PA 1

https://github.com/comp346/W15

Objectives

- Explain the specification
- Multiple threading
- Issues without Synchronization
- Simple solution

Setup

1. Create a new project
2. Extract the provided file
3. Copy java files to your project
4. Run

What are inside?

- Account
- Depositor
- Withdrawer
- AccountManager

Account's Properties

Account Number, Name & Balance. Balance can be changed.

Account's Behaviors

- Open or create account
- Deposit
- Withdraw
- Check balance

Account Skeleton

```
class Account {
    private int accountID;
    private String name;
    private double balance;

    //Skeleton
    public Account(int acc, String name, double balance) { ??? }
    public void deposit(double amount) { ??? }
    public void withdrawn(double amount) { ??? }
    public double getBalance() { ??? }
    public String toString() { ??? }
}
```

Inheritance

Dove can inherit general properties of a bird: fly, eat ...

```
public class Dove extends Bird {

}
```

A bird can inherit from an animal ...

```
public class Bird extends Animal {

}
```

Hierrachy

Every class in java either is inherited directly or indirectly from `Object`

Thread is an execution that executes a **given task** concurrently with other threads.

```
class CustomTask extends Thread {
    public void run { ??? }
}
```

Create a thread then call `start`

```
CustomTask t = new CustomTask();
t.start();
```

Depositor is an execution which executes **deposit N times** to a given account.

```
public class Depositor extends Thread {
    private Account account;

    public Depositor(Account account) {
        this.account = account;
    }

    public void run() {
        for(int i = 0; i < times; i++) {
            account.deposit(amount);
        }
    }
}
```

Withdrawer is an execution which executes **withdraw N times** to a given account.

```
public class Withdrawer extends Thread {
    private Account account;

    public Depositor(Account account) {
        this.account = account;
    }

    public void run() {
        for(int i = 0; i < times; i++) {
            account.withdraw(amount);
        }
    }
}
```

Account Manager

1. Application Entry - has `static void main` method
2. Create 10 accounts
3. Print out balances
4. Each account, create 1 depositor and 1 withdrawer
5. Start all depositors and withdrawers
6. Waits for all finished

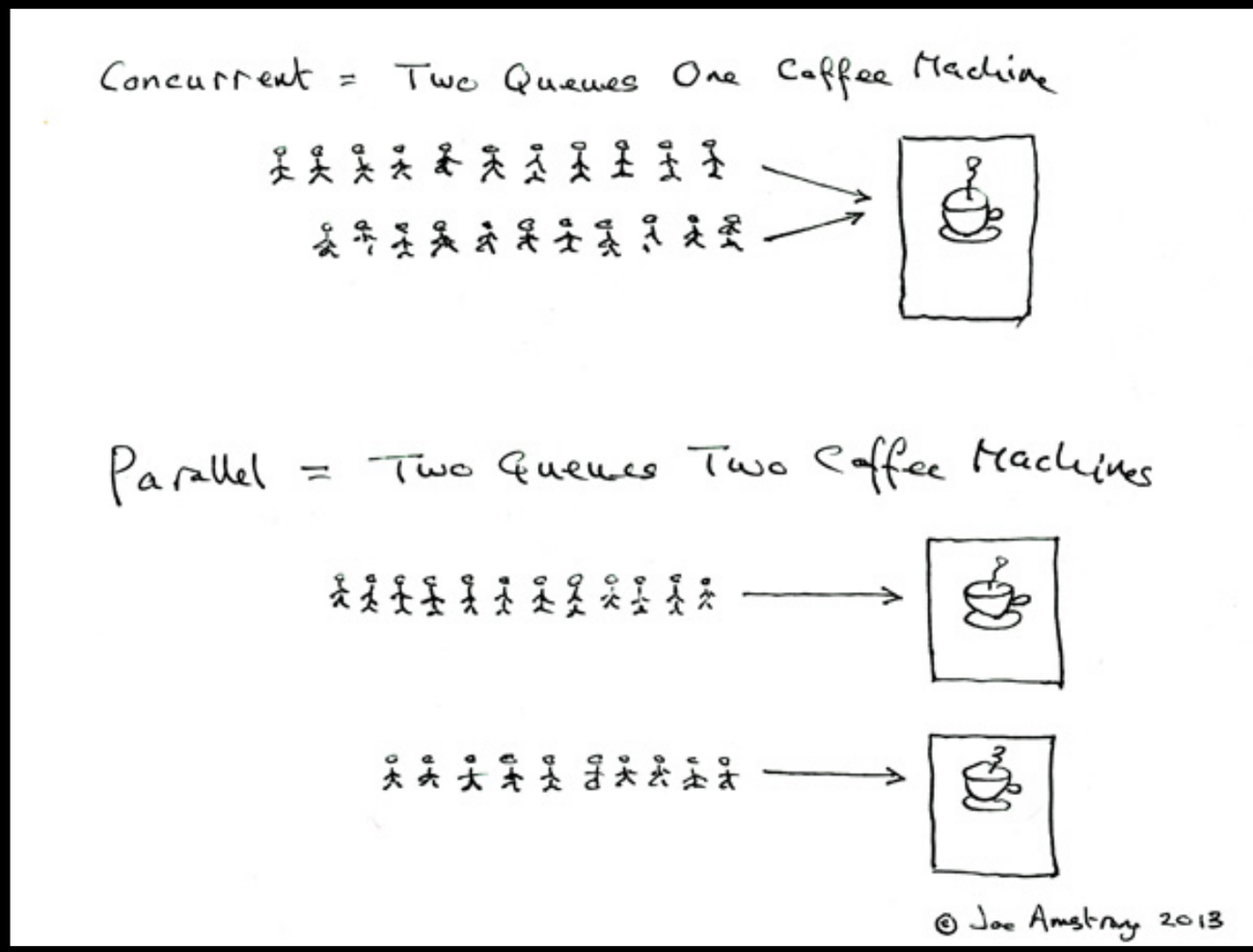
```
public class AccountManager {
    public static void main(String[] args) {
        int n = 10;
        //Create and initialize accounts
        Account[] accounts = new Account[n];
        accounts[0] = new Account(1234, "Mike", 1000);
        accounts[9] = new Account(9999, "Alex", 2000);
        for(int i = 0; i < n; i++){
            System.out.println(accounts[i]);
        }
        //Depositors and withdrawers
        Depositor[] ds = new Depositor[n];
        Withdrawer[] ws = new Withdrawer[n];
        for(int i = 0; i < n; i++) {
            ds[i] = new Depositor(accounts[i]);
            ws[i] = new Withdrawer(accounts[i]);
        }
        //More to come
    }
}
```

```java
public class AccountManager {
    public static void main(String[] args) {
        int n = 10;
        // .. Omitted step 2, 3 and 4
        for(int i = 0; i < n; i++) {
            ds[i].start();
            ws[i].start();
        }
        //Waits for finish
        for(int i = 0; i < n; i++) {
            ds[i].join();
            ws[i].join();
        }
        //Check balance again
        for(int i = 0; i < n; i++){
            System.out.println(accounts[i]);
        }
    }
}
```

