EECS2030 Advanced Object-Oriented Programming (Fall 2021)

Q&A - Lecture 2b

Wednesday, October 6

Lectures 2a W3

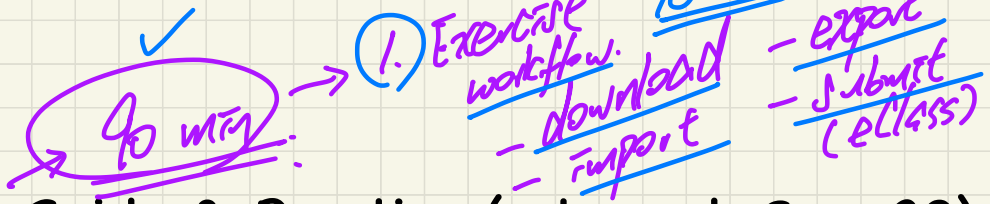
2b W4

3 W5

↳ released after P.T.I.

No scheduled TA sessions during reading week.

Announcement



- Programming Test 1 Guide & Practice (released: Sep. 29)
- Tutorial Video on Lab1 Solution Walk-Through
- Lecture W4 (released: Sep. 27)
- Lab2 (released: Oct. 1; due: Oct. 15)

Assume that a Person class is already defined, and it has an attribute name, a constructor that initializes the person's name from the input string. Consider the following fragment of Java code (inside some main method):

```
1  Person p0 = new Person("Suyeon");
2  Person p1 = new Person("Yuna");
3  Person p2 = new Person("Sunhye");
4  Person p3 = new Person("Jihye");
5  p1 = p2;
6  p3 = p0;
7  Person[] persons = {p1, p2, p3, p0};
8  p1 = persons[2];
9  persons[3] = p0;
10 System.out.println("Done!");
```

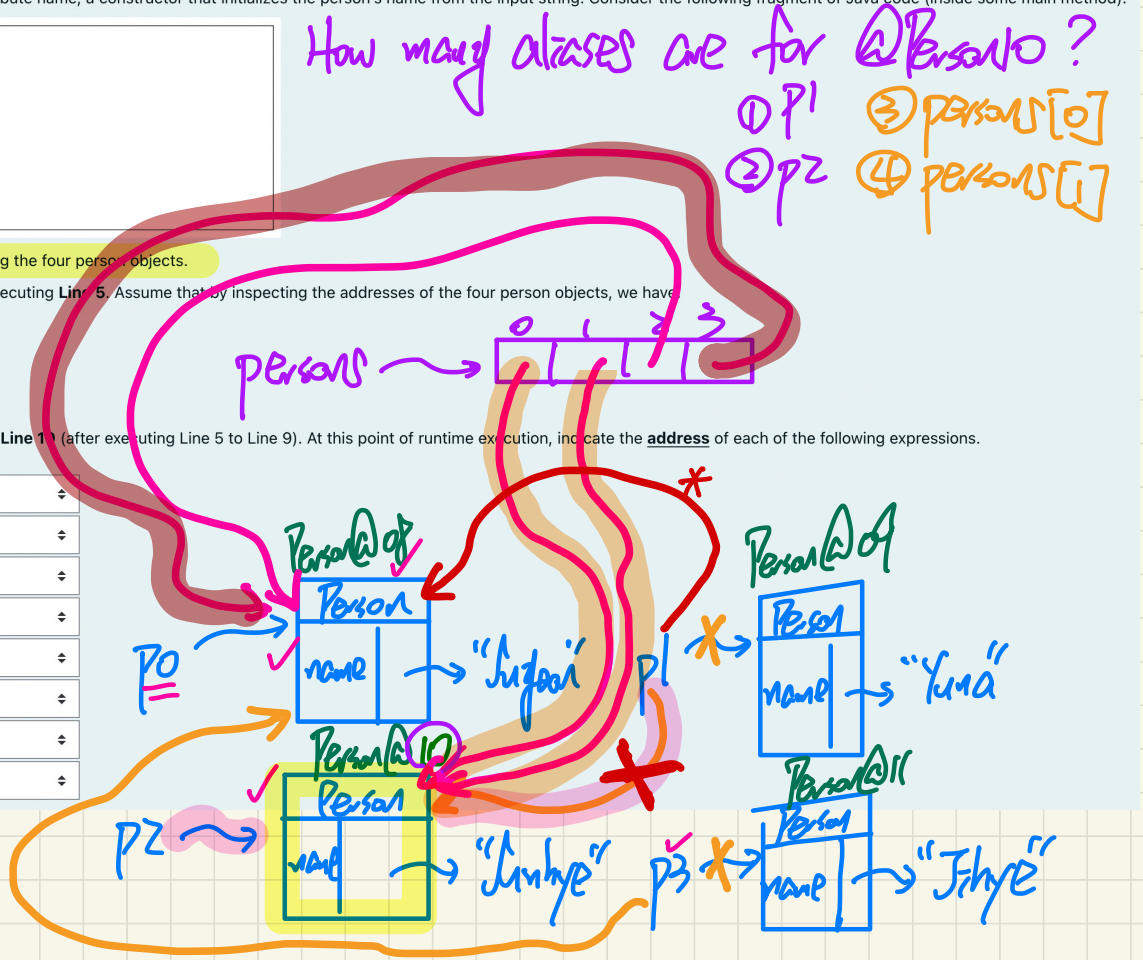
Say in Eclipse we place a breakpoint at **Line 5**, right after creating the four person objects.

