Lecture 2

Monday September 9

- waiting list?
- Lab 0
- Office Hours: 4-6 Mon Tue Wed
- Java Tutorial Series

Constructors not using this Keyword

```java
public class Person {
    /*
     * Attributes.
     * These are variable declared at the class level.
     * All methods may use them.
     */
    int age;
    String nationatlity;
    double weight; /* kg */
    double height; /* meters */

    /*
     * Constructors.
     */
    public Person(int newAge, double newWeight, double newHeight) {
        age = newAge;
        weight = newWeight;
        height = newHeight;
    }
}
```

this

shadowing → logical error: assign param to itself.

1. same parameter & attribute names ?
2. implicit "this"

```java
public class Tester {

    public static void main(String[] args) {
        Person jim = new Person(45, 72, 1.72);
        Person jonathan = new Person(62, 65, 1.81);
    }

}
```

Effect of Creating a New Object

```java
public class Person {
    /*
     * Attributes
     */
    int age;
    String nationality;
    double weight; /* kg */
    double height; /* meters */

    /*
     * Constructors
     */
    Person (int age, double weight, double height) {
        this.age = age;
        this.weight = weight;
        this.height = height;
    }
```

MODEL

usage

```
new Person(45, 72, 1.72)
new Person(62, 65, 1.81)
```

```java
public static void main(String[] args) {
    Person jim = new Person(45, 72, 1.72);
    Person jonathan = new Person(62, 65, 1.81);
}
```

TESTER

(SEQUENCE OF BYTES) → MEMORY

Jim

Person

Jim: getWeight(3)

Jonathan getWeight()

→ printf(Jonathan)

Jonathan

w | 75

aliasing:

multiple copies of the same address.

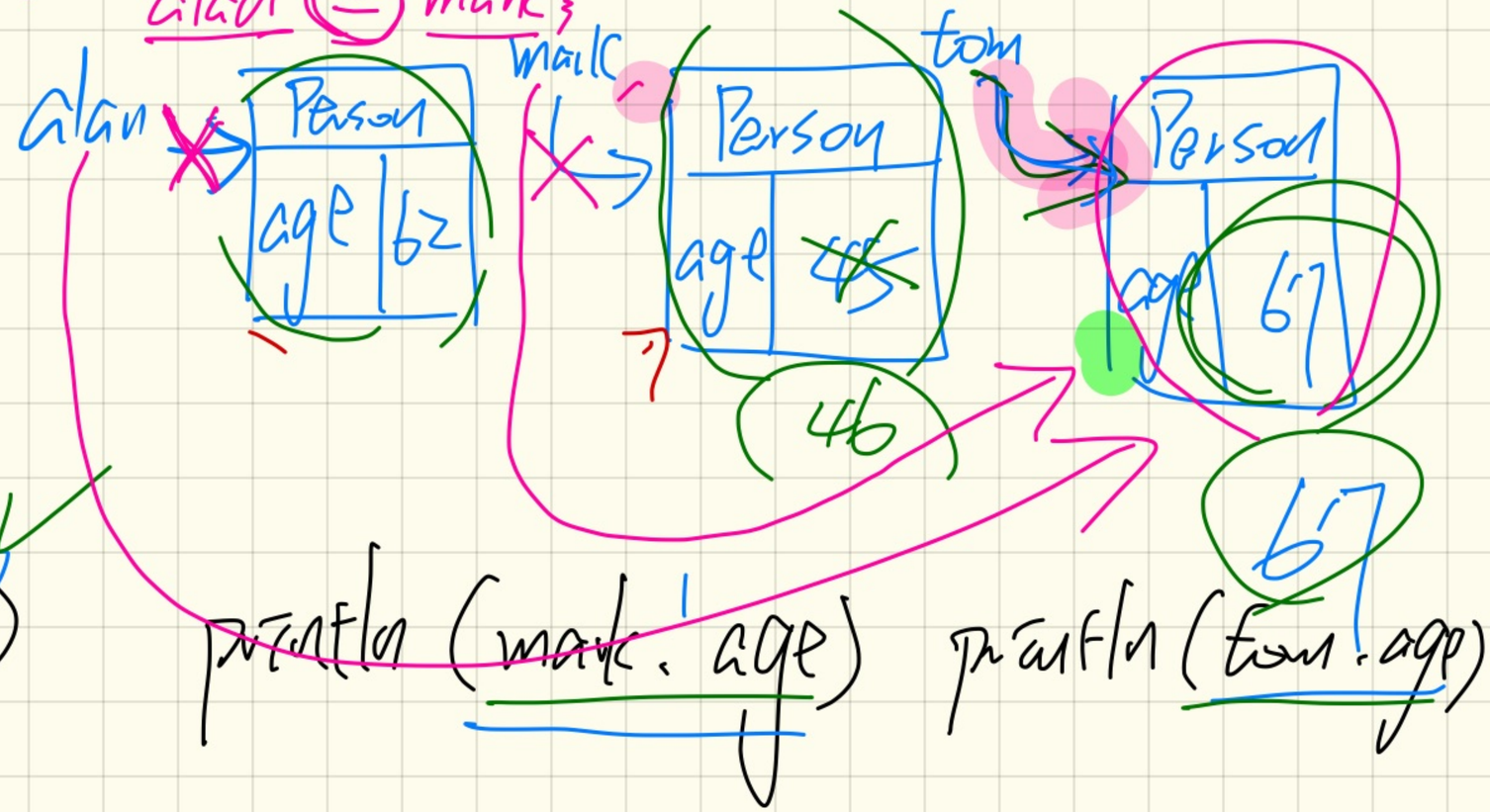
Jim = Jonathan

```
Person alan = new Person(62);
Person mark = new Person(45);
Person tom = new Person(67);

alan = mark;
mark = tom;
alan.getOlder();
println(alan.age)
```

```
class Person{
  int age;
}
```

