Lecture 5 - May 20

Review of OOP

Tracing OOP: Arrays and Aliasing
Static vs. Non-Static Variables
Using Static Variables to Manage IDs

Announcements/Reminders

- Today's class: notes template posted
- ProgTest0 this Friday (May 23)
  + Guide (policies & requirements) posted
  + PracticeTest0 posted
- Priorities:
  + Lab 1
  + Review slides on **Classes and Objects**

Anonymous Objects


```
1 double square(double x) {
2   double sqr = x * x;
3   return sqr; }
```

named exp

```
1 double square(double x) {
2   return x * x; }
```

anonymous exp.

```
1 Person getP(String n) {
2   Person p = new Person(n);
3   return p; }
```

named obj.

```
1 Person getP(String n) {
2   return new Person(n); }
```

anonymous obj.

Exercise

```
class Member {
  private Order[] orders;
  private int noo;
  /* constructor ommitted */
  public void addOrder(Order o) {
    this.orders[this.noo] = o;
    this.noo++;
  }
  public void addOrder(String n, double p, double q) {
    Order no = new Order(n, p, q);
    this.orders[this.noo] = no;
    this.noo++;
  }
}
```

used as a helper method.

duplicates "smell" in code!

v1 → helper method call.

v2 → anonymous object passed as arg.

v3

v1. `Order no = new Order(n, p, q);`

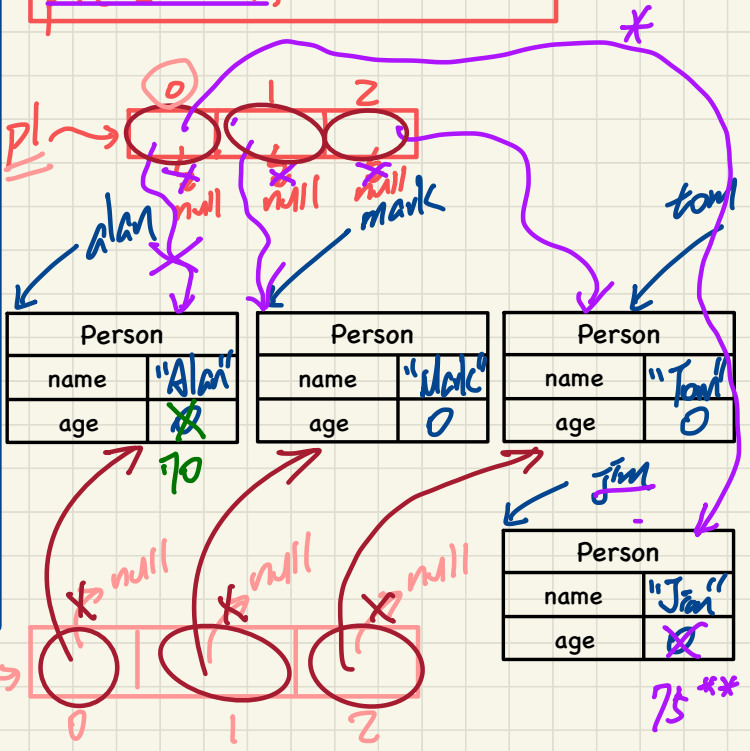
v2. `this.addOrder(no);`

v3. `this.addOrder(new Order(n, p, q))`

Copying Reference Values: Aliasing


```
Person alan = new Person("Alan");
Person mark = new Person("Mark");
Person tom = new Person("Tom");
Person jim = new Person("Jim");
Person[] persons1 = {alan, mark, tom};
Person[] persons2 = new Person[persons1.length];
for(int i = 0; i < persons1.length; i ++) {
  persons2[i] = persons1[i]; }
persons1[0].setAge(70);
System.out.println(jim.getAge());
System.out.println(alan.getAge());
System.out.println(persons2[0].getAge());
persons1[0] = jim;
persons1[0].setAge(75);
System.out.println(jim.getAge());
System.out.println(alan.getAge());
System.out.println(persons2[0].getAge());
```

```
Person[] p1 = new Person[3];
p1[0] = alan;
p1[1] = mark;
p1[2] = tom;
```



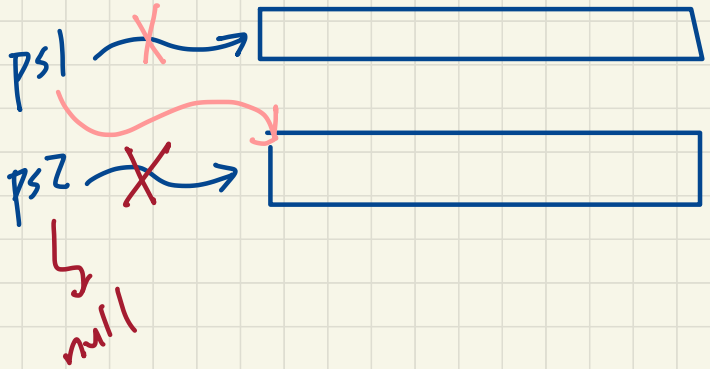

```
It #1: i == 0  p2[0] = p1[0];
It #2: i == 1  p2[1] = p1[1];
It #3: i == 2  p2[2] = p1[2];
```

