Written Test 1
Review Session

Monday September 30

"F" == String literal

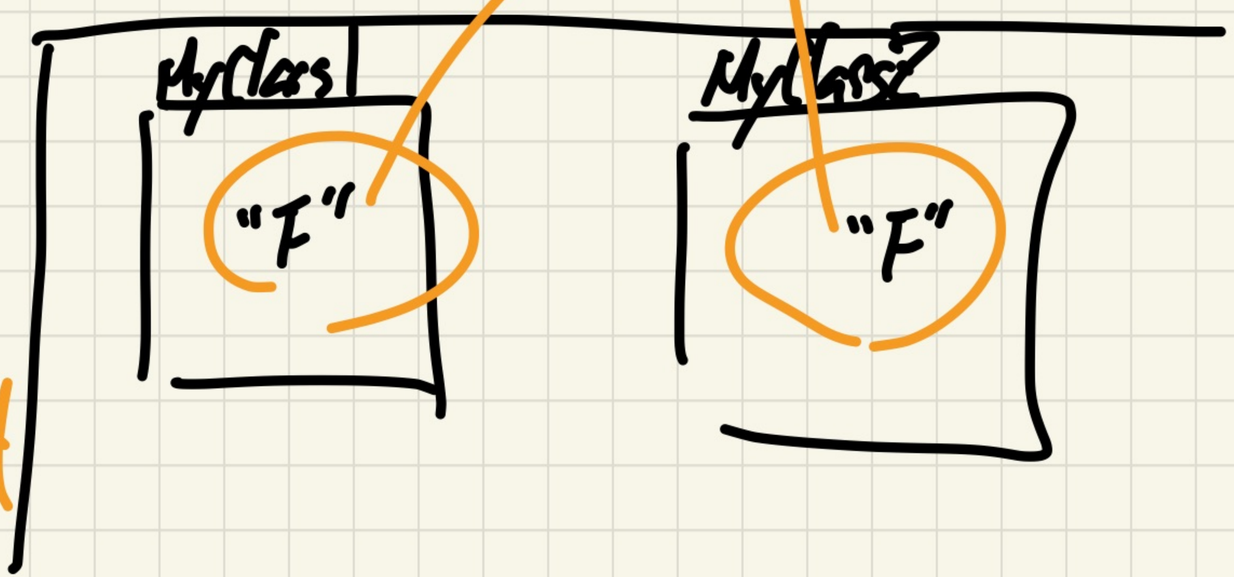
there's only a single object created for this literal

new String("F")