mark = tom;
alan = mark;



→ Person alan = new Person(45);

→ Person tom = new Person(62);

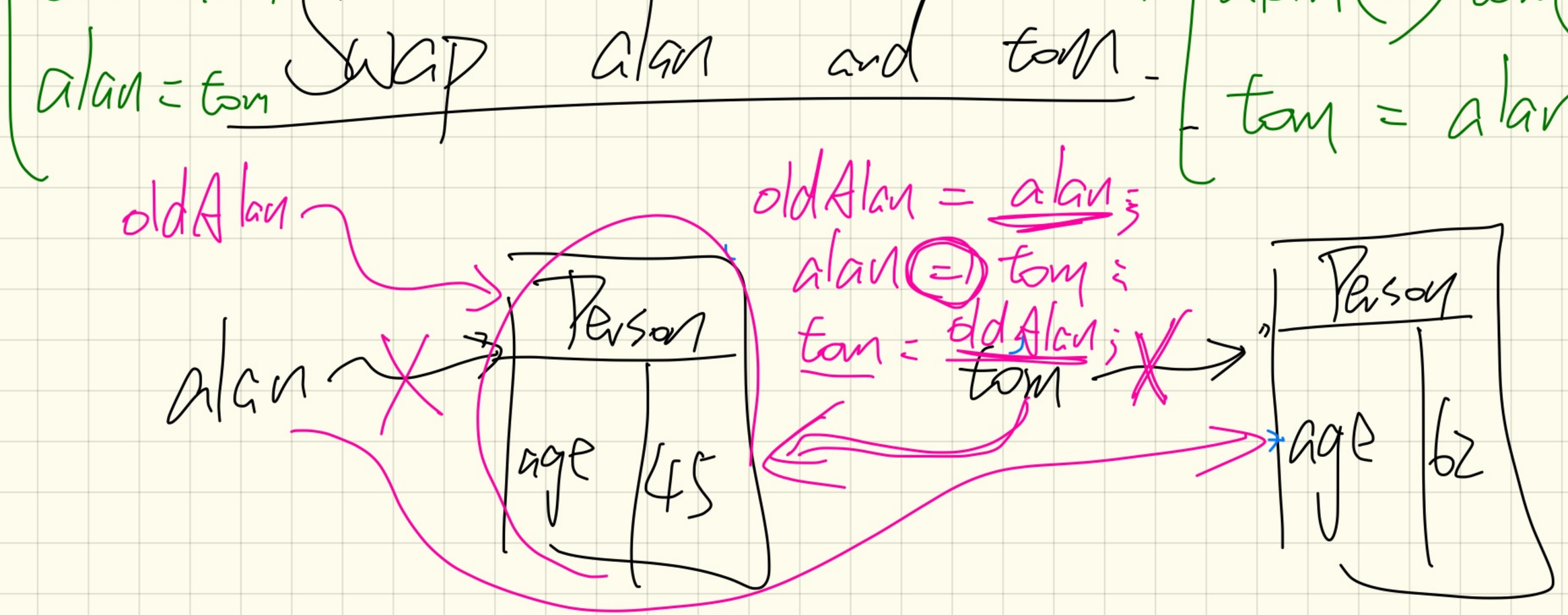
Person oldAlan;

oldAlan = alan;

alan = tom;

tom = oldAlan;

Swap alan and tom.



int [];

Person alan = new ...

