Threads: the Stories

Hsuan-Tien Lin

Deptartment of CSIE, NTU

OOP Class, June 9, 2009

Story of a Bank: Part I

Once upon a time, a bank uses the following system to allow customers to spend in local stores easily

```
1        localcredit = getCredit(customer);
2        tospend = getPrice(item);
3        if (tospend <= localcredit){
4          newcredit = localcredit - tospend;
5          notifyNewCredit(newcredit);
6        }
```

Story of a Bank: Part II

Normally,

```
1        localcredit1 = getCredit(customer1); //10000
2        tospend1 = getPrice(item1); //3000
3        if (tospend1 <= localcredit1){
4          newcredit1 = localcredit1 - tospend1; //7000
5          notifyNewCredit(newcredit1);
6        }
7        localcredit2 = getCredit(customer2); //10000
8        tospend2 = getPrice(item2); //2000
9        if (tospend2 <= localcredit2){
10         newcredit2 = localcredit2 - tospend2; //8000
11         notifyNewCredit(newcredit2);
12       }
```

Story of a Bank: Part III

Unfortunately, customer 1 and 2 share the same account but go to different stores

```
1   localcredit1 = getCredit(customer1); //10000
2   localcredit2 = getCredit(customer2); //10000
3   tospend1 = getPrice(item1); //3000
4   if (tospend1 <= localcredit1){
5     newcredit1 = localcredit1 - tospend1; //7000
6     notifyNewCredit(newcredit1);
7   }
8   tospend2 = getPrice(item2); //2000
9   if (tospend2 <= localcredit2){
10    newcredit2 = localcredit2 - tospend2; //8000
11    notifyNewCredit(newcredit2);
12  }
13  getCredit(customer1); //8000
14  getCredit(customer2); //8000
```

Story of a Bank: The End

Local copies are not trustworthy. Must update global copy **atomically**

An Example with Counter Threads

Story of a Couple: Part I

Once upon a time, a couple shares a credit card account. To prevent overdraft, they agreed on the following protocol for using the credit card:

```
1       tospend = getPrice(item);
2       currentlimit = checkCreditbyCellphone();
3       if (tospend <= currentlimit)
4         do_transaction(); //atomically
```

Story of a Couple: Part II

Normally,

```
1        tospend = getPrice(item); //by George: 50000
2        currentlimit = checkCreditbyCellphone(); //60000
3        if (tospend <= currentlimit) //by Mary: yes
4          do_transaction(); //atomically
5        tospend = getPrice(item); //by Mary: 20000
6        currentlimit = checkCreditbyCellphone(); //10000
7        if (tospend <= currentlimit) //by Mary: no
8          do_transaction(); //atomically
```

Story of a Couple: Part III

Unfortunately,

```
1        tospend = getPrice(item); //by George: 50000
2        currentlimit = checkCreditbyCellphone(); //60000
3        //George drives to the store
4        tospend = getPrice(item); //by Mary: 20000
5        currentlimit = checkCreditbyCellphone(); //60000
6        if (tospend <= currentlimit) //by Mary: yes
7          do_transaction();
8        if (tospend <= currentlimit) //by George: yes
9          do_transaction(); //OVERDRAFT!!
```

Story of a Couple: The End

Spent should happen **immediately** after checking

An Example with Couple Threads

Synchronization

`synchronized`: binds operations altogether (with respect to a lock)

- synchronized method: the lock is the class (for static method) or the object (for non-static method)
  - usually used to protect the variables within the class/object
- synchronized block: the lock is explicitly provided
  - flexible, fine-grained use

