Addison Chan

Luc

10/24/16

## Coupling

- Dependency between multiple classes
- Preferably: low coupling rather than high coupling

### Parameter Variable

- Passed in as values, not references, meaning that parameter variables don't get updated. You get a copy of the object, so the actual doesn't get changed.

```
public class BankAccount{
    public double balance;
    // Bad because BankAccount object in BankAccount
    public void transfer(double amount, BankAccount otherAcct){

        // not recommended because amount is modified (not accessible anymore)
        amount = amount + 5

        double amount1 = amount + 5  // recommended because different
        balance = balance - amount;
        double newBalance = otherAcct.balance + amount;
    }
}
BankAccount a = new BankAccount(1000);
BankAccount b = new BankAccount(500);
a.transfer(100,b);
```

## Documentation of Methods

- Preconditions and Postconditions

### Precondition

- Condition before program started
- Requirement that the caller of a method must obey

- Caller (user) of the method is responsible to call the method correctly
- If the method is called improperly anyway, the method is not responsible for producing a correct result
- Always after param (last line)
- Whenever method has parameters, there should be a precondition

```java
/**
 * Deposits money into this account
 * @param amount amount of money to deposit
 * (Precondition: amount >= 0)
 */
public void deposit(double amount){
    balance += amount;
    if (amount < 0)
        break;
}
```

## Postcondition

- When a method is called in accordance to is precondiitons, then the method promises to do its job correctly
- No need for postcondition if it is redundant (see getBalance below)

```java
/**
 * Deposits money into this account
 * (Postcondition: balance >= 0)
 * @param amount amount of money to deposit
 * (Precondition: amount >= 0)
 */
public void deposit(double amount){
}

/**
 * Returns the current of this account
 * (Postcondition: the return value equals to the account balance)
 * ^ extra because repeated
 * @return the account balance
 */
public double getBalance(){
    return balance;
}
```

## Static Methods (Class methods)

- A method that is not invoked on an object

- Simply implement the methods in the class and call it without constructing an object of that class

```
public class Square{
    /**
     * Returns the area of a square with the given side
     * (Postcondition area > 0)
     * @param side the length of the side of a square
     * @return area the area of the square
     * (Precondition side > 0)
     */
    public static double getArea(double side){
        return side*side;
    }
}
```

---

Tester class:

```
public class Tester{
    public static void main(String[] args){

        // not going to compile because no Square object
        System.out.println(Square.getArea(3));

        // changed getArea to static and now it works
        System.out.println(Square.getArea(3));
    }
}
```