EEC S1022 Programming for Mobile Computing (Winter 2021)

Q&A - Lectures W6

Monday, March 1

loops
Arrays

OO X

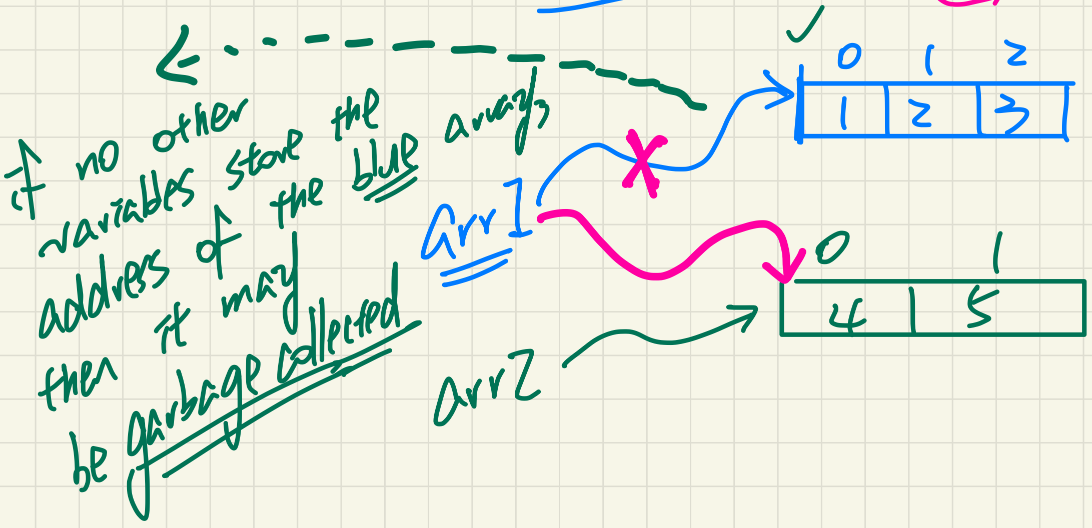
- Programming Test 2 on Wed and Thu (2 hours, 4 problems)
- Written Test 2 Guide
- Lab6 (tutorial videos + written notes)
- Lectures W7

```
int[] arr1 = {1, 2, 3};

int[] arr2 = {4, 5};
```

$\boxed{\text{Arr1 == arr2}}$ → are arr1 and arr2 pointing to the same array object?

F.



if no other variables store the address of the blue arrays then it may be garbage collected

arr1 = arr2;

↳ copy address stored in arr2 into arr1.

Given two arrays arr1 and arr2

1. if arrays store primitive values e.g. int[]

arr1.equals(arr2) ✓

~~String~~
↳ reference.

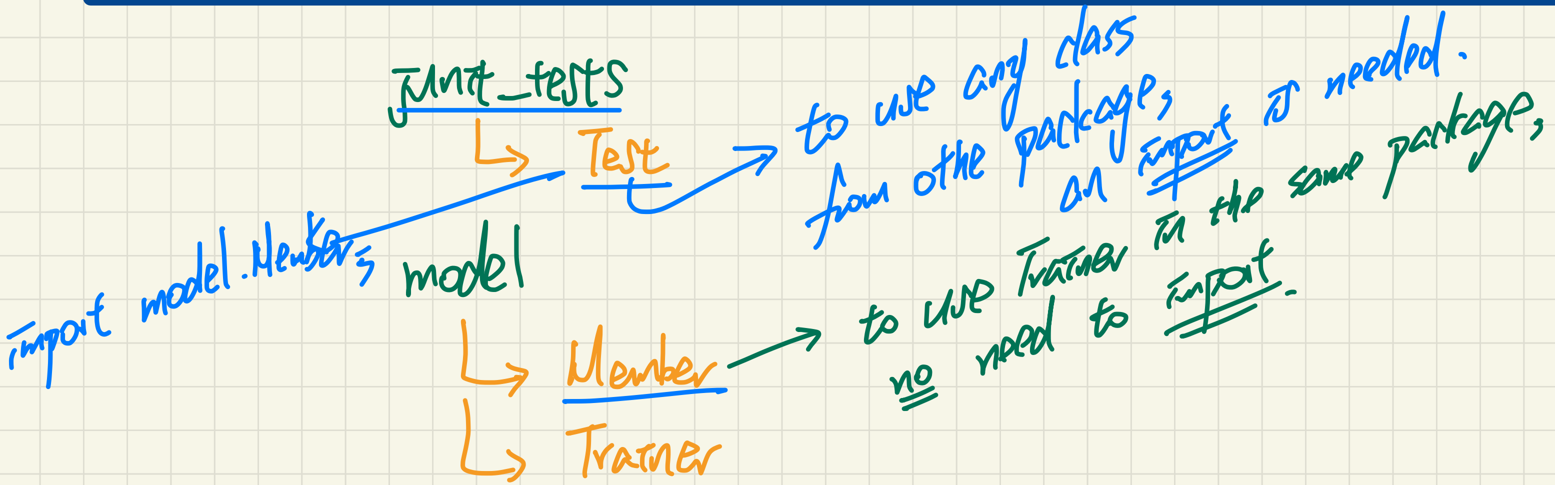
2. if array store reference values e.g. Member[]

