountDownThread();

countDownThread.start();

}

}

1. Exercise 2: Create a thread by implementing Runnable

**package** tut4;

**public** **class** CountDownThread2 **implements** Runnable {

@Override

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

**int** count = 10;

**for** (**int** i = count; i > 0; i--) {

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

**try** {

Thread.*sleep*(1000);

} **catch** (InterruptedException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

System.***out***.println("Time is up");

}

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

CountDownThread2 countDownThread = **new** CountDownThread2();

Thread thread = **new** Thread(countDownThread);

thread.start();

}

}

1. **HelloMain thread and HelloThread**

**package** tut4;

**public** **class** HelloMain {

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

**int** idx = 1;

**for** (**int** i = 0; i < 2; i++) {

System.***out***.println("Main thread running " + idx++);

// Sleep 2101 milliseconds.

Thread.*sleep*(2101);

}

HelloThread helloThread = **new** HelloThread();

// Run thread

helloThread.start();

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

System.***out***.println("Main thread running " + idx++);

// Sleep 2101 milliseconds.

Thread.*sleep*(2101);

}

System.***out***.println("==> Main thread stopped");

}

}

**package** tut4;

**public** **class** HelloThread **extends** Thread {

@Override

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

**int** index = 1;

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

System.***out***.println(" - HelloThread running " + index++);

**try** {

// Sleep 1030 milliseconds.

Thread.*sleep*(1030);

} **catch** (InterruptedException e) {

}

}

System.***out***.println(" - ==> HelloThread stopped");

}

}

1. Exercise 4:

**class** A **extends** Thread

{

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

{

System.***out***.println("Thread A started");

**for**(**int** i=1;i<=4;i++)

{

System.***out***.println("\t From ThreadA: i= "+i);

}

System.***out***.println("Exit from A");

}

}

**class** B **extends** Thread

{

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

{

System.***out***.println("Thread B started");

**for**(**int** j=1;j<=4;j++)

{

System.***out***.println("\t From ThreadB: j= "+j);

}

System.***out***.println("Exit from B");

}

}

**class** C **extends** Thread

{

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

{

System.***out***.println("Thread C started");

**for**(**int** k=1;k<=4;k++)

{

System.***out***.println("\t From ThreadC: k= "+k);

}

System.***out***.println("Exit from C");

}

}

**class** ThreadTest

{

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

{

**new** A().start();

**new** B().start();

**new** C().start();

}

}

1. **Thread priority**

**class** A **extends** Thread

{

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

{

System.***out***.println("Thread A started");

**for**(**int** i=1;i<=4;i++)

{

System.***out***.println("\t From ThreadA: i= "+i);

}

System.***out***.println("Exit from A");

}

}

**class** B **extends** Thread

{

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

{

System.***out***.println("Thread B started");

**for**(**int** j=1;j<=4;j++)

{

System.***out***.println("\t From ThreadB: j= "+j);

}

System.***out***.println("Exit from B");

}

}

**class** C **extends** Thread

{

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

{

System.***out***.println("Thread C started");

**for**(**int** k=1;k<=4;k++)

{

System.***out***.println("\t From ThreadC: k= "+k);

}

System.***out***.println("Exit from C");

}

}

**class** ThreadPriority

{

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

{

A threadA=**new** A();

B threadB=**new** B();

C threadC=**new** C();

threadC.setPriority(Thread.***MAX\_PRIORITY***);

threadB.setPriority(threadA.getPriority()+1);

threadA.setPriority(Thread.***MIN\_PRIORITY***);

System.***out***.println("Started Thread A");

threadA.start();

System.***out***.println("Started Thread B");

threadB.start();

System.***out***.println("Started Thread C");

threadC.start();

System.***out***.println("End of main thread");

}

}

1. **Part 2: Thread synchronization**

Let’s consider the example below: There is 20 000 000 in one bank account. Almost at one moment, the husband and the wife withdraw from that same bank account: husband withdraws 20 000 000, wife withdraws 15 000 000. If we don’t synchronize threads, both of them do it successfully.