every time a new object is created

a single object

anonymous object



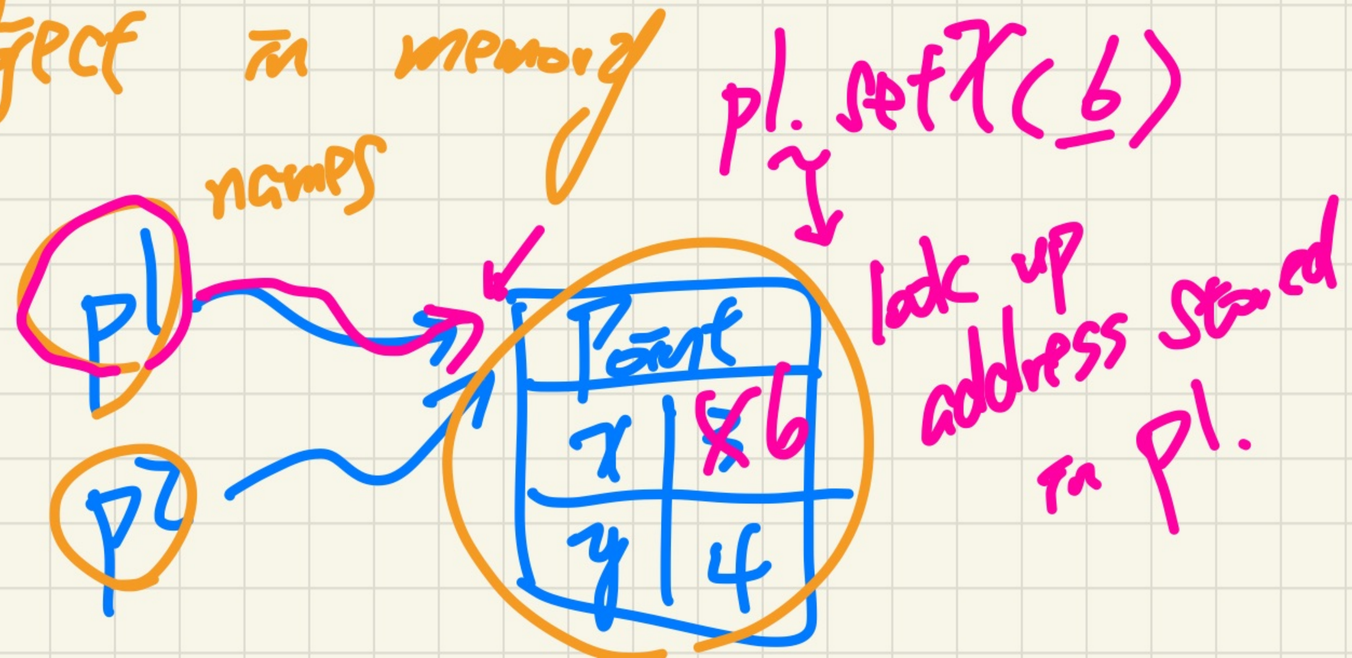
Anonymous Objects

objects for which you do not store their addresses in variables

```
Point p1 = new Point(3,4);
```

p1: name of variable storing address that object's

new Point(3,4): object in memory




```
Point p2 = p1;
```

When to use a.o.?

↳ When you only want to pass its address, without calling any methods on it.

```
class Point {

    Point newPoint(int x, int y) {
        Point rp = new Point(x, y);
        return rp;
    }
}
```

Point: return the address of some Point object

new Point(x, y): no method calls on rp.

return new Point(x, y);

Exceptions: Error handling

↳ force caller to "handle" errors when they occur.

catch-or-specify req.

class Account {

int balance;

Account(String name){ this.name = name; }

throws NAE

void withdraw(int a){

if(a<0 || a>balance){

throw new NAE("Error: neg. amount");

}

}

header very original source of NAE has the potential of throwing an exception

String name;

class NAE extends Exception{

:

}

↳1. Use eclipse to generate

✓1. Type this new class

catch option → usually not taken by the callee throwing the exception.

void withdraw(...){

try{ if(...){ throw new NAE(...)}

} catch (NAE ...){...}

}

class Client {

Account acc;

Client(String name){
this.acc = new Account(name);
}

/* charge the account of this client by amount 'a' */
void pay(int a){
acc.withdraw(a);
}

→ unhandle Exception

In order to see what methods we can call on "acc", go to the declare type

Specify → convenient for the current caller

void pay(int a) throws NAE {
acc.withdraw(a)
}

any caller of Client.pay must handle NAE

catch NAE

any caller of Client.pay does not have to handle

void pay(int a){
try{
acc.withdraw(a)
}catch(NAE e){ ... }
}

Tester. main

C1. m1

specify

call chain.

void m1() throws NAE {

C2. m2

specify

throws NAE

main(..){

try{

...

}catch{

:

}

}

void m2() throws NAE{

...

}

C3. m3

:

Tester.main

no longer need to handle NAE
∵ it's handled

L1.m1

void m1(){
try{ ... }
catch{ ... }
}

catch → first catch option is implemented

occasion when

specify

L2.m2

throws NAE

void m2() throws NAE{
...
}

L2.m3

throws → throws → throws

C1.m1

void m1() { throws NAE }

C2 o = new C2();

o.m2();

redundant

X

try { o.m2(); } catch (NAE ...) { ... }

C2.m2

void m2() throws NAE {

if(...){

throw new NAE(...);

}

}

```
void m1() throws E1, E2 {
    if(...) { throw new E1(...); }
    if(...) { throw new E2(...); }
    :
    :
}
```

Q7

Person[] personS = {p1, p2};

does not store Person addresses

Person[] persons = new Person[2];

Array of

Store address of a Person object

stores the starting address of an array

persons[0] = p1;

persons[1] = p2;

F

persons[0] == p2

persons[1] == p1

p1

p2

Person

n.

