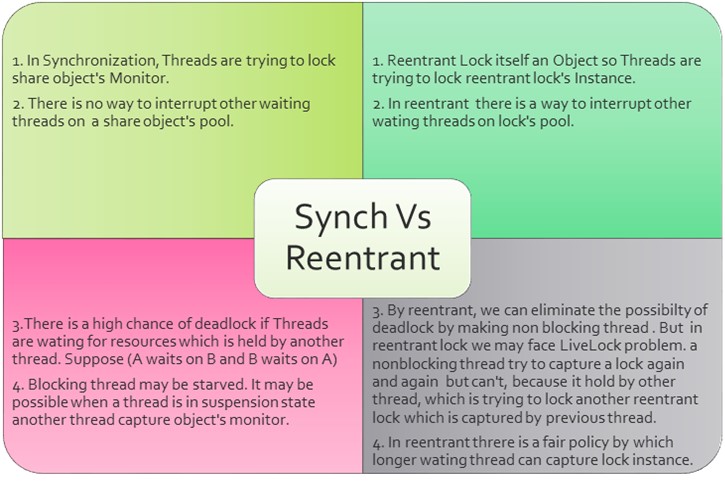
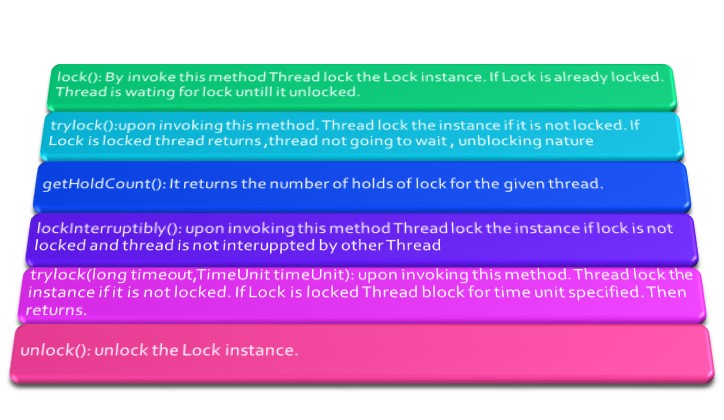
Java Concurrency Reentrant Lock

# Meet Reentrant Lock

In this article, we will discuss about, java 5 inclusion Reentrant Lock. Reentrant Lock someway similar to synchronize block. But it provides much more flexibility than synchronize. I encourage all my readers to use Reentrant lock as it is Extrinsic in nature and developer has much more control over it unlike synchronize.

Benefits of using Reentrant Lock over synchronize.



Reentrant Lock has some important methods. Please see the picture below

Reentrant Lock Working: Once A thread lock reentrant lock , Lock hold count is increase by 1 when same thread release it it’s count decreased by 1.

Example: Usage of Reentrant lock in Transfer money from One Account to Another Account.

Code :

**package** com.example.concurrency.reentrant;

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

**public** **class** Account {

**private** ReentrantLock implicitLock= **new** ReentrantLock();

**private** String name;

**private** Integer balance=10000;

**public** ReentrantLock getImplicitLock() {

**return** implicitLock;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** Integer getBalance() {

**return** balance;

}

**public** **void** setBalance(Integer blance) {

**this**.balance = blance;

}

**public** **boolean** debit(Integer amount)

{

**if**(amount > balance)

{

System.***out***.println(name + " says ::"+ amount + " grater than current balance" );

**return** **false**;

}

balance = balance -amount;

System.***out***.println(name + " says ::"+ amount + " Debited Success Fully" );

**return** **true**;

}

**public** **void** credit(Integer amount)

{

balance = balance +amount;

System.***out***.println(name + " says ::"+ amount + " Credited Success Fully" );

}

}

**package** com.example.concurrency.reentrant;

**import** java.util.concurrent.ExecutorService;

**import** java.util.concurrent.Executors;

**public** **class** AcountTransfer {

**public** **void** transfer(Account from,Account to,Integer amount)

{

**try**

{

from.getImplicitLock().lock();

to.getImplicitLock().lock();

**boolean** flag = from.debit(amount);

**if**(flag)

{

to.credit(amount);

}

System.***out***.println(from.getName() + " says :: now balance is " + from.getBalance());

System.***out***.println(to.getName() + " says :: now balance is " + to.getBalance());

}

**catch**(Exception ex)

{

ex.printStackTrace();

}

**finally**

{

to.getImplicitLock().unlock();

from.getImplicitLock().unlock();

}

}

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

ExecutorService service = Executors.*newFixedThreadPool*(3);

Account from = **new** Account();

from.setBalance(20000);

from.setName("Shamik Mitra");

Account to = **new** Account();

to.setName("Samir Mitra");

AcountTransfer transfer = **new** AcountTransfer();

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

{

service.submit(**new** Runnable(){

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

{

transfer.transfer(from, to, 200);

System.***out***.println(Thread.*currentThread*().getName() +" says :: Transfer successfull");

}

});

}

}

}

Output :

Shamik Mitra says ::200 Debited Success Fully

Samir Mitra says ::200 Credited Success Fully

Shamik Mitra says :: now balance is 19800

Samir Mitra says :: now balance is 10200

Shamik Mitra says ::200 Debited Success Fully

pool-1-thread-1 says :: Transfer successfull

Samir Mitra says ::200 Credited Success Fully

Shamik Mitra says :: now balance is 19600

Samir Mitra says :: now balance is 10400

pool-1-thread-2 says :: Transfer successfull

Shamik Mitra says ::200 Debited Success Fully

Samir Mitra says ::200 Credited Success Fully

Shamik Mitra says :: now balance is 19400

Samir Mitra says :: now balance is 10600

pool-1-thread-3 says :: Transfer successfull

Shamik Mitra says ::200 Debited Success Fully

Samir Mitra says ::200 Credited Success Fully

Shamik Mitra says :: now balance is 19200

Samir Mitra says :: now balance is 10800

pool-1-thread-1 says :: Transfer successfull