1. class from API
2. class you declare.

arr1.equals(arr2) X

↳ compare not the content but addresses stored in each index.

For Lab 6, my model.Registration class used a class variable of type Instructor (found in model.Instructor class).
Why don't I need to import model.Instructor at the top of the model.Registration file?


junit_tests
Test
model
Member
Trainer
import model.Member;
to use any class from othe package, an import is needed.
to use Trainer in the same package, no need to import.

When is the use of **this** required?

1. "this" is implicit for each mention of an attribute. (except when variable shadowing is in place)
2. "this" is necessary when an attribute name clashes with the input parameter names
3. "this" denotes the $\boxed{\text{context object}}$.
   ① object creation
   ② method call.

```
public class Rectangle {
    private int length;
    private int width;

    public Rectangle(int newLength, int newWidth) {
        length = newLength;
        width = newWidth;
    }
}
```

mentions of attributes.

implicitly:

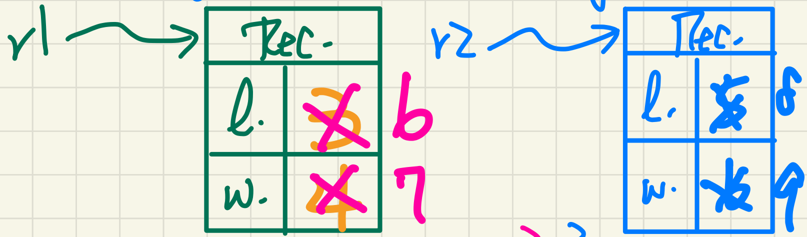
```
r1 this.length = newLength;
r1 this.width = newWidth;
```

context object.

context object.

Rectangle r1 = new Rectangle(3, 4);

Rectangle r2 = new Rectangle(5, 6)



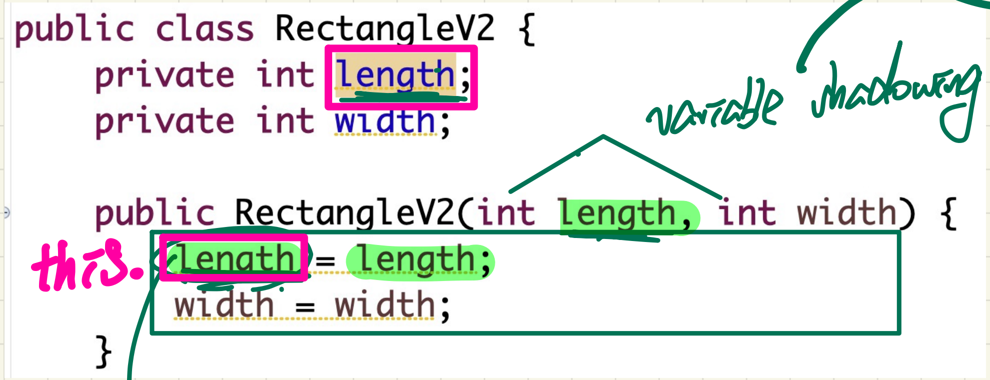
3 3

```
public void growBy(int units) {
    ~~this~~.length += units; 3
    ~~this~~.width += units; 3
}
```

r1 r2

r1.growBy(3);

r2.growBy(3);



```
public class RectangleV2 {
    private int length;
    private int width;

    public RectangleV2(int length, int width) {
        length = length;
        width = width;
    }
```

this.

