Lecture 3 - Sep. 14

Review on OOP

Object Orientation
Tracing OO Programs, Aliasing, Arrays

- Lab0 Part 1 Due Soon
- Lab0 Part 2 Released on Tuesday
- Lab1 to be released on Friday

1. ref. type
2. Arrays

Constructors not using this Keyword

model

```java
public class Person {
    /*
     * Attributes.
     * Person instances have the same attribute names.
     * Person instances have specific attribute values.
     */
    double weight;
    double height;

    /*
     * Constructors
     */
    public Person() {

    }

    public Person(double newWeight, double newHeight) {
        weight = newWeight;
        height = newHeight;
    }
}
```

JUnit

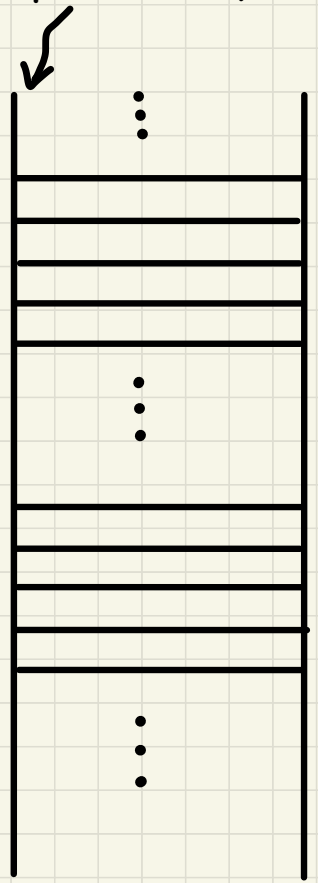
```java
@Test
public void test_1() {
    Person jim = new Person(72, 1.81);
    Person jonathan = new Person(65, 1.67);
    assertTrue(jim != jonathan);
    assertFalse(jim == jonathan);
    assertNotSame(jim, jonathan);
    assertNotEquals(jim, jonathan);
}
```

```java
public static void main(String[] args) {
    Person jim = new Person(72, 1.81);
    Person jonathan = new Person(65, 1.67);
    System.out.println(jim);
    System.out.println(jonathan);
}
```

console

- Default Constructor?
- Parameters vs. Arguments
- Reference Variables

memory
(sequence of bytes)



Parameters vs. Arguments

Template Definition

```
class Point {
   Point(double x, double y) {…}

   double getDistanceFrom(Point other) {…}

   void move(char direction, double units) {…}
}
```

