Lecture 5 - Sep. 21

Review on OOP

More Advanced Use of this
Static Variables

Announcements

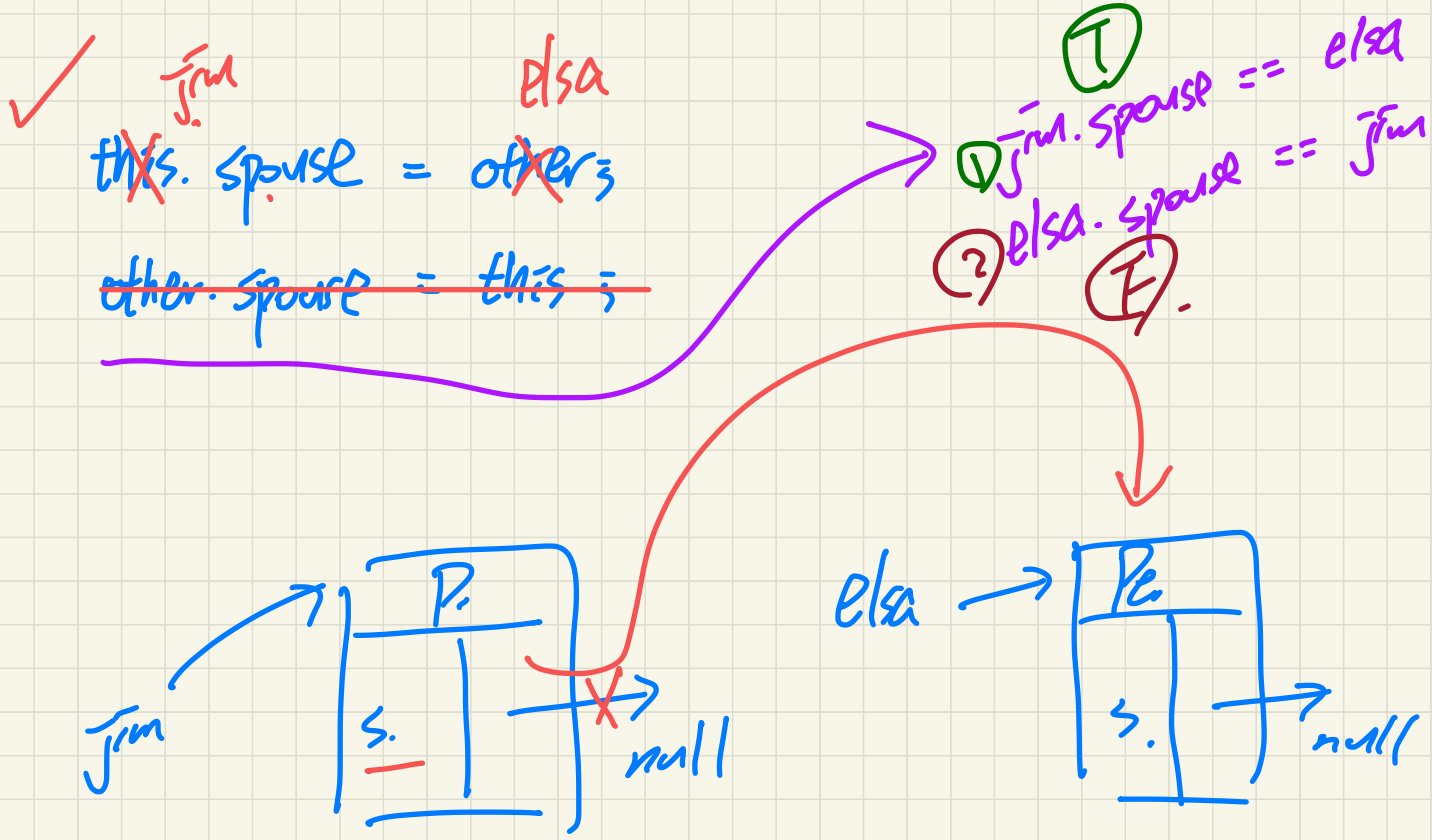
- Lab1 released (scheduled lab sessions & office hours)
- Lab0 Part 2 Due on Friday
- WrittenTest1 → WSC.
  - make sure you try logging into eClass in WSC
  - A guide and some practice questions released soon
- Programming Test 1 (60 to 65 min) → WSC
  - Identical format as Lab1
  - Number of starter tests will be smaller
  - Guide, Practice Test, Mockup Test to be announced

Example: Reference to this

```
public class Person {
  private String name;
  private Person spouse;
  public Person(String name) {
    this.name = name;
  }
  public void marry(Person other) {
    if(this.spouse != null || other.spouse != null) {
      /* Error: both must be single */
    }
    else { this.spouse = other; other.spouse = this; }
  }
}
```

```
Person jim = new Person("Jim");
Person elsa = new Person("Elsa");
jim.marry(elsa);
```

- normal
  - ↳ marry
- abnormal
  - ↳ e.g. can't marry one to themselves
  - ↳ e.g. can't marry someone not single


✓
jim
elsa
this.spouse = other;
other.spouse = this;
① jim.spouse == elsa
② elsa.spouse == jim
T
T
P
s.
null
jim
elsa
P
s.
null