variable shadowing

What's being shadowed? (att. or param.?)

↳ "Jackie"

this mention refers to the input parameter length. (rather than attribute that's been hidden).

↳ in this case, use of "this" is necessary to disambiguate.

Constructor must have *distinct* lists of *parameter types*.

- ① `Person(String n),Person(String n, int age)` ✓
- ② `Person(String n, int age),Person(int age, String n)` ✓
- ③ `Person(String fN, int age),Person(String lN, int id)` ✗

new Person("Jim", 46);

Person("Jim", 23)

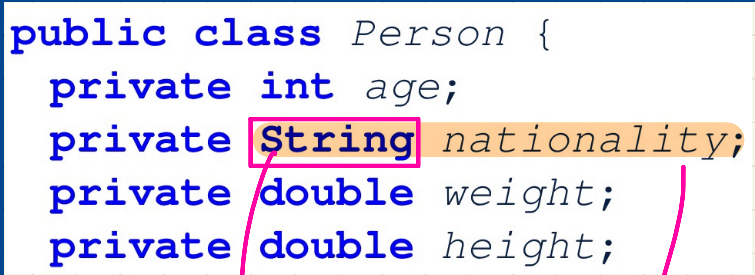
```
1 Point p1 = new Point(3, 4);
2 Point p2 = new Point(-4, -3);
3 System.out.println(p1.getDistanceFromOrigin());
4 System.out.println(p2.getDistanceFromOrigin());
5 p1.moveUp(1);
6 p2.moveUp(1);
7 System.out.println(p1.getDistanceFromOrigin());
8 System.out.println(p2.getDistanceFromOrigin());
```

- **Lines 3 and 7:** invoking the same accessor method on the same instance *may* return *distinct* values, why? [redacted]

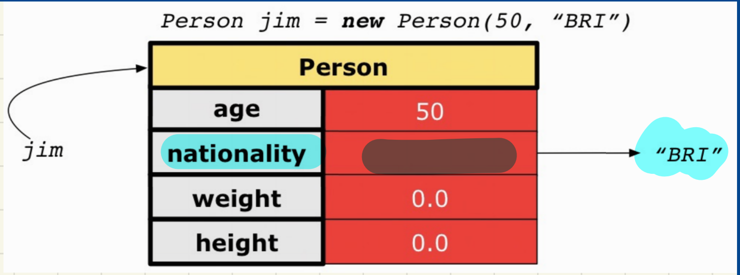
this call is made after the change on p1 at line 5

```
p1.getDistanceFromOrigin(); → y == ~~0~~ 4
① p1.moveUp(1);   p1.moveUp(-1);
② p1.moveUp(1);   p1.moveUp(-2);
p1.getDistanceFromOrigin();
```

In this particular illustration of object creation, why is the String attribute shown to be separate from the columns, pointing away from it to the value it is assigned to?



```
public class Person {
  private int age;
  private String nationality;
  private double weight;
  private double height;
```



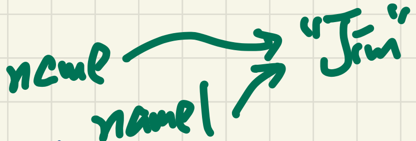
a reference type

stores the address of some String object.

```
public static int[] task1(int n){
        int[] result = null;
        result = new int[4];

                                        boolean result = false;
        return result;                  result = true;
}
```

Call by value



* p1.n = name;