More on the Lock

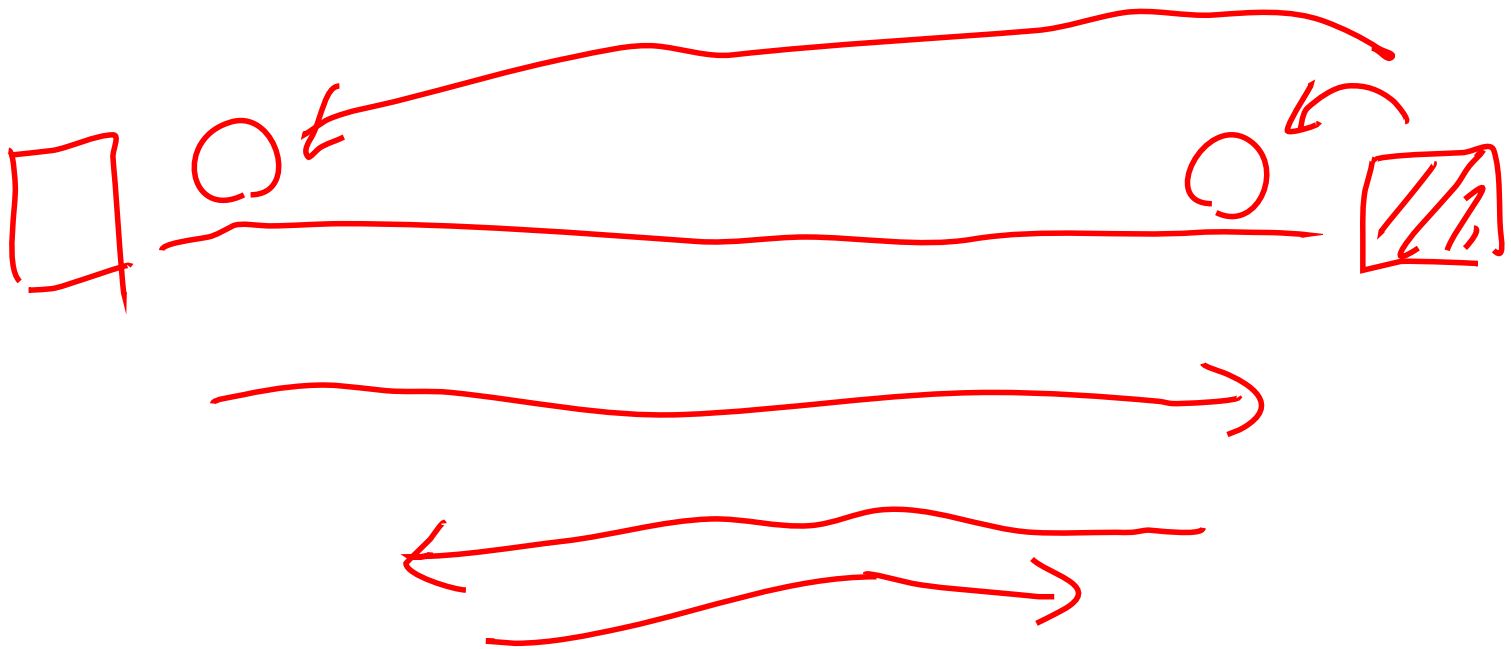
- after getting the lock, can “use” any synchronized method/block that depends on the lock
- lock releases after the method/block finishes (by return or exception)

An Example with Counter Threads

An Example with Couple Threads

A Story of the Black and White Goats: Deadlock

A Story of the Black and White Goats: Starvation



A Story of the Black and White Goats: Livelock

Ways to be Polite

```
1  synchronized void make_payment(int amount){
2    while(no_money()){
3      this.complain(1000);
4    }
5  }
```

```
1  synchronized void make_payment(int amount){
2    while(no_money()){
3      Thread.sleep(1000);
4    }
5  }
6  // ...
```

```
1  synchronized void make_payment(int amount){
2    while(no_money()){
3      this.wait(1000);
4    }
5  }
6  // ...
```

Wait until Notified

```
1   synchronized void make_payment(int amount){
2     while(no_money()){
3       this.wait();
4     }
5   }
6
7   synchronized void get_money(int amount){
8     money += amount;
9     notifyAll();
10  }
```

notify