Person mark = new ...

Person tom = new ...

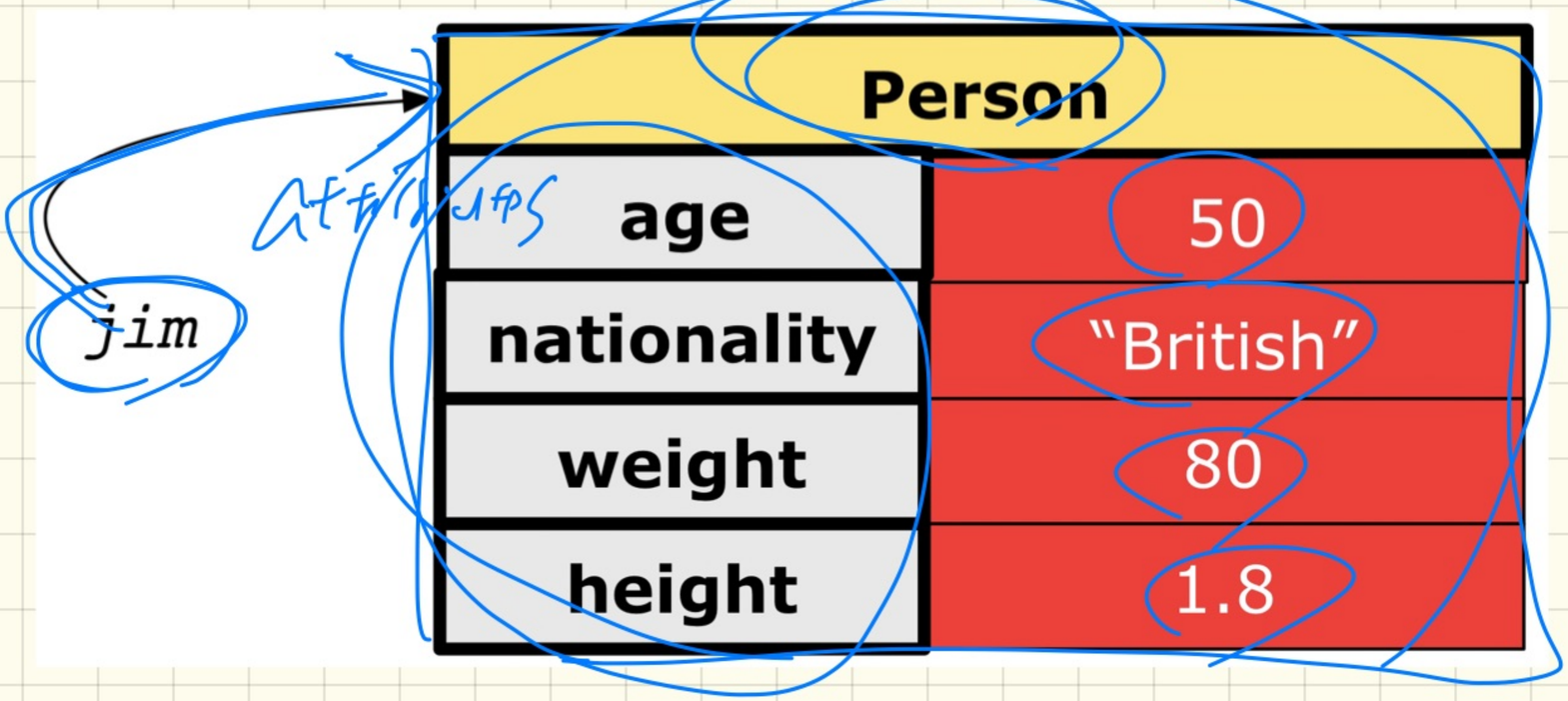
alan = mark;

alan = tom;