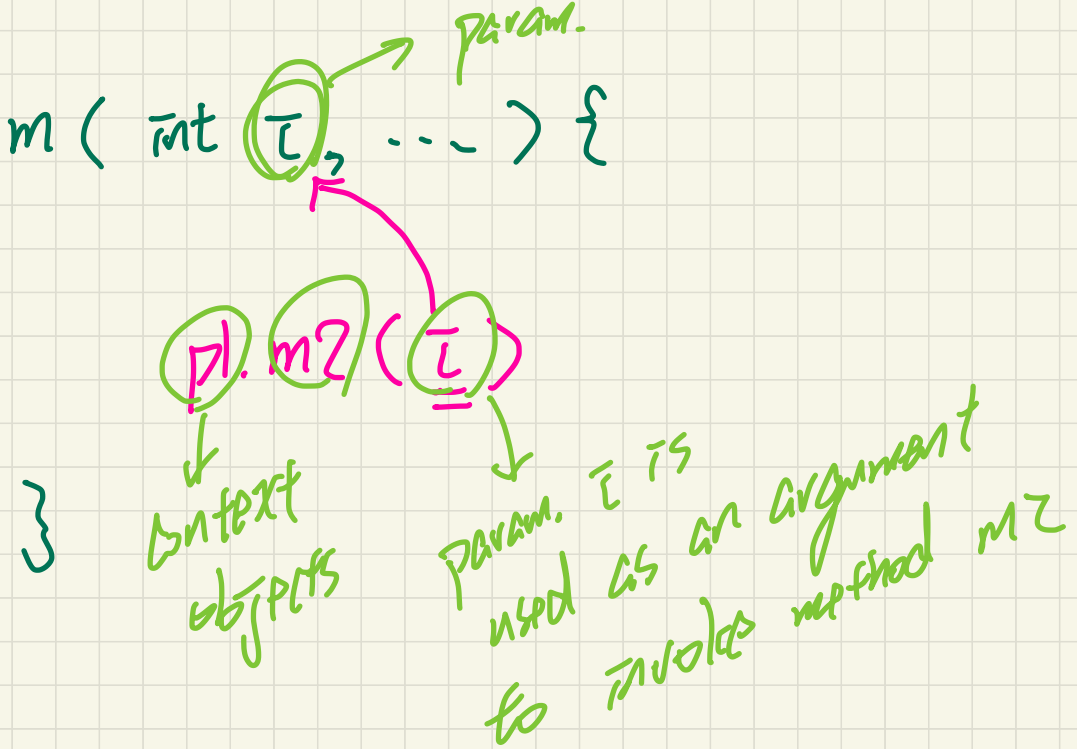
Method Usages

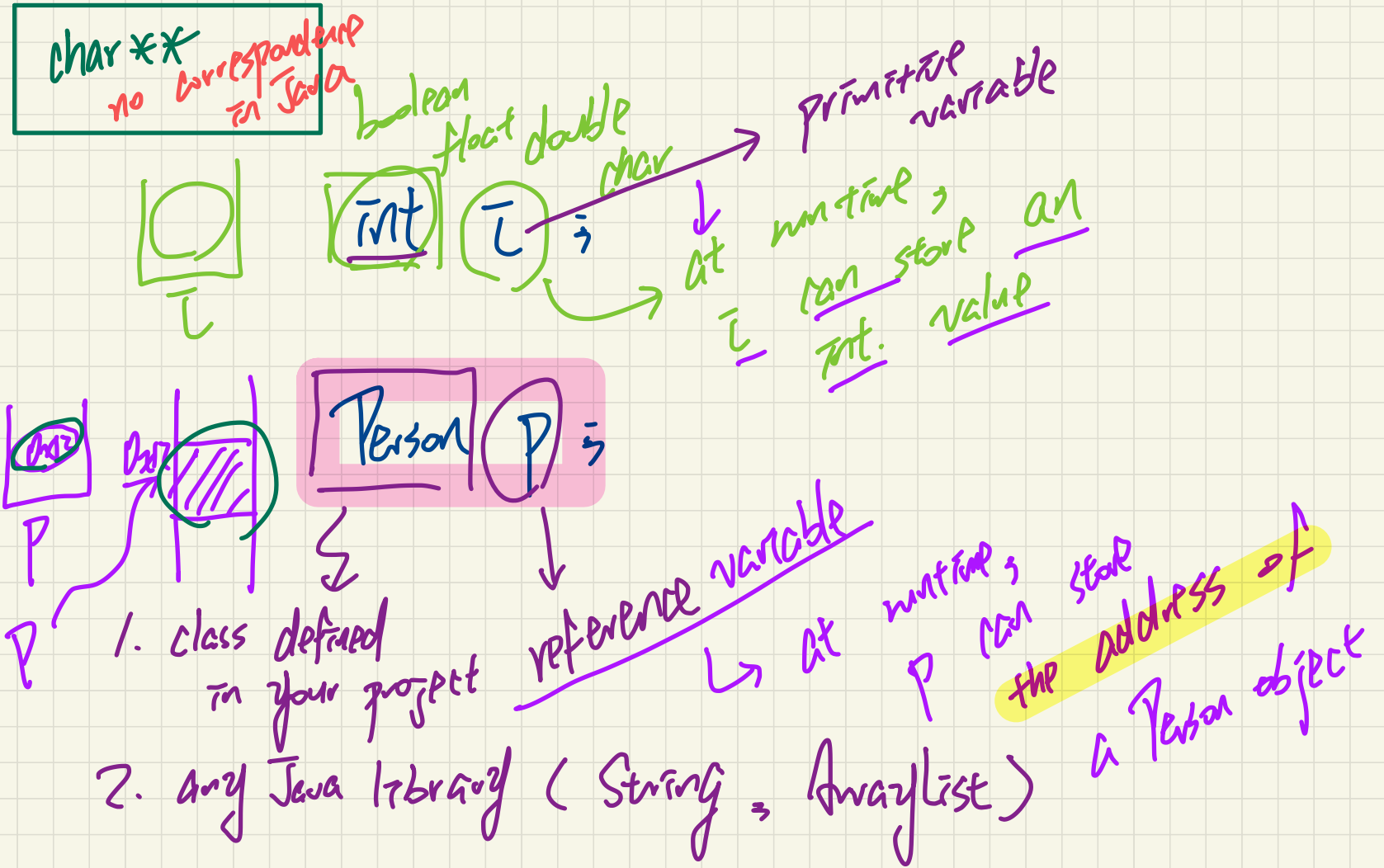
```
class PointTester {
   static void main(String[] args) {
      Point p1 = new Point(2.5, -3.6);
      Point p2 = new Point(-4.8, 5.9);
      double dist1 = p1.getDistanceFrom(p2);
      double dist2 = p2.getDistanceFrom(p1);
      p1.move('R', 7.6);
   }
}
```