"Kleeyeon"

"Jiyeon"

persons

persons[0] == p1

persons[0] == p2

persons[1] == p1

int[] ns = {1, 2, 3};

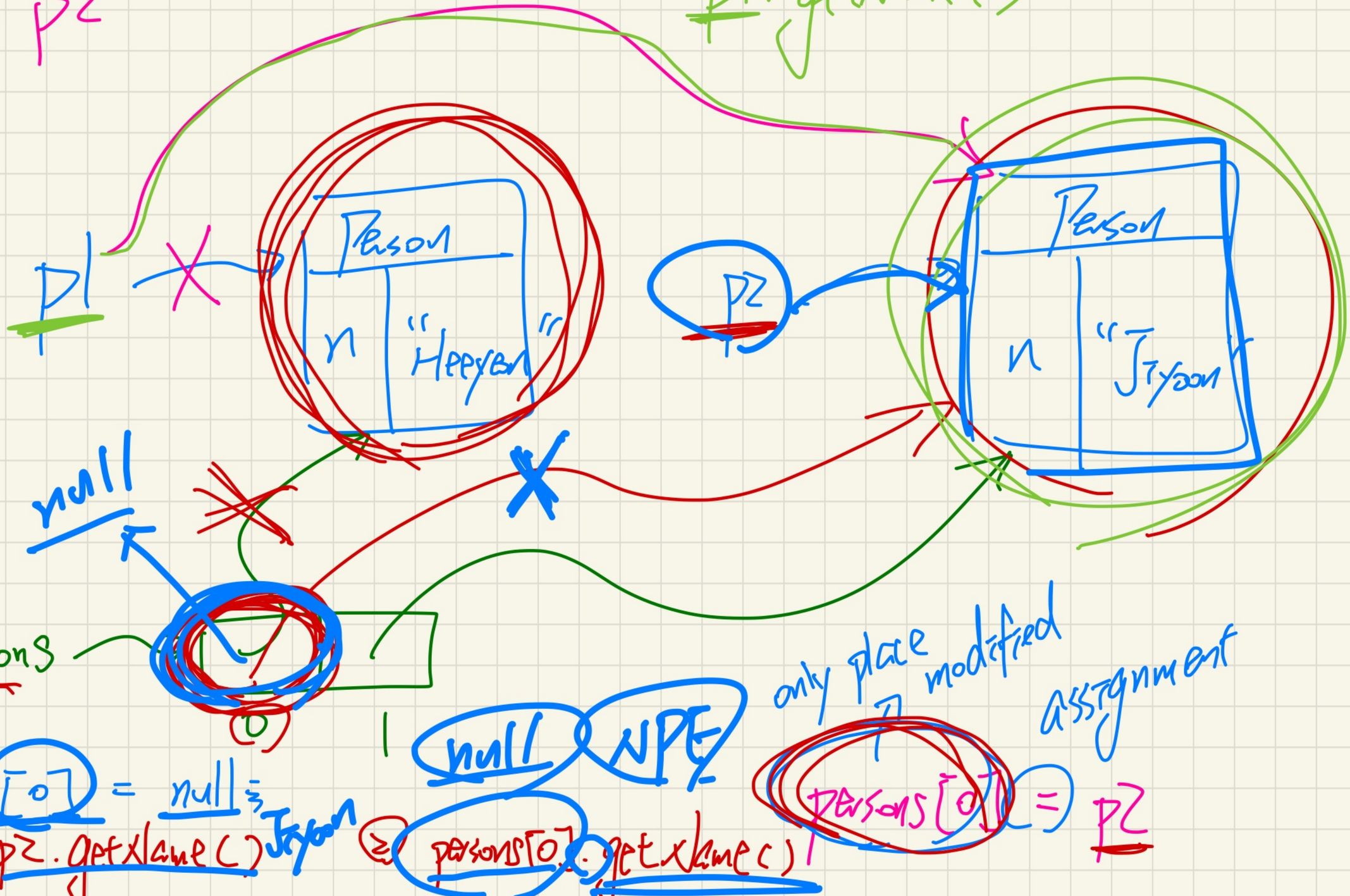
ns

Store starting address of the array

ns[0] ns[1] ns[2]