Tracing OO Code: Visualizing Objects

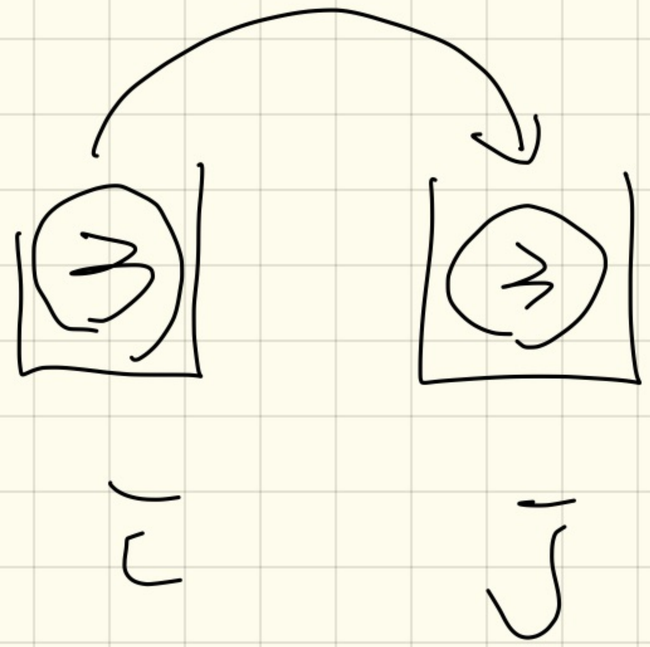
To visualize an object:

- Draw a rectangle box to represent *contents* of that object:
  - Title indicates the *name of class* from which the object is instantiated.
  - Left column enumerates *names of attributes* of the instantiated class.
  - Right column fills in *values* of the corresponding attributes.
- Draw arrow(s) for *variable(s)* that store the object's *address*.

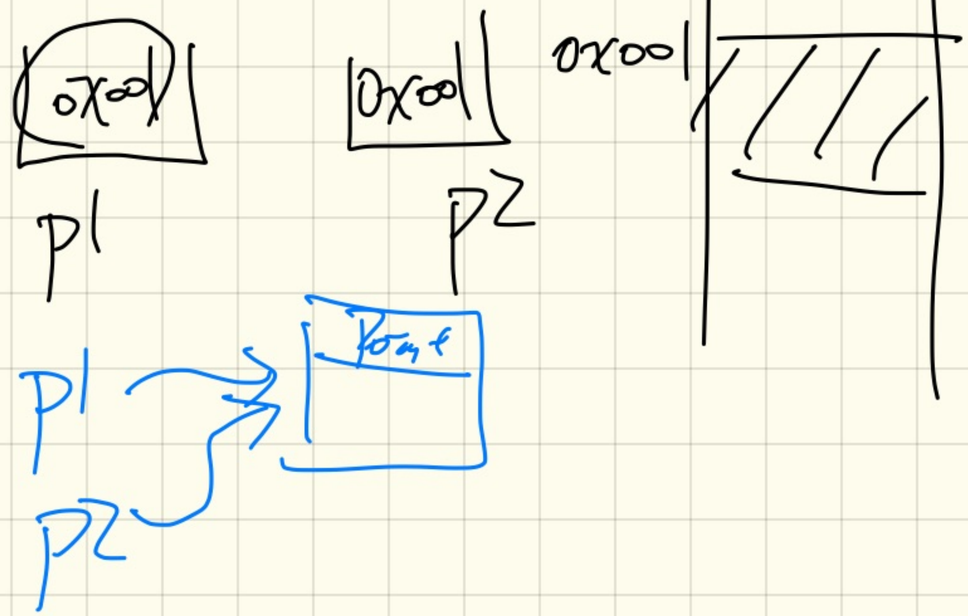


Short Circuit -

```
int i = 3;
int j = i;
```




```
Point p1 = new Point(..);
Point p2 = p1;
```



Person[ ] persons1 = {alan, mark, tom};

ps1[0].setAge(70)

array of person addresses

Person[ ] persons1 = new Person[3];

persons1[0] = alan; persons1[1] = mark; persons1[2] = tom;

alan → Person { n. "Alan", a. 70 }

mark → Person { n. "Mark", a. 0 }

tom → Person { n. "Tom", a. 0 }

Jim → Person { n. "Jim", a. 0 }

persons1 → [null | null | null]

alan
ps1[0]
ps2[2]

persons2 → [null | null | null]

persons2[0] = ps1[1]
persons2[1] = ps1[2]
persons2[2] = ps1[0]

persons2[i] = persons1[(i+1) % ps1.length]

i: 0 1 2

(i+1): 1 2 3

ps1.length: 3 3 3

(i+1) % ps1.length: 1 2 0

0) Person alan;

address of person object

Person[] persons = new Person[3];

~~persons~~ = { dan, mark, tom };

only usable in initialization;

not re-assignment

Persons[] ps = - - -

X ps[0] = "Jackie";

String