Q: Can parameters be used as arguments?

Can arg. be used as parameters?

No.

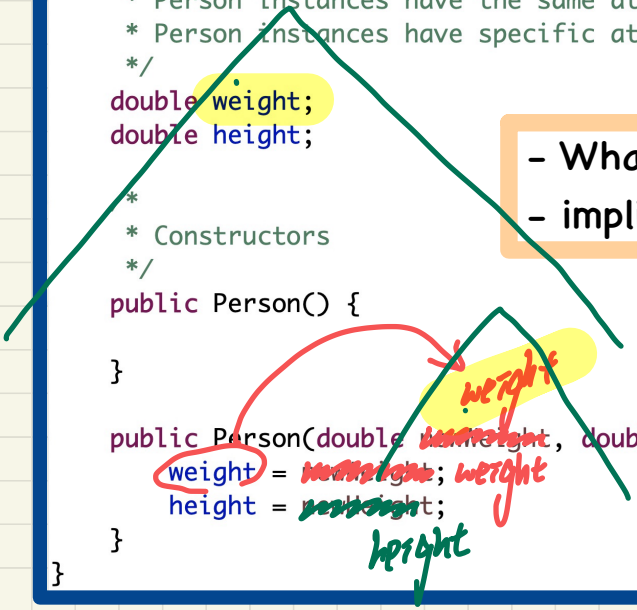



char**
no correspondence in Java
boolean float double char
int
i
primitive variable
i
at runtime, can store an int. value
Person p =
p
p
1. class defined in your project
reference variable
at runtime, p can store the address of a Person object
2. any Java library (String, ArrayList)

Constructors not using this Keyword

model

```
public class Person {
    /*
     * Attributes.
     * Person instances have the same attribute names.
     * Person instances have specific attribute values.
     */
    double weight;
    double height;

    /*
     * Constructors
     */
    public Person() {

    }

    public Person(double weight, double height) {
        weight = weight;
        height = height;
    }
}
```



Question

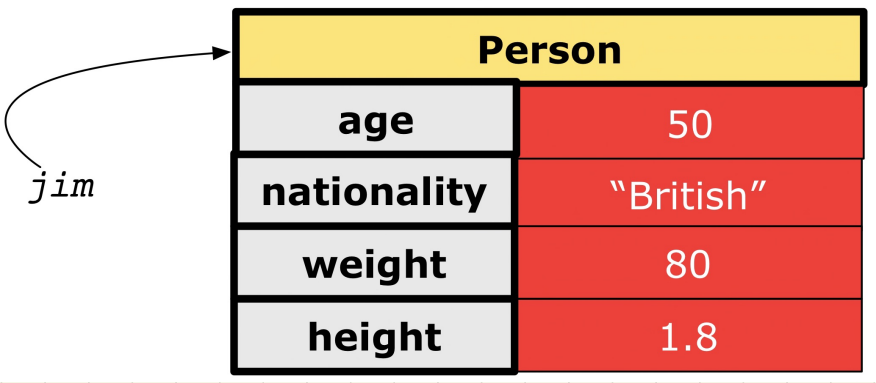
- What if names of parameter & attribute are the same?
- implicit "this"

variable shadowing

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*.