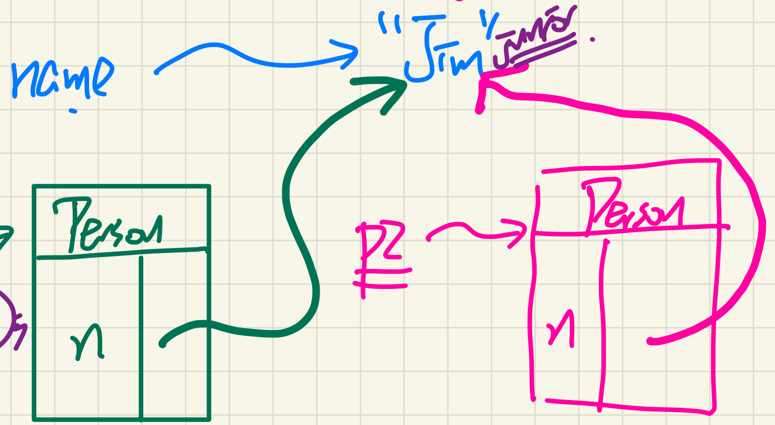
** p2.n = name;

```
String name = "Jim";
Person p1 = new Person(name);
String name1 = "Jim";
Person p2 = new Person(name);
                       name1?
```

name1

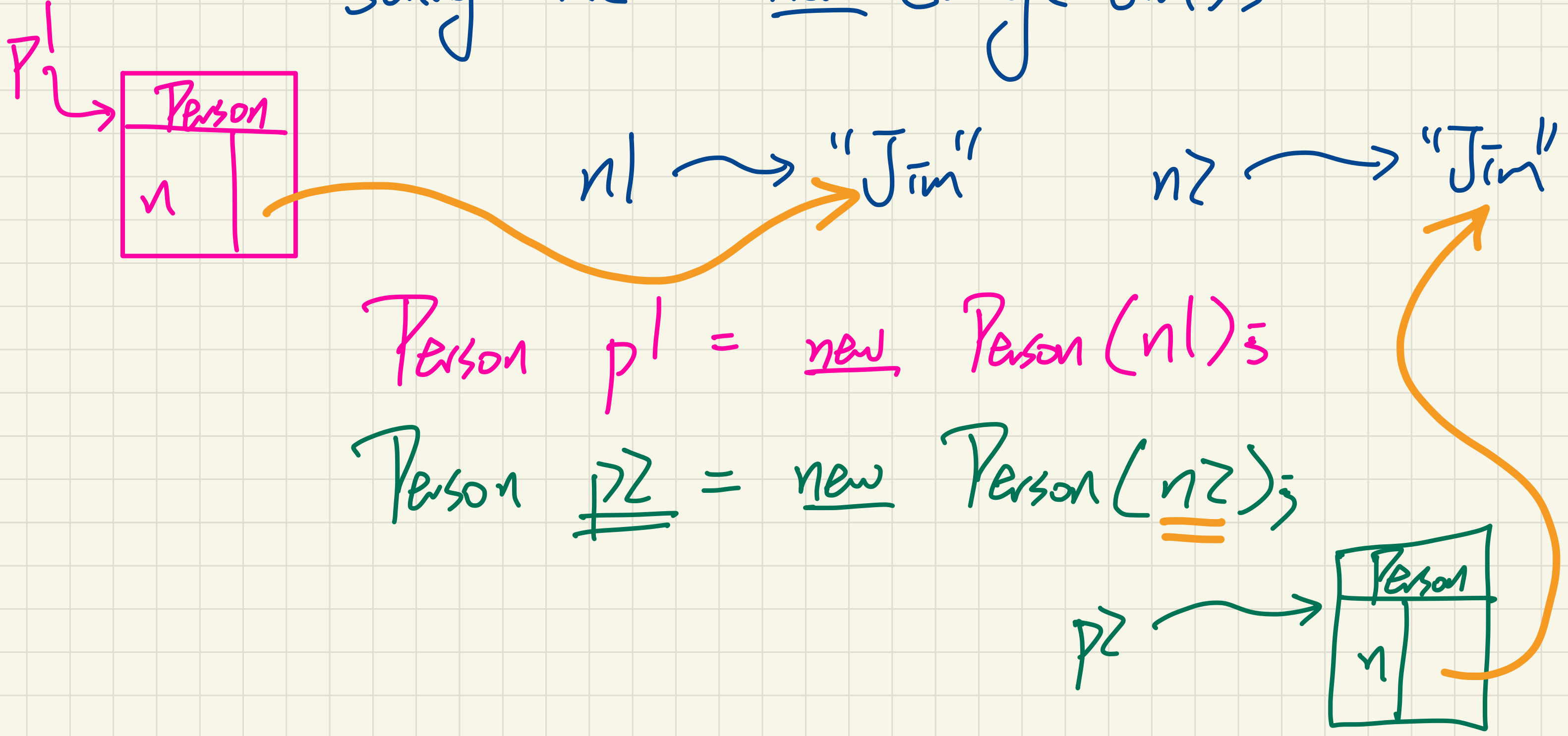
```
Person(String ~~x~~ name) {
  *   this.n = ~~x~~;
 **               name
}     p1 p2
```