By launching the debugger, the execution pauses right before executing **Line 5**. Assume that by inspecting the addresses of the four person objects, we have:

- p0 stores the address: Person@08
- p1 stores the address: Person@09
- p2 stores the address: Person@10
- p3 stores the address: Person@11

Say we continue with the execution and **step over** all the way to **Line 10** (after executing Line 5 to Line 9). At this point of runtime execution, indicate the **address** of each of the following expressions.

p3	Choose...
p1	Choose...
p2	Choose...
persons[2]	Choose...
persons[1]	Choose...
persons[3]	Choose...
p0	Choose...
persons[0]	Choose...



```java
public class E {
    public void em() {
        F obj = new F();
        obj.fm2();
        /* a line of computation: details omitted */
    }
}
```

```java
public class F {
    public void fm1() {
        H obj = new H();
        obj.hm1();
        /* a line of computation: details omitted */
    }

    public void fm2() {
        H obj = new H();
        obj.hm2();
        /* a line of computation: details omitted */
    }
}
```

```java
public class G {
    public void gm() {
        E obj1 = new E();
        obj1.em();
        /* a line of computation: details omitted */
    }
}
```

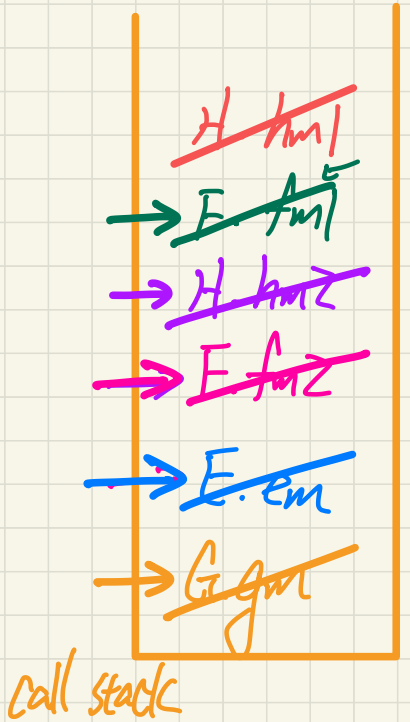
starting point

```java
public class H {
    public void hm1() {
        /* a line of computation: details omitted */
        /* a line of computation: details omitted */
    }

    public void hm2() {
        F obj = new F();
        obj.fm1();
        /* a line of computation: details omitted */
    }
}
```

G 5
G 6
E 5
E 6
F 11
F 12
H 10
H 11
F 5
F 6

H 5
H 6
F 7
H 12
F 13
E 7
G 7

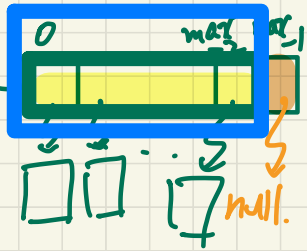


Now, in the context of some **JUnit Test** class, consider the following fragment of code being run as JUnit Test and choose **all** descriptions that are **correct**.

```
RefurbishedStore rs = new RefurbishedStore();
for(int i = 1; i < rs.getMaxCapacity(); i ++) {
    /* Product of each entry is expected to be set later. */
    rs.addEntry(new Entry("sn " + i, null));
}
assertEquals(rs.getMaxCapacity() - 1, rs.getNumberOfEntries());
boolean b = false;
for(int i = 0; i < rs.getNumberOfEntries(); i ++) {
    b = b && rs.getPrivateEntriesArray()[i].getProduct() == null;
}
assertTrue(b);
```

how many iterations?
↓
rs.getMaxCap() - 1.

rs.es



to know if there will be an assertion error, do we need to trace **?

↳ No :) the bounds of loop counter [0, noe - 1] is valid.

② b initialized to false can never turn true by &&.

Hints:

- The number of correct answers may be **one** and may be **more than one** - consider carefully.
- Consider the **sequential** flow of execution at runtime.

- ☐ a. No runtime errors will occur.
- ☐ b. A NullPointerException will occur.
- ☐ c. None of the listed answer is correct.
- ☐ d. An assertion failure will occur.
- ☐ e. A different runtime error (other than ArrayIndexOutOfBoundsException, NullPointerException, and assertion failure) will occur.
- ☐ f. An ArrayIndexOutOfBoundsException will occur.