Effects of Creating New Objects

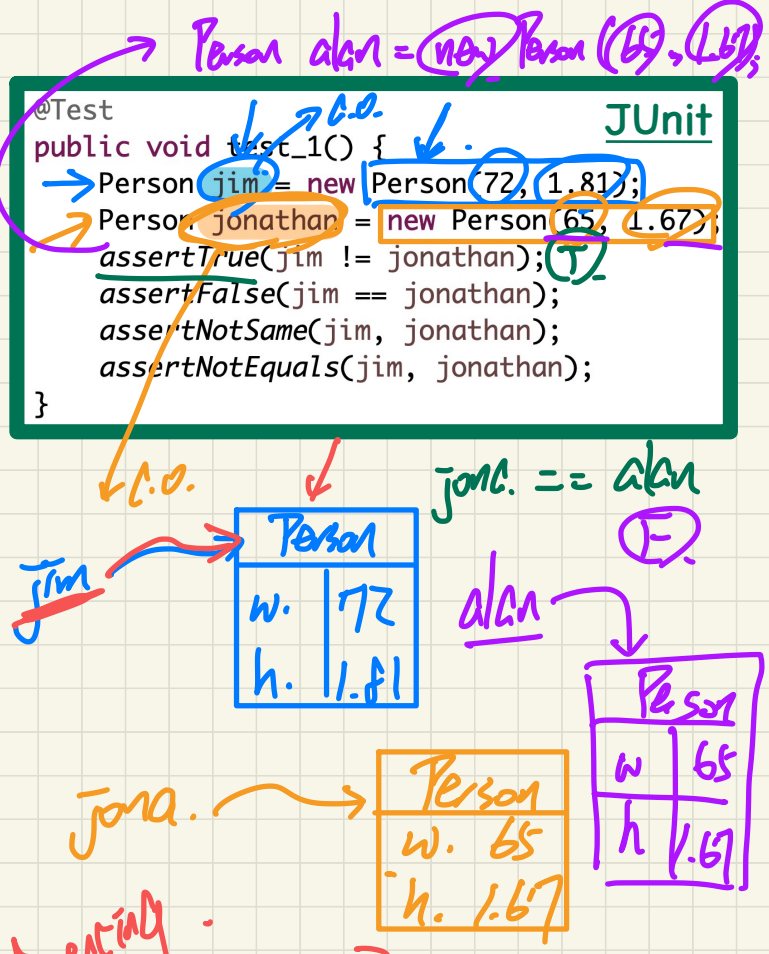
model

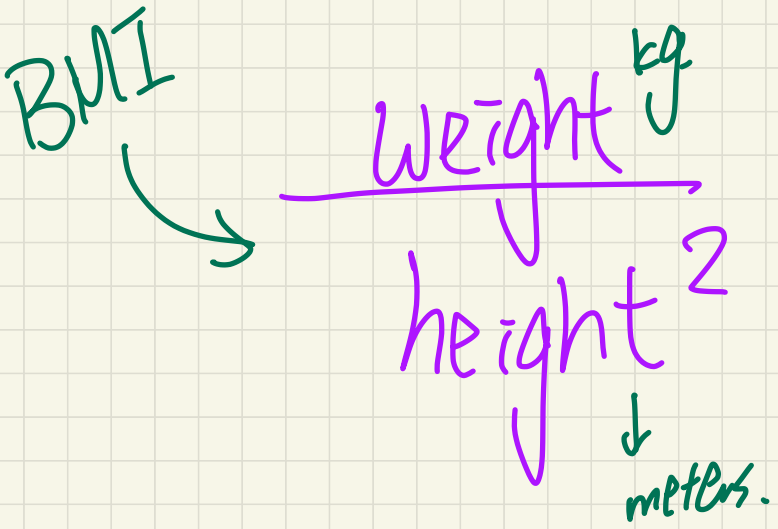
```
public class Person {
    /*
     * Attributes.
     * Person instances have the same attribute names.
     * Person instances have specific attribute values.
     */
    double weight;
    double height;

    /*
     * Constructors
     */
    public Person() {

    }

    public Person(double weight, double height) {
        this.weight = weight;
        this.height = height;
    }
}
```

- Variable Shadowing
- Visualizing Objects
- Context Object
- this
- dot notation

JUnit

```
@Test
public void test_1() {
    Person jim = new Person(72, 1.81);
    Person jonathan = new Person(65, 1.67);
    assertTrue(jim != jonathan);
    assertFalse(jim == jonathan);
    assertNotSame(jim, jonathan);
    assertNotEquals(jim, jonathan);
}
```

