

# Lecture 4

## Part F

### ***Classes and Objects - Static Variables***



## non-static variables

int l;

- attribute

- instance-specific: ① each object of the class

- initialized in constructors. has its own copy.

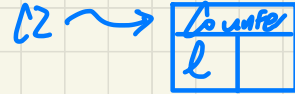
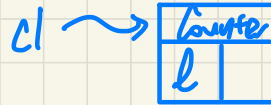
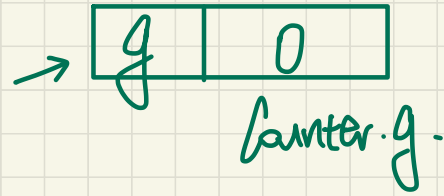
## static variables

static int g = 0;

- instance-independent: ① all objects of the class

share the same copy.

- initialized upon declaration.



② To access it, a context object is necessary. e.g. c1.l.

② To access it, class name suffices.

# Declaring Global Variables among Objects



```
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++;
    }
}
```

initialization.

inst. of non-static variable.

l specific to context obj.

g shared by all Counter instances.

static g already available



c1



c2

```
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();

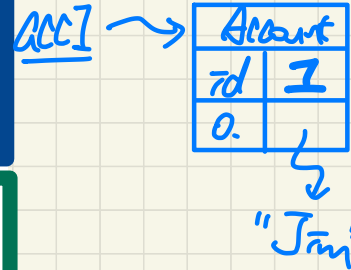
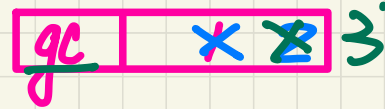
        Counter.g = Counter.g + 1; // Counter.global ++;
    }
}
```

Access to static variables does not require a CO.

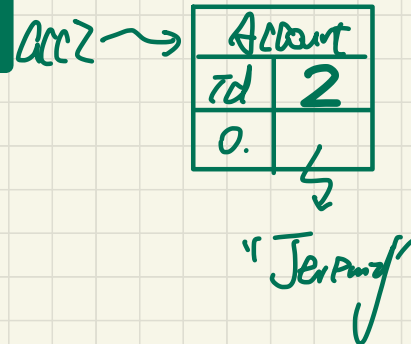
Use of a context object var to access a static var is unnecessary.

# Managing Account IDs: Automatic

```
class Account {  
    private static int globalCounter = 1;  
    private int id; String owner;  
    public Account(String owner) {  
        acc1 this.id = globalCounter; "Jim"  
        acc2 globalCounter++; "Jeremy"  
        this.owner = owner; } }  
acc1  
acc2
```



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



STEP. ACCOUNTS[STEP. NOA] = ACC2;

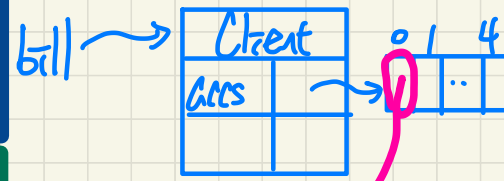
# Misuse of Static Variables

1

NOA ~~1~~ ~~2~~ 2

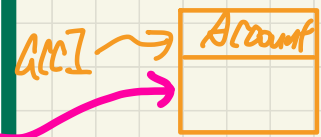
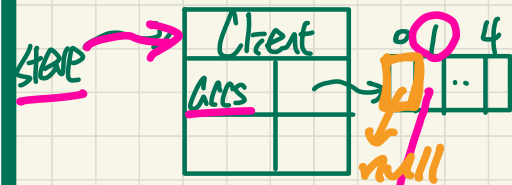
```
public class Client {  
    .private Account[] accounts; ✓  
    .private static int numberOfAccounts = 0;  
    public void addAccount(Account acc) {  
        accounts[this.numberOfAccounts] = acc;  
        this.numberOfAccounts++;  
    }  
}
```

*Annotations:*  
- this (blue arrow)  
- bill (blue arrow)  
- step (green arrow)  
- acc1 (green arrow)  
- acc2 (green arrow)  
- NOA (pink box)  
- acc (green arrow)  
- acc1 (green arrow)  
- acc2 (green arrow)  
- \*\* (green text)



```
public class ClientTester {  
    Client bill = new Client("Bill");  
    Client steve = new Client("Steve");  
    Account acc1 = new Account();  
    Account acc2 = new Account();  
    bill.addAccount(acc1);  
    /* bill bill.getAccounts()[0] */  
    steve.addAccount(acc2);  
    /* mistakenly added to steve.getAccounts()[1]! */  
}
```

*Annotations:*  
- Client bill (blue arrow)  
- Client steve (green arrow)  
- Account acc1 (orange arrow)  
- Account acc2 (orange arrow)  
- bill.addAccount(acc1) (blue arrow)  
- steve.addAccount(acc2) (green arrow)  
- bill (purple box)  
- steve (green box)  
- mistakenly (purple text)



bill.accounts[bill.NOA] = acc1;

# Use of Static Variables: Common Error

weak but poor design.

static (good solution?).

Design

① All bank objects share the same branch name?

② Each bank object has its own instance-specific branch name.

```
1 public class Bank {
2     private String branchName;
3     public String getBranchName() { return this.branchName; }
4     private static int nextAccountNumber = 0;
5     public static String getInfo() {
6         nextAccountNumber++;
7         return this.branchName + nextAccountNumber;
8     }
9 }
```

requires a context object

cannot use non-static variable from a static context.

Contradictory!

To access:  
Bank.getInfo()  
cannot send as a context object.