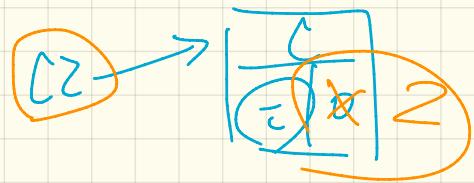
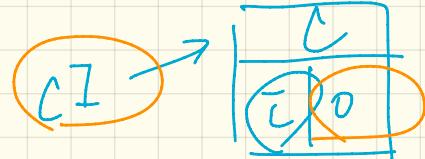


Wednesday March 20
Lecture 20

class C {

int i;

non-static



Static int j;

all objects of type C share a single copy of j.

C.i
C2.i

```

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

        System.out.println("c1's local: " + c1.l);
        System.out.println("c2's local: " + c2.l);
        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 + 1
    }
}

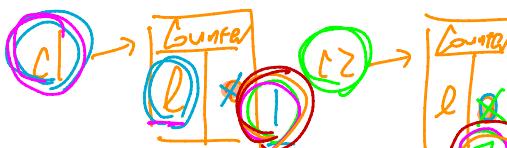
```

Counter.g is static, so it is shared by all objects of the Counter class. It is not a local variable, so it cannot be accessed via individual objects (c1.g, c2.g). It is a global variable, so it can be accessed via the class name (Counter.g).

c1.l and *c2.l* are local variables, so they are unique to each object. They can be accessed via the object name (c1.l, c2.l).

c1.g and *c2.g* are global variables, so they are shared by all objects of the Counter class. They can be accessed via the object name (c1.g, c2.g).

Counter.g is a static global variable, so it is shared by all objects of the Counter class. It can be accessed via the class name (Counter.g).



X *Counter.g* is not Counter object. *Counter.g* is global variable.

✓ *Counter.g* is static global variable.

```

static int g;
public class Counter {
    int l;
    static int g = 0;

    Counter() {
        l = 0;
    }

    void incrementLocal() {
        l++;
        c1.g += 1;
    }

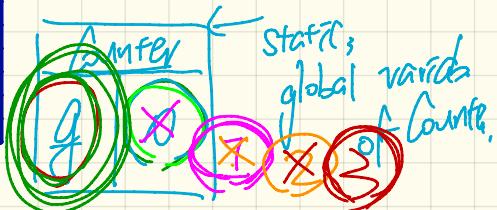
    void incrementGlobal() {
        g++;
    }
}

```

Counter.g is static, so it is shared by all objects of the Counter class. It is not a local variable, so it cannot be accessed via individual objects (c1.g, c2.g). It is a global variable, so it can be accessed via the class name (Counter.g).

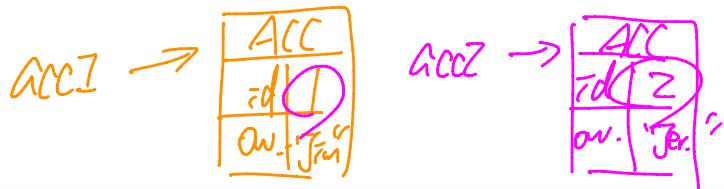
c1.g and *c2.g* are local variables, so they are unique to each object. They can be accessed via the object name (c1.g, c2.g).

Counter.g is a static global variable, so it is shared by all objects of the Counter class. It can be accessed via the class name (Counter.g).



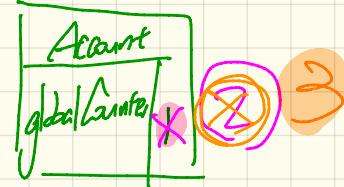
Managing Account ID: Manually

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

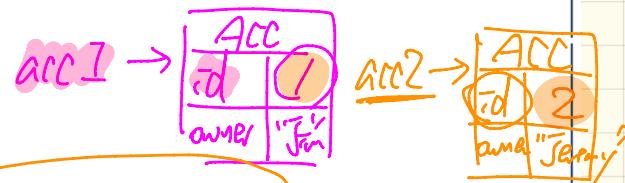


```
class AccountTester {  
    Account acc1 = new Account(1, "Jim");  
    Account acc2 = new Account(2, "Jeremy");  
    System.out.println(acc1.id != acc2.id);  
}
```

Managing Account ID: Automatically



```
class Account {  
    static int globalCounter = 1;  
    int id; String owner;  
    Account(String owner) {  
        this.id = globalCounter; // This line is highlighted in yellow  
        globalCounter++; // This line is highlighted in yellow  
        this.owner = owner; } }
```



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

Misuse of Static Variables

Tutorial:

```
class Client {  
    Account[] accounts;  
    int noa;
```

accounts;

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

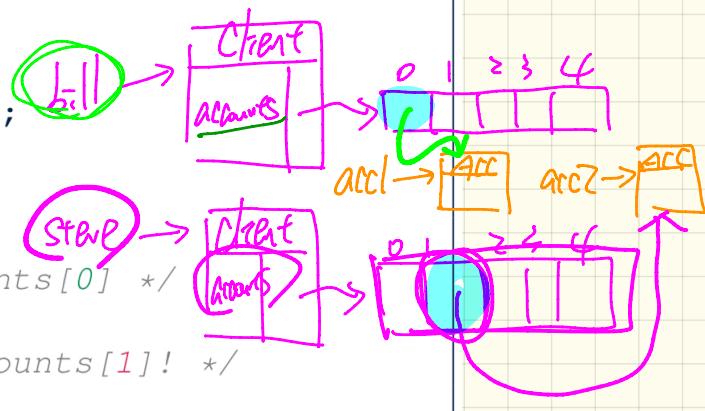
Step. accounts[noa] = acc;
noa++;

7



2

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



```
1 public class Bank {  
2     public string branchName;  
3     public static int nextAccountNumber = 1;  
4     public static void useAccountNumber() {  
5         system.out.println (branchName + ...);  
6         nextAccountNumber++;  
7     }  
8 }
```

non-static

A non-static variable used
in the static context!

II. BranchName

X not a C.O.

Bank. useAccountNumber()

but branchName is non-static,
which requires a C.O.
inconsistent