EECS2030 Advanced Object-Oriented Programming (Fall 2021)

Q&A - Lecture 2a

Thursday, September 30

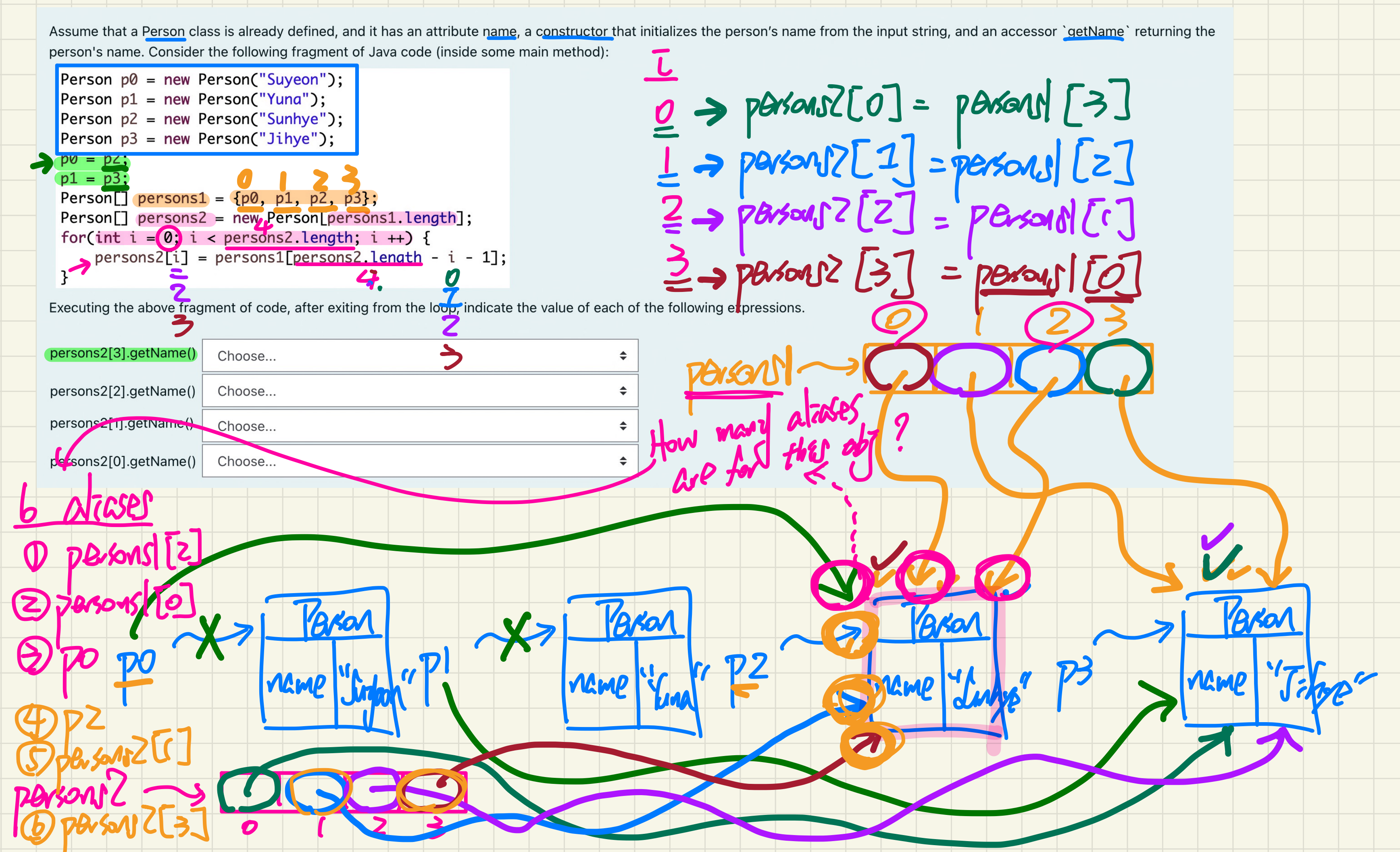
Announcement

2pm EST

- Lab1 (due: Oct. 1)