Re-assigning Array References

```
Person[] ps1 = new ...;
Person[] ps2 = new ...;
```

①

	①	②	③
ps1 == null	F	F	F
ps2 == null	F	F	T
ps1 == ps2	F	T	F



copy the address stored in ps1 into ps2

```
ps1 = ps2;
```

②

```
ps2 = null;
```

③

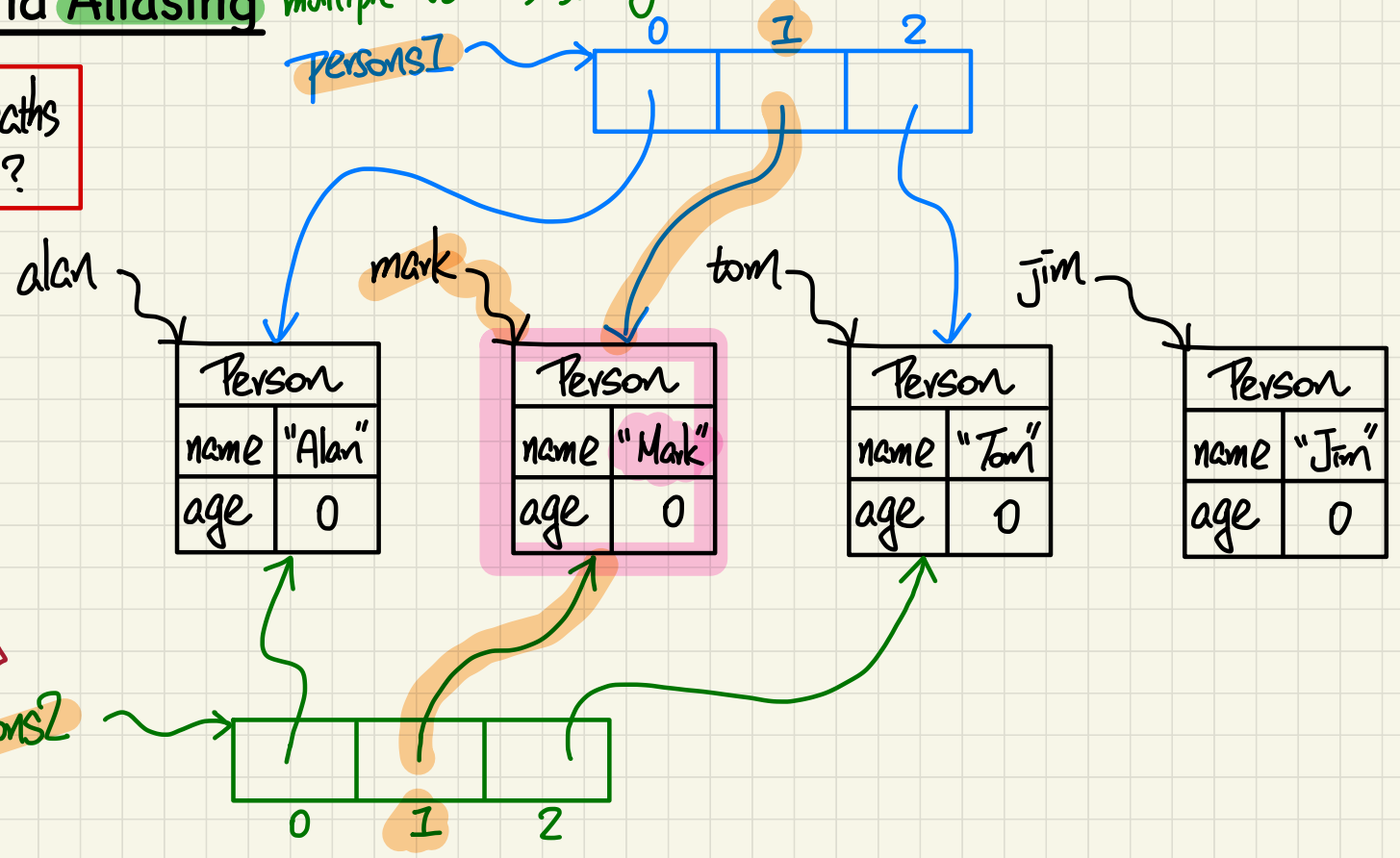
Arrays and Aliasing

multiple variables storing the same address.

All alias paths to "Mark"?

① 3 variables storing the address of "Mark" object

② mark
persons1[1]
persons2[1]



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;
  }
}
```

explicit parameter, presumably not duplicated.

Goal

Auto ID generation & management

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

1. burden on the Account user to specify unique id.

2. when id's are duplicated, not a compilation error.

Declaring Global Variables among Objects

* Each counter instance/obj has its own copy (instance-specific value).
** All counter instances share the same copy (global)

```
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 ++;
    }
}
```

```
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 ++;
    }
}
```

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; } }
```

list of parameters not including id anymore (not burden of user of Account class. anymore).

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