**public** **class** BankAccount {

**long** amount = 20000000;

**public** **boolean** checkAccountBalance(**long** withDrawAmount) {

**try** {

Thread.*sleep*(2000);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

**if** (withDrawAmount <= amount) {

**return** **true**;

}

**return** **false**;

}

**public** **void** withdraw(String threadName, **long** withdrawAmount) {

System.out.println(threadName + " sees balance: " + amount );

System.out.println(threadName + " wants to withdraw: " + withdrawAmount);

**if** (checkAccountBalance(withdrawAmount)) {

**try** {

Thread.*sleep*(2000);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

amount -= withdrawAmount;

System.out.println(threadName + " withdrew successfully: " + withdrawAmount);

System.out.println(threadName + " sees balance: " + amount);

}

//System.***out***.println(threadName + " sees balance: " + amount);

}

}

**public** **class** WithdrawThread **extends** Thread {

String threadName = "";

**long** withdrawAmount = 0;

BankAccount bankAccount;

**public** WithdrawThread(String threadName, BankAccount bankAccount, **long** withdrawAmount) {

**this**.threadName = threadName;

**this**.bankAccount = bankAccount;

**this**.withdrawAmount = withdrawAmount;

}

@Override

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

bankAccount.withdraw(threadName, withdrawAmount);

}

}

**public** **class** MainThread {

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

BankAccount bankAccount = **new** BankAccount();

WithdrawThread husbandThread = **new** WithdrawThread("Husband", bankAccount, 15000000);

husbandThread.start();

WithdrawThread wifeThread = **new** WithdrawThread("Wife", bankAccount, 20000000);

wifeThread.start();

System.***out***.println("Main Thread Ends.");

}

}

The result:

Main Thread Ends.

Husband sees balance: 20000000

Wife sees balance: 20000000

Husband wants to withdraw: 15000000

Wife wants to withdraw: 20000000

Wife withdrew successfully: 20000000

Wife sees balance: 0

Husband withdrew successfully: 15000000

Husband sees balance: -15000000

Main Thread Ends.

Husband sees balance: 20000000

Husband wants to withdraw: 15000000

Wife sees balance: 20000000

Wife wants to withdraw: 20000000

Wife withdrew successfully: 20000000

Wife sees balance: -15000000

Husband withdrew successfully: 15000000

Husband sees balance: -15000000

**5.1 Mutual Exclusive synchronization by key word “synchronized”:**

**Solution 1: “synchronized” the methods**

//import java.io.\*;

**public** **class** BankAccount {

**long** amount = 20000000;

**public** **synchronized** **boolean** checkAccountBalance(**long** withDrawAmount) {

System.out.println("Account balance: " + amount );

**try** {

Thread.*sleep*(2000);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

**if** (withDrawAmount <= amount) {

**return** **true**;

}

**return** **false**;

}

**public** **synchronized** **void** withdraw(String threadName, **long** withdrawAmount) {

System.***out***.println(threadName + " wants to withdraw: " + withdrawAmount);

**if** (checkAccountBalance(withdrawAmount)) {

**try** {

Thread.*sleep*(2000);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

amount -= withdrawAmount;

System.***out***.println(threadName + " withdrew successfully: " + withdrawAmount);

} **else** {

System.***out***.println(threadName + " withdraws error!");

}

System.***out***.println(threadName + " sees balance: " + amount);

}

}

The result:

Main Thread Ends.

Husband wants to withdraw: 15000000

Account balance: 20000000

Husband withdrew successfully: 15000000

Husband sees balance: 5000000

Wife wants to withdraw: 20000000

Account balance: 5000000

Wife withdraws error!

Wife sees balance: 5000000

**Solution 2: “synchronized” inside the methods.**

**The husband/wife don’t need to wait so long as in the solution 1.**

**public** **class** BankAccount {

**long** amount = 20000000;

**public** **boolean** checkAccountBalance(**long** withDrawAmount) {

**try** {

Thread.*sleep*(2000);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

**if** (withDrawAmount <= amount) {

**return** **true**;

}

**return** **false**;

}

**public** **void** withdraw(String threadName, **long** withdrawAmount) {

System.***out***.println(threadName + " checks: " + withdrawAmount);

**synchronized** (**this**) {

**if** (checkAccountBalance(withdrawAmount)) {

**try** {

Thread.*sleep*(2000);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

amount -= withdrawAmount;

System.***out***.println(threadName + " withdraws successfully: " + withdrawAmount);

} **else** {

System.***out***.println(threadName + " withdraws error!");

}

}

System.***out***.println(threadName + " sees balance: " + amount);

}

}

Result:

Main Thread Ends.