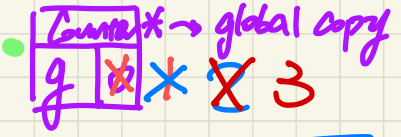
Managing Account IDs: Manual

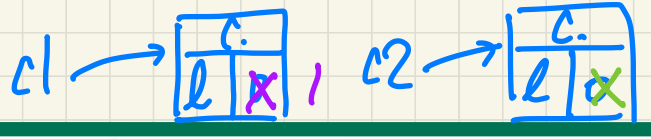

```
public class Account {
  private int id;
  private String owner;
  public int getID() { return this.id; }
  public Account(int id, String owner) {
    this.id = id;
    this.owner = owner;
  }
}
```

```
class AccountTester {
  Account acc1 = new Account(1, "Jim");
  Account acc2 = new Account(2, "Jeremy");
  System.out.println(acc1.getID() != acc2.getID());
}
```

manual management of id's

Declaring Global Variables among Objects





```java
public class Counter {
    private int l;
    static int g = 0;

    public Counter() {
        this.l = 0;
    }

    public int getLocal() {
        return this.l;
    }

    public void incrementLocal() {
        this.l ++;
    }

    public void incrementGlobal() {
        g ++;
    }
}
```

```java
public class CounterTester {
    public static void main(String[] args) {
        Counter c1 = new Counter();
        Counter c2 = new Counter();

        System.out.println("c1's local: " + c1.getLocal());
        System.out.println("c2's local: " + c2.getLocal());
        System.out.println("Global accessed via c1: " + c1.g);
        System.out.println("Global accessed via c2: " + c2.g);
        System.out.println("Global accessed via Counter: " + Counter.g);

        c1.incrementLocal();

        c2.incrementLocal();

        c1.incrementGlobal();

        c2.incrementGlobal();

        Counter.g = Counter.g + 1; // Counter.global ++;
    }
}
```

```java
public class Counter {
    private int l;
    static int g = 0;

    public Counter() {
        this.l = 0;
        g ++;
    }

    public int getLocal() {
        return this.l;
    }

    public void incrementLocal() {
        this.l ++;
    }

    public void incrementGlobal() {
        g ++;
    }
}
```

init. done only once

expanded wherever new Counter obj is created

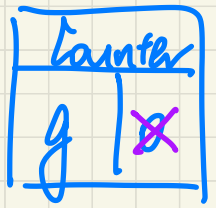
println(Counter.g); 0

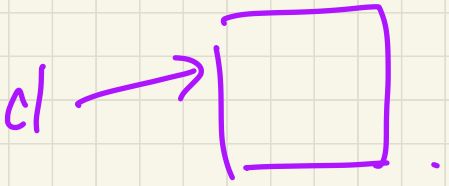
Counter c1 = new Counter(); 1

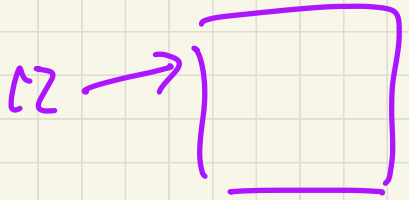
println(Counter.g); 1

Counter c2 = new Counter(); 2

println(Counter.g); 2



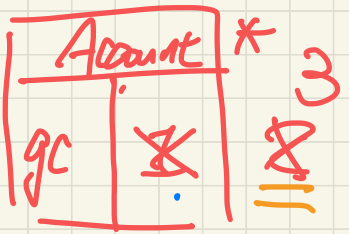


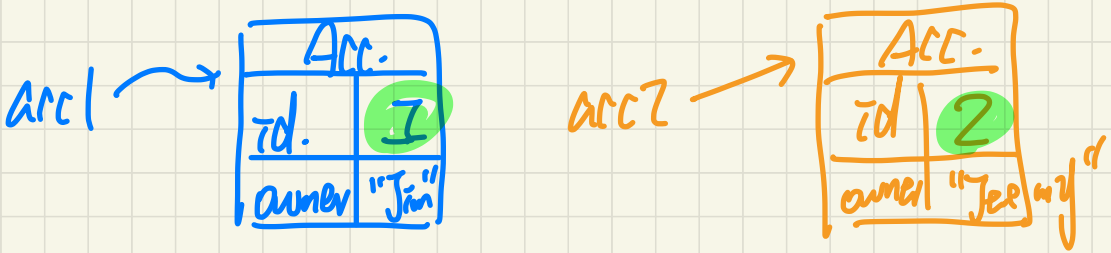


Managing Account IDs: Automatic

```
class Account {
  private static int globalCounter = 1;
  private int id; String owner;
  public Account(String owner) {
    this.id = globalCounter;
    globalCounter ++;
    this.owner = owner; } }
```

```
class AccountTester {
  Account acc1 = new Account("Jim");
  Account acc2 = new Account("Jeremy");
  System.out.println(acc1.getID() != acc2.getID()); }
```





Misuse of Static Variables

```
public class Client {
  private Account[] accounts;
  private static int numberOfAccounts = 0;
  public void addAccount(Account acc) {
    accounts[this.numberOfAccounts] = acc;
    this.numberOfAccounts ++;
  } }
```

```
public class ClientTester {
  Client bill = new Client("Bill");
  Client steve = new Client("Steve");
  Account acc1 = new Account();
  Account acc2 = new Account();
  bill.addAccount(acc1);
    /* [redacted] */
  steve.addAccount(acc2);
    /* [redacted] */
}
```

bill.addAccount(acc3);
steve.addAccount(acc4);

Exercise