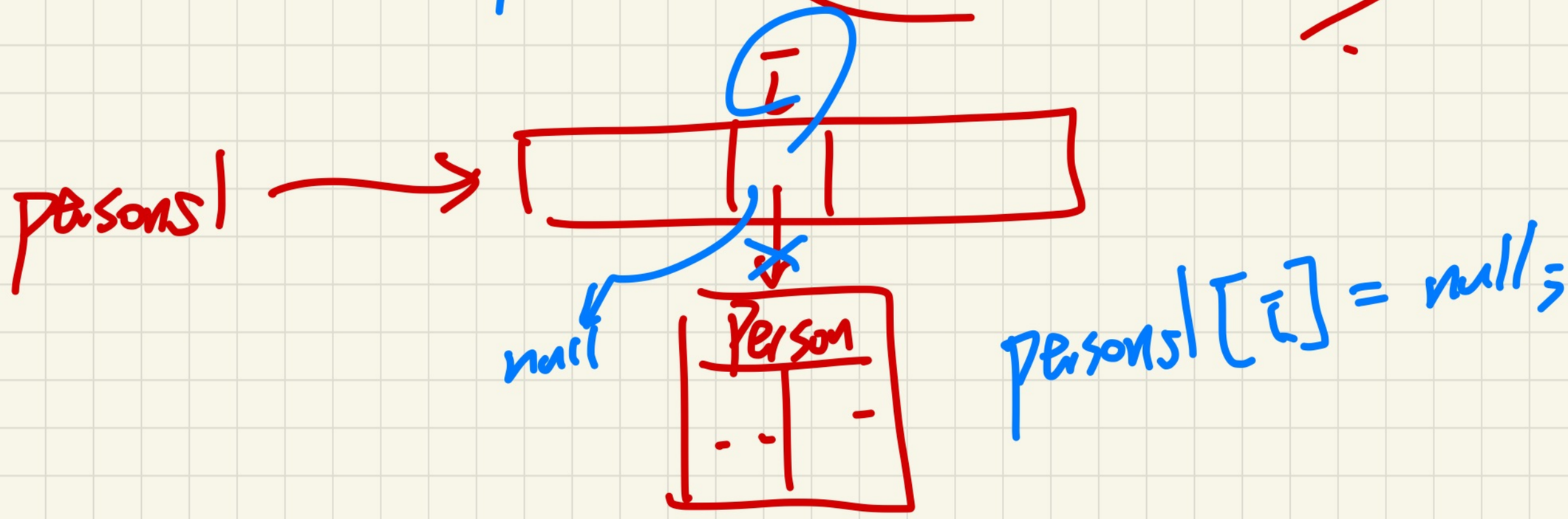
p1 = p2

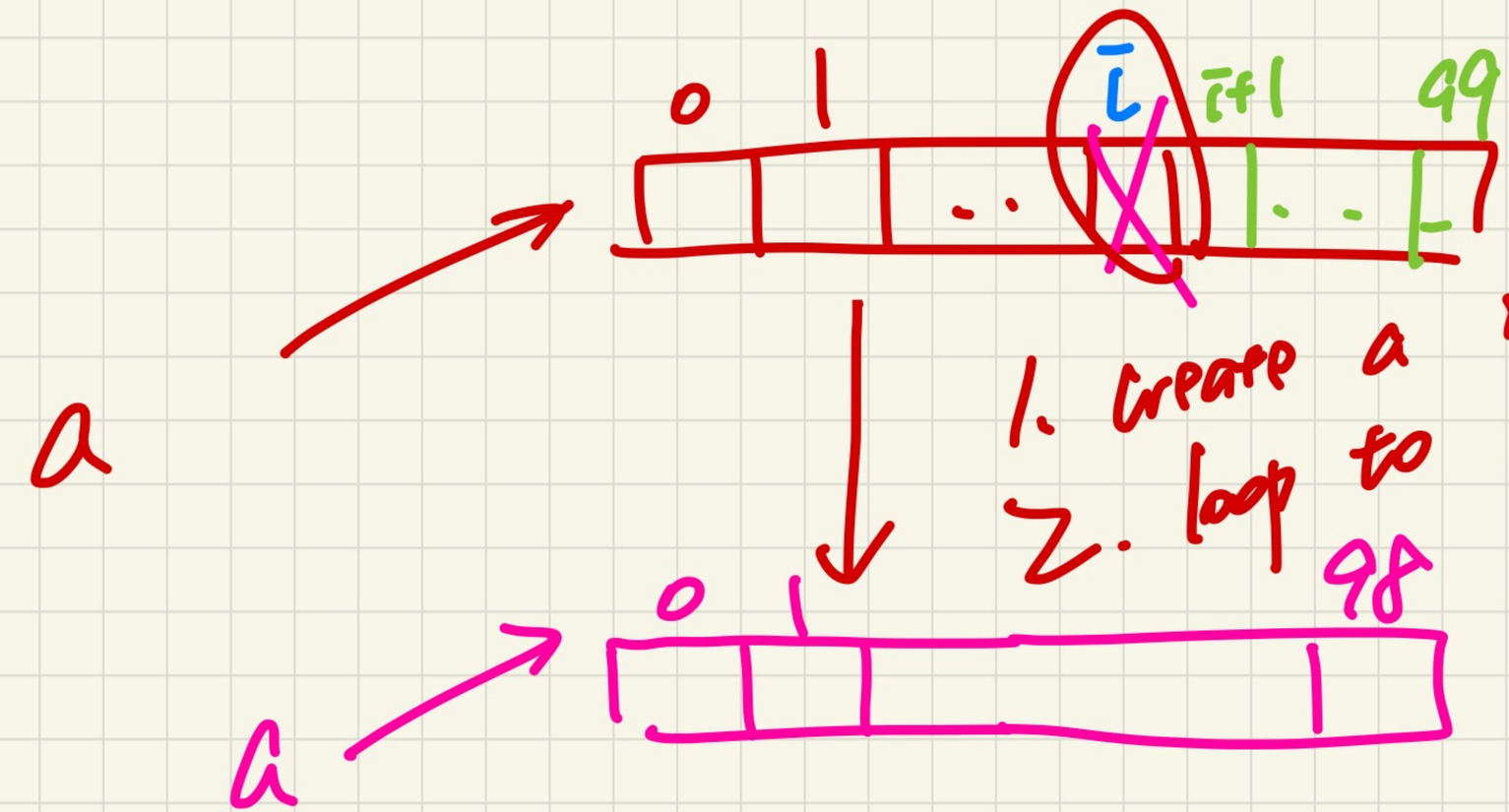
p1.getName()



```
Person[] persons1 = {...};

Person[] persons2 = new Person[3];
```