2pm 2pm EST

- Written Test (due: Sep. 30 - Oct. 1)
- Lecture W4 (released: Sep. 27)
- Programming Test 1 Guide & Practice (released: Sep. 29)
- Lab2 (to be released: Oct. 1)



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, and an accessor `getName` returning the person's name. Consider the following fragment of Java code (inside some main method):

```java
Person p0 = new Person("Suyeon");
Person p1 = new Person("Yuna");
Person p2 = new Person("Sunhye");
Person p3 = new Person("Jihye");
p0 = p2;
p1 = p3;
Person[] persons1 = {p0, p1, p2, p3};
Person[] persons2 = new Person[persons1.length];
for(int i = 0; i < persons2.length; i ++) {
    persons2[i] = persons1[persons2.length - i - 1];
}
```

Executing the above fragment of code, after exiting from the loop, indicate the value of each of the following expressions.

persons2[3].getName() Choose...

persons2[2].getName() Choose...

persons2[1].getName() Choose...

persons2[0].getName() Choose...

i

0 → persons2[0] = persons1[3]

1 → persons2[1] = persons1[2]

2 → persons2[2] = persons1[1]

3 → persons2[3] = persons1[0]

How many aliases are for this obj ?

6 aliases

① persons1[2]

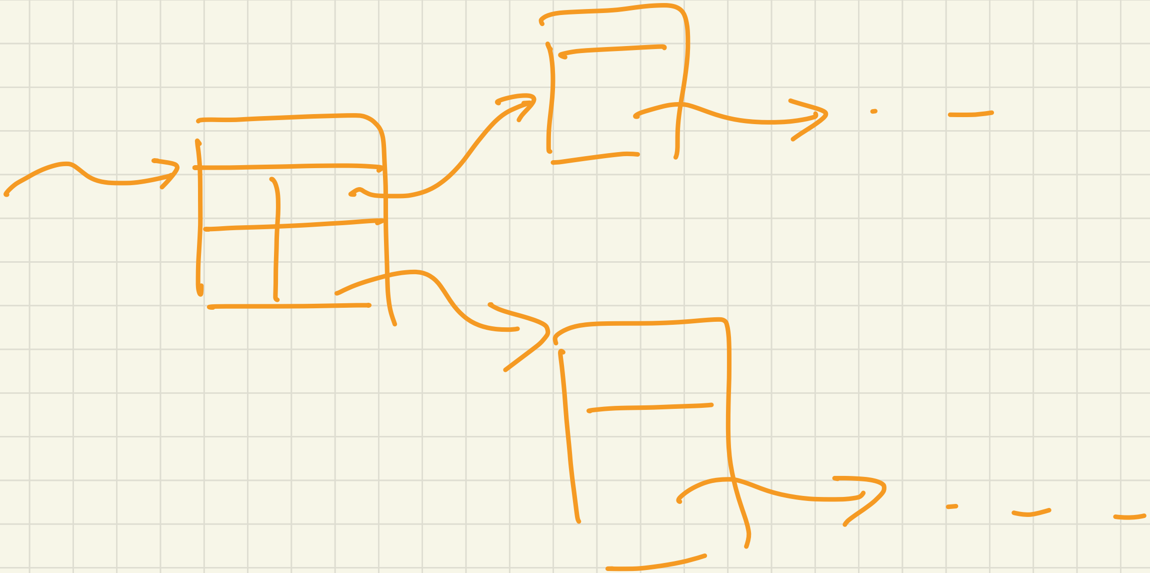
② persons1[0]