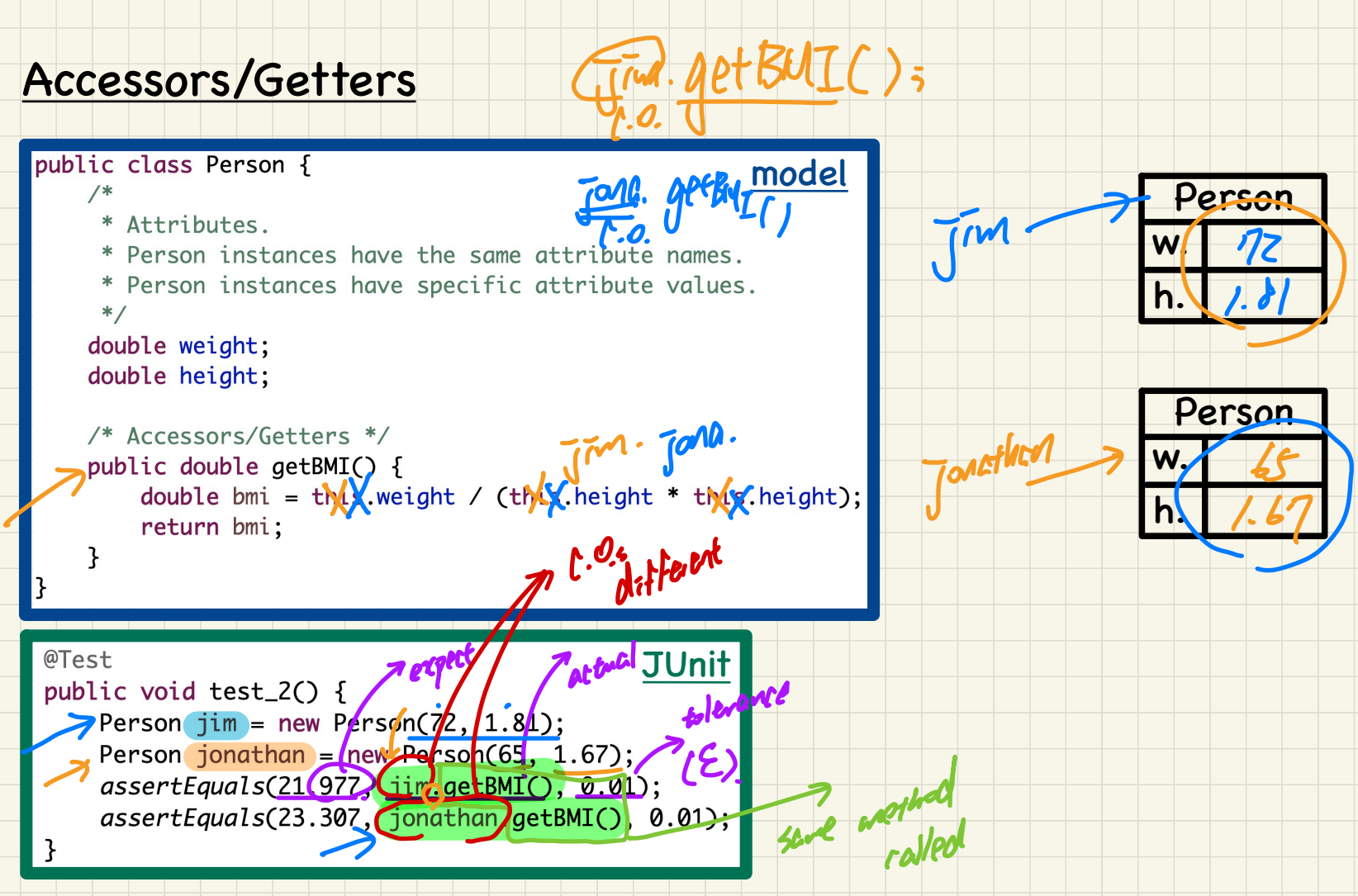



BMI
weight
kg
height
2
meters.