Wife checks: 20000000

Husband checks: 15000000

Wife withdrew successfully: 20000000

Wife sees balance: 0

Husband withdraws error!

Husband sees balance: 0

**5.2 Cooperation synchronization by wait(), notify() and notifyAll():**

Problem:

There is only 5 000 000 in the bank account of a couple. The wife want to withdraw 10 000 000. One day, the husband deposits 5 000 000, the wife can withdraw money succefully right after that.

We need to add 2 methods withdrawWhenBalanceEnough() và deposit() to the class BankAccount.

**public** **class** BankAccount **extends** Object {

**long** amount = 5000000;

**public** **synchronized** **boolean** checkAccountBalance(**long** withDrawAmount) {

**try** {

Thread.*sleep*(2000);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

**if** (withDrawAmount <= amount) {

System.***out***.println("Account balance: "+amount);

**return** **true**;

}

**return** **false**;

}

**public** **synchronized** **void** withdraw(String threadName, **long** withdrawAmount) {

System.***out***.println(threadName + " wants to withdraw: " + withdrawAmount);

**if** (checkAccountBalance(withdrawAmount)) {

**try** {

Thread.*sleep*(2000);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

amount -= withdrawAmount;

System.***out***.println(threadName + " withdraws successfully: " + withdrawAmount);

} **else** {

System.***out***.println(threadName + " withdraws error!");

}

System.***out***.println(threadName + " sees balance: " + amount);

}

**public** **synchronized** **void** withdrawWhenBalanceEnough(String threadName, **long** withdrawAmount) {

System.***out***.println(threadName + " wants to withdraw: " + withdrawAmount);

**while** (!checkAccountBalance(withdrawAmount)) {

System.***out***.println(threadName + " waits for balance enough");

**try** {

wait();

} **catch** (InterruptedException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

**try** {

Thread.*sleep*(2000);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

amount -= withdrawAmount;

System.***out***.println(threadName + " withdrew successfully: " + withdrawAmount);

}

**public** **synchronized** **void** deposit(String threadName, **long** depositAmount) {

System.***out***.println(threadName + " deposits: " + depositAmount);

**try** {

Thread.*sleep*(2000);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

amount += depositAmount;

notify();

}

}

**public** **class** WithdrawThread **extends** Thread {

String threadName = "";

**long** withdrawAmount = 0;

BankAccount bankAccount;

**public** WithdrawThread(String threadName, BankAccount bankAccount, **long** withdrawAmount) {

**this**.threadName = threadName;

**this**.bankAccount = bankAccount;

**this**.withdrawAmount = withdrawAmount;

}

@Override

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

bankAccount.withdrawWhenBalanceEnough(threadName, withdrawAmount);

}

}

**public** **class** DepositThread **extends** Thread {

String threadName = "";

**long** depositAmount = 0;

BankAccount bankAccount;

**public** DepositThread(String threadName, BankAccount bankAccount, **long** depositAmount) {

**this**.threadName = threadName;

**this**.bankAccount = bankAccount;

**this**.depositAmount = depositAmount;

}

@Override

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

bankAccount.deposit(threadName, depositAmount);

}

}

**public** **class** MainThread {

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

BankAccount bankAccount = **new** BankAccount();

WithdrawThread wifeThread = **new** WithdrawThread("Wife", bankAccount, 10000000);

wifeThread.start();

DepositThread husbandThread = **new** DepositThread("Husband", bankAccount, 5000000);

husbandThread.start();

}

}

The result:

Wife wants to withdraw: 10000000

Wife waits for balance enough

Husband deposits: 5000000

Account balance: 10000000

Wife withdrew successfully: 10000000