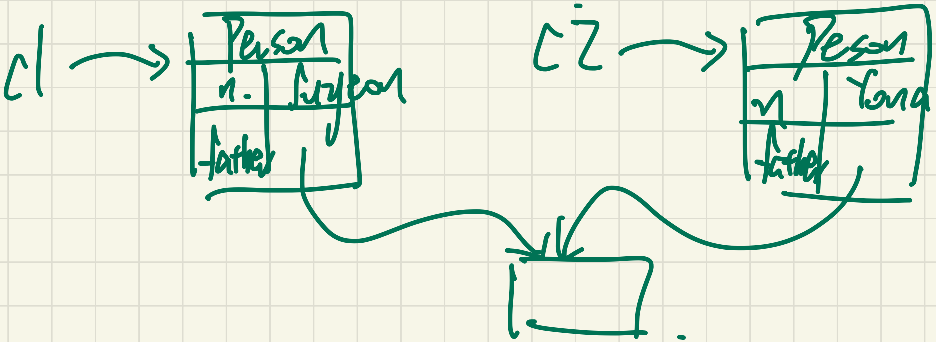
③ p0

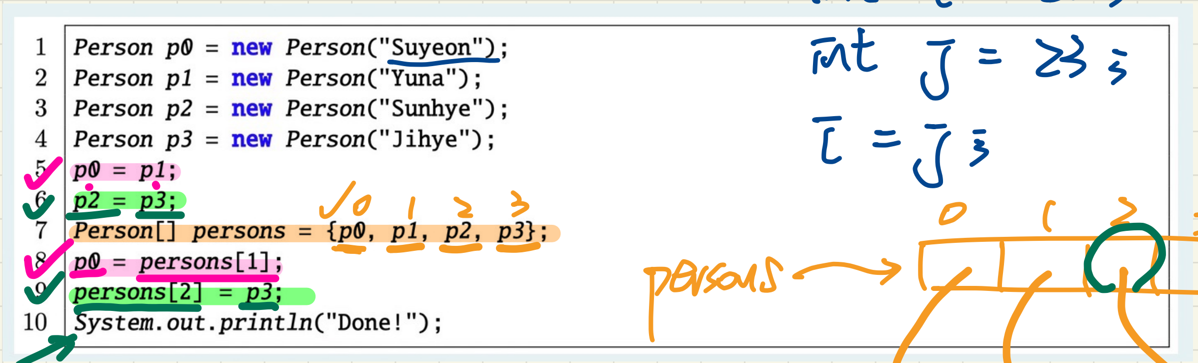
④ p2

⑤ persons2[1]

⑥ persons2[3]