Accessors/Getters

```java
public class Person {
    /*
     * Attributes.
     * Person instances have the same attribute names.
     * Person instances have specific attribute values.
     */
    double weight;
    double height;

    /* Accessors/Getters */
    public double getBMI() {
        double bmi = this.weight / (this.height * this.height);
        return bmi;
    }
}
```

```java
@Test
public void test_2() {
    Person jim = new Person(72, 1.81);
    Person jonathan = new Person(65, 1.67);
    assertEquals(21.977, jim.getBMI(), 0.01);
    assertEquals(23.307, jonathan.getBMI(), 0.01);
}
```



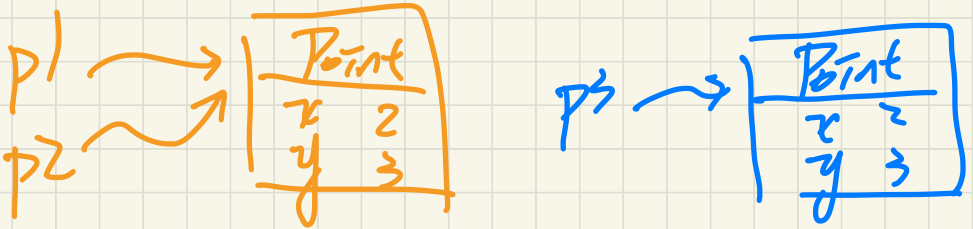
Copying Primitive vs. Reference Values

Primitive

```
int i = 3;
int j = i;   System.out.println(i == j);/*true*/
int k = 3;   System.out.println(k == i && k == j);/*true*/
```

Reference

```
Point p1 = new Point(2, 3);
Point p2 = p1;   System.out.println(p1 == p2);
Point p3 = new Point(2, 3);
Systme.out.println(p3 == p1 || p3 == p2);/*false*/
Systme.out.println(p3.x == p1.x && p3.y == p1.y);
Systme.out.println(p3.x == p2.x && p3.y == p2.y);
```



Exercise

Person[] persons1; → stores the beginning address of the array

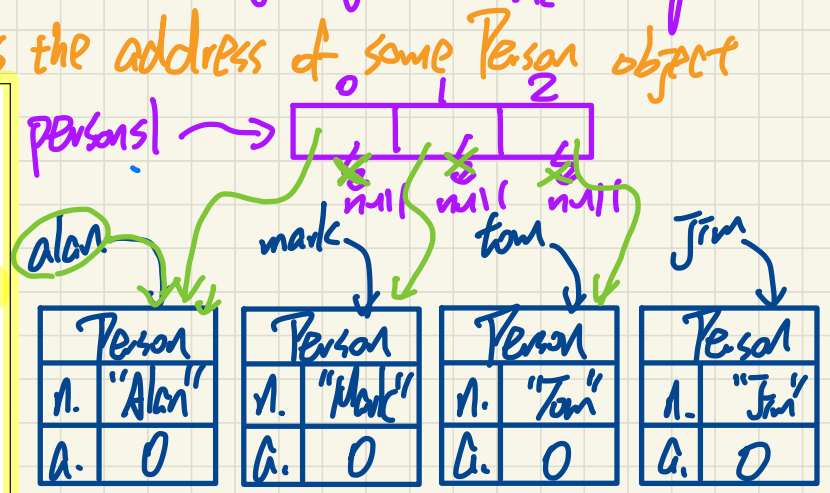
each index of the array stores the address of some Person object

```
1  Person alan = new Person("Alan");
2  Person mark = new Person("Mark");
3  Person tom = new Person("Tom");
4  Person jim = new Person("Jim");
5  Person[] persons1 = {alan, mark, tom};
6  Person[] persons2 = new Person[persons1.length];
7  for(int i = 0; i < persons1.length; i ++) {
8    persons2[i] = persons1[i];
9  }
10 persons1[0].setAge(70);
11 System.out.println(jim.getAge());
12 System.out.println(alan.getAge());
13 System.out.println(persons2[0].getAge());
14 persons1[0] = jim;
15 persons1[0].setAge(75);
16 System.out.println(jim.getAge());
17 System.out.println(alan.getAge());
18 System.out.println(persons2[0].getAge());
```

name
age

3

3 iterations: 0, 1, 2

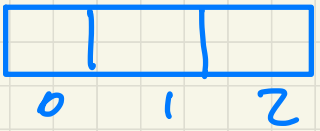


Person[] persons1 = new Person[3]; ✓

persons1[0] = alan; // copy add. stored in alan to index 0.
persons1[1] = mark;
persons1[2] = tom;

step of array

persons2 →



1st iteration

persons2[0] = persons1[0];

% Remainder = modulo op.

Person[] ps = new Person[MAX];

10

① MAX indices in the array

② Range of indices: 0 .. MAX-1

10

① ps.length == MAX

largest index: ps.length-1