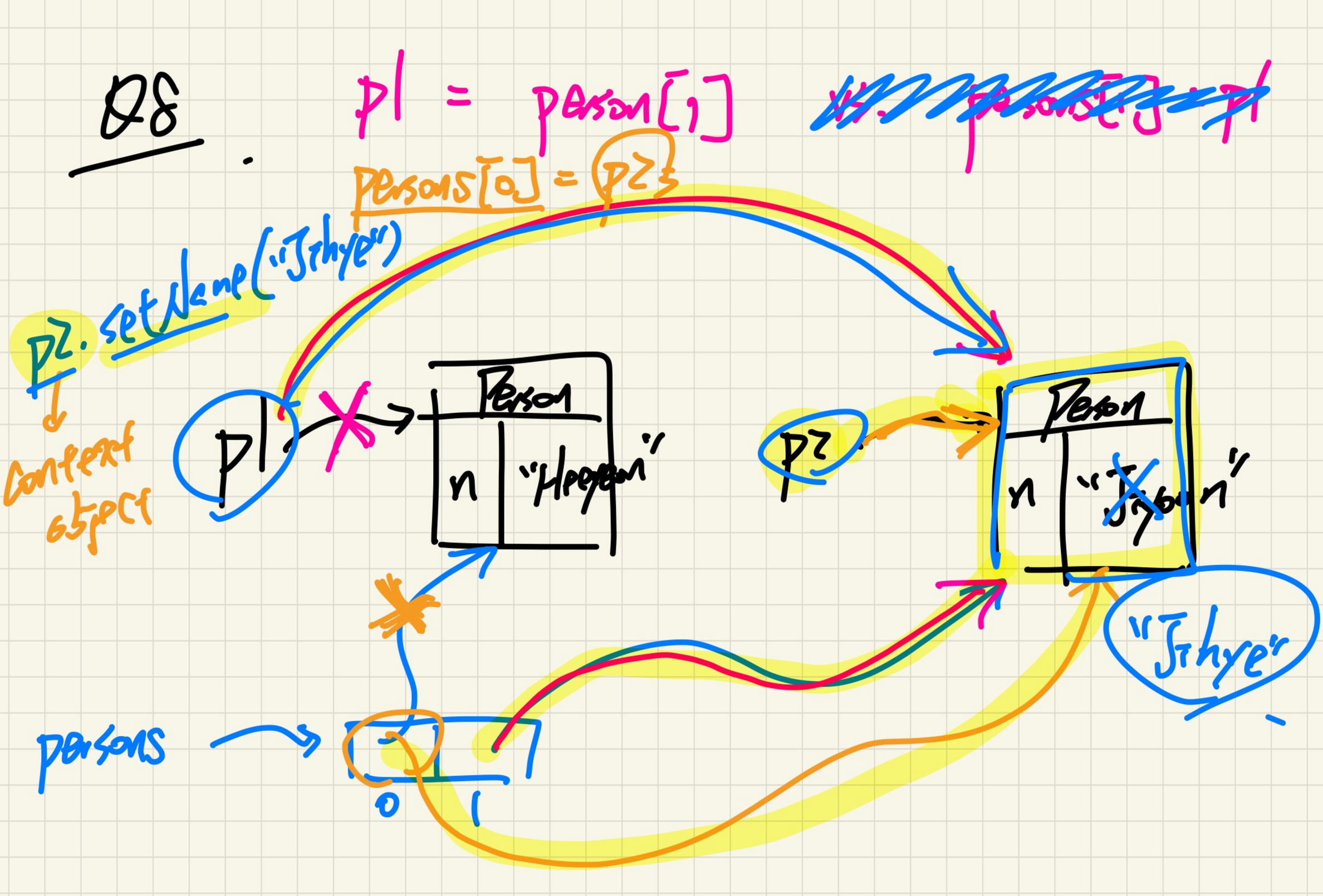

a
0
1
i
i+1
99
1. create a new, smaller array
2. loop to copy everything except a[i]
into new array.
a
0
98

08

p1 = person[1]

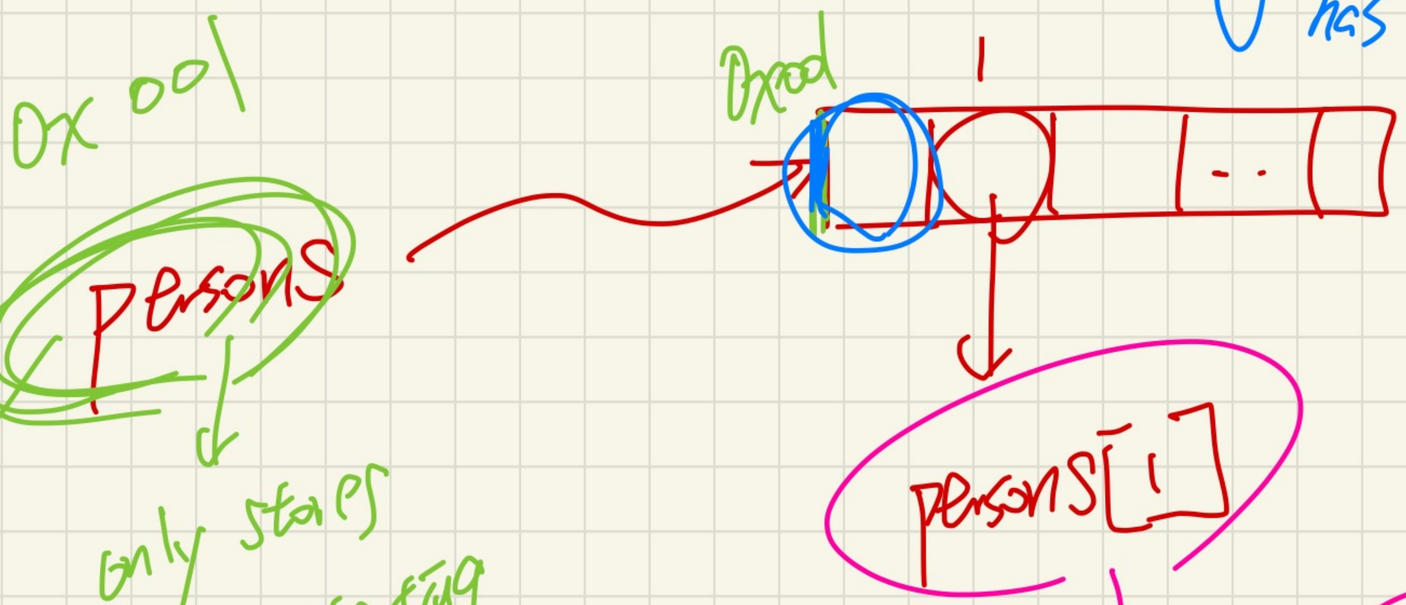
persons[0] = p2

p2.setName("Jihye")



Person[] persons = - - -

Starting address of the array has the same value as persons[0]



does not store any Person objects address

only stores the starting address of the array

stores a Person objects address