p1.name.append("junior");



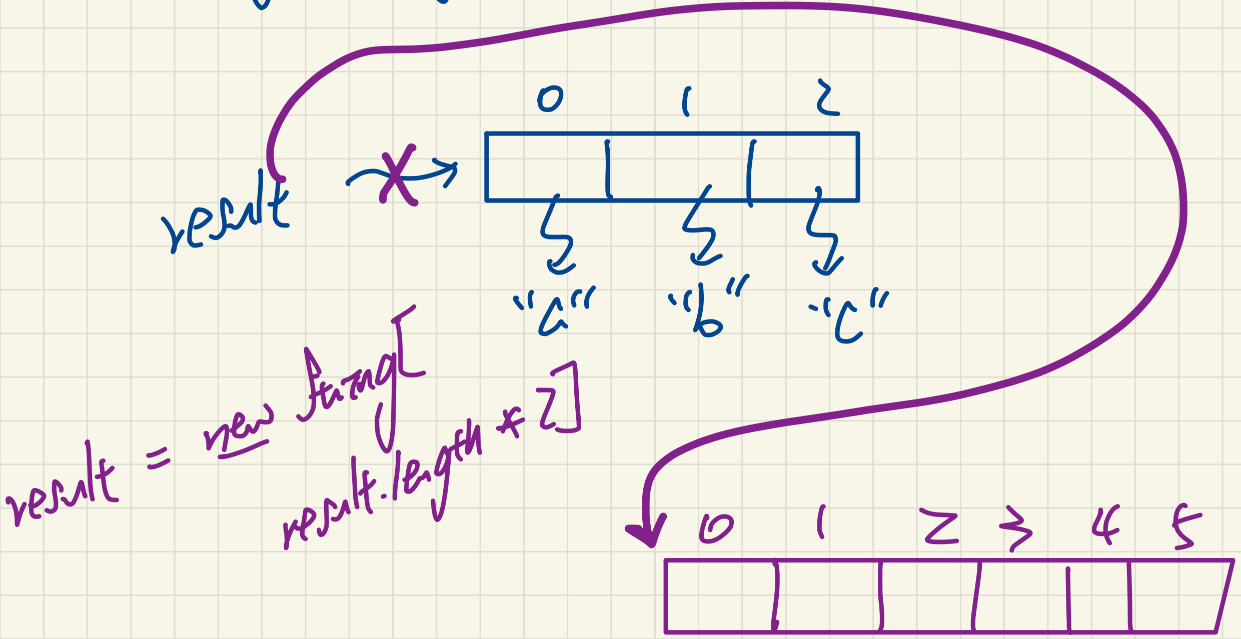
1 1 2 3

```
String n1 = new String("Jim");
String n2 = new String("Jim");
```



```
Person p1 = new Person(n1);
Person p2 = new Person(n2);
```

Dynamically increase the size of array



0
1
2
result
"a"
"b"
"c"
result = new String[result.length * 2]
0
1
2
3
4
5