

Atomicity and Deadlock

November 29, 2007

1

Check-then-Act Race

```
public void transfer (Account fromAccount,  
                    Account toAccount, int amount) {  
    if (fromAccount.getBalance() >= amount) {  
        fromAccount.withdraw (amount);  
        toAccount.deposit(amount);  
    }  
}
```

2

Good Interleaving

Account1	Account2	Transfer 1->2	Transfer 1->2
balance = 100			
	balance=1900		
		balance good	
lock			
balance = 0			
unlock			
	lock		
	balance = 2000		
	unlock		
			balance bad

3

Bad Interleaving

Account1	Account2	Transfer 1->2	Transfer 1->2
balance = 100			
	balance = 1900		
		balance good	
			balance good
lock			
balance = 0			
unlock			
	lock		
	balance = 2000		
	unlock		
lock			
balance = 0			
unlock			
	lock		
	balance = 2100		
	unlock		

4

Atomic Check-and-Act

```
public void transfer (Account fromAccount,
                    Account toAccount, int
                    amount) {
    synchronized (fromAccount) {
        if (fromAccount.getBalance() >= amount) {
            fromAccount.withdraw (amount);
            toAccount.deposit(amount);
        }
    }
}
```

5

Interleaving

Account1	Account2	Transfer 1->2	Transfer 1->2
balance = 100			
	balance = 1900		
lock			
		balance good	
wait for lock			
balance = 0			
	lock		
	balance = 2000		
	unlock		
unlock			
lock			
			balance bad
unlock			

6

Deadlock

- Deadlock occurs when threads are all waiting for locks and no thread can proceed.

```
// Create first thread to repeatedly transfer in one
// direction
Transferrer t1 = new Transferrer(account1, account2);
Thread t1Thread = new Thread(t1);

// Create second thread to repeatedly transfer in
// the other direction.
Transferrer t2 = new Transferrer(account2, account1);
Thread t2Thread = new Thread(t2);

t1Thread.start();
t2Thread.start();
```

7

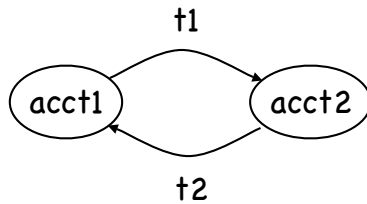
Interleaving with Deadlock

Account1	Account2	Transfer 1->2	Transfer 2->1
balance = 100			
	balance = 1900		
	lock		
			balance good
	balance = 1800		
lock			
		balance good	
balance = 0			
	wait for lock		
wait for lock			

8

What causes Deadlock?

- Threads are holding locks while waiting for other locks
- There is a circular dependency between the holders and the waiters



9

Avoiding Deadlock 1

- Threads are holding locks while waiting for other locks
- Solution: don't hold 2 locks at once.

```
public void transfer (Account fromAccount,
                    Account toAccount, int amount) {
    boolean withdrawWorked = false;
    synchronized (fromAccount) {
        if (fromAccount.getBalance() >= amount) {
            fromAccount.withdraw (amount);
            withdrawWorked = true;
        }
    }
    if (withdrawWorked) {
        toAccount.deposit (amount);
    }
}
```

10

Avoiding Deadlock 2

- There is a circular dependency between the holders and the waiters
- Solution: Make sure threads always acquire locks in the same order.

```
public void transfer (Account fromAccount, Account toAccount, int amount) {
    Account accountToLock;

    if (fromAccount.getAccountNumber() < toAccount.getAccountNumber()) {
        accountToLock = fromAccount;
    }
    else {
        accountToLock = toAccount;
    }
    synchronized(accountToLock) {
        if (fromAccount.getBalance() >= amount) {
            fromAccount.withdraw (amount);
            toAccount.deposit(amount);
        }
    }
}
```

11

Account1	Account2	Transfer 1->2	Transfer 2->1
balance = 100			
	balance = 1900		
lock			balance good
	lock		
	balance = 1800		
wait for lock			
	unlock		
balance = 200			
unlock			
lock			
		balance good	
balance = 100			
	lock		
	balance = 1900		
	unlock		
unlock			

12