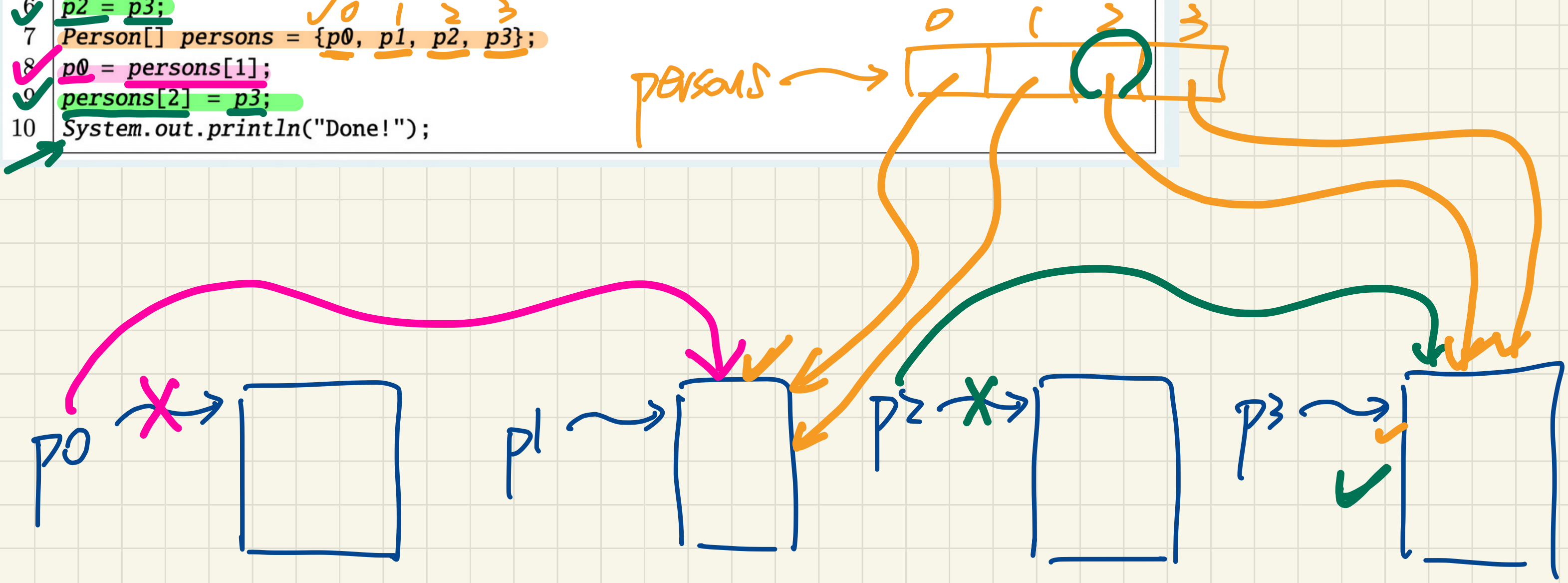


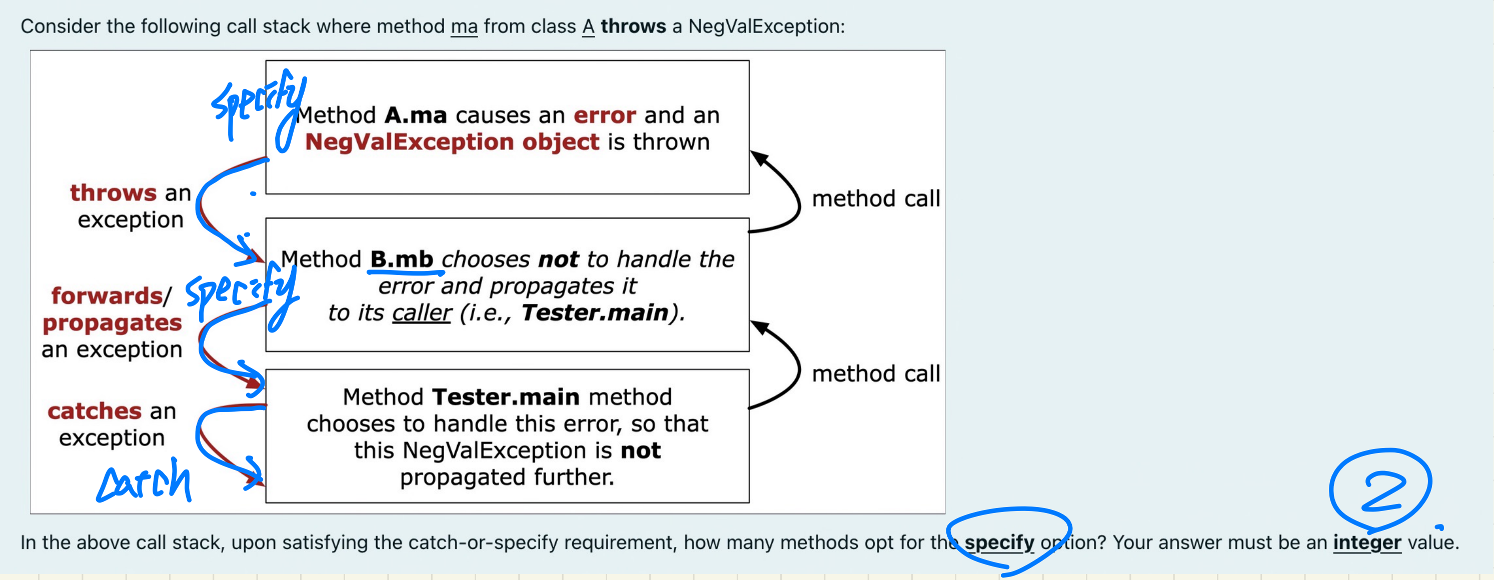

c1
Person
n.
Python
father
c2
Person
n
Yong
father



```java
Person p0 = new Person("Suyeon");
Person p1 = new Person("Yuna");
Person p2 = new Person("Sunhye");
Person p3 = new Person("Jihye");
p0 = p1;
p2 = p3;
Person[] persons = {p0, p1, p2, p3};
p0 = persons[1];
persons[2] = p3;
System.out.println("Done!");
```

```java
int i = 23;
int j = 23;
i = j;
```





Consider the following call stack where method ma from class A **throws** a NegValException:

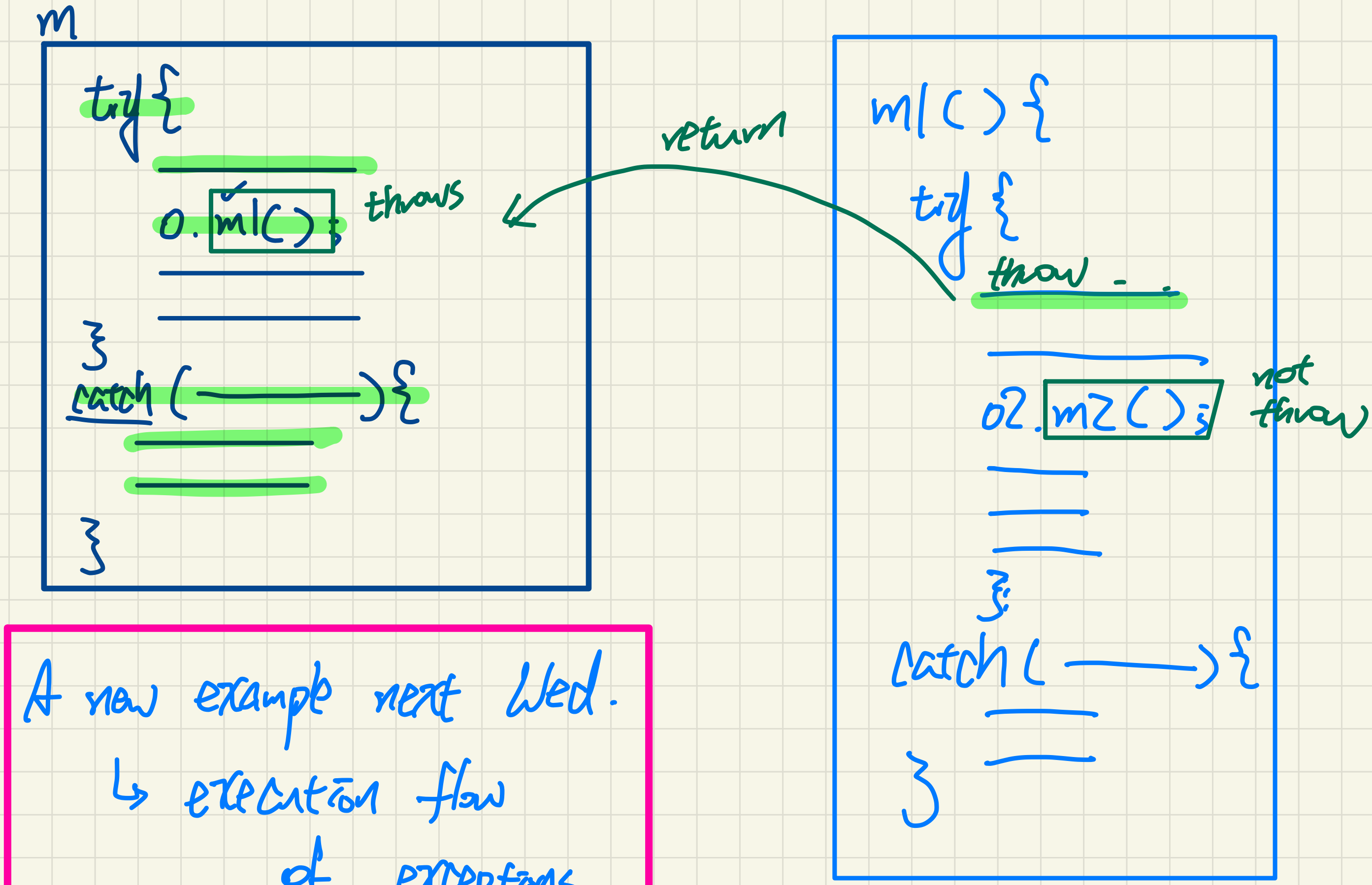
Method **A.ma** causes an **error** and an **NegValException object** is thrown

Method **B.mb** *chooses **not** to handle the error and propagates it to its caller (i.e., **Tester.main**).*

Method **Tester.main** method chooses to handle this error, so that this NegValException is **not** propagated further.

In the above call stack, upon satisfying the catch-or-specify requirement, how many methods opt for the **specify** option? Your answer must be an **integer** value.

2



A new example next Wed.

↳ execution flow of exceptions