Re-run the application

1. Remember the balance of the first account
2. Is it the same as the initialized balance ?
3. Re-run once more, is it the same as the previous steps.

What happens if both lines approach at the same time?



Race condition

```
public class UnsafeSequence {
    private int value = 0;
    public int getNext() {
        return value++;
    }
}
```

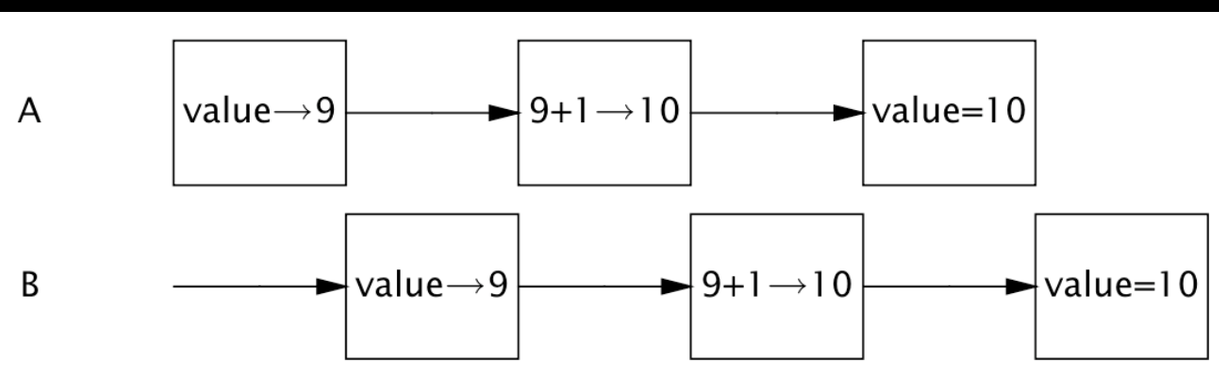


FIGURE 1.1. Unlucky execution of `UnsafeSequence.getNext`.

Synchronized (Intrinsic Lock)

Only one thread is allowed to enter a method or a block of statements.

```
public class SynchronizedSequence {
    private int value;
    public synchronized int getNext() {
        return value++;
    }
}
```

http://docs.oracle.com/javase/tutorial/essential/concurrency/locksync.html

Package

- Group of related classes (Example System.out.println())
- Create package
- Import package

Task & Submission

Please refer the specification

Thank you