EECS2030 Advanced Object-Oriented Programming (Fall 2021)

Q&A - Lecture 3

Thursday, October 21

Announcement

- aggregation & composition
- copy constructor

- Lecture W6 (released: Oct. 18) 2pm EST.
- Lab3 (released: Oct. 20; due: Nov 1)
- Written Test 2 (due: Oct. 28/29)

Sir, instead of using three different if-statements, could we use one if-statement and use else if for the second and third boolean conditions? And putting the comparison and type casting at the else block? YES.

```
public class PointV2 {
  private int x;
  private int y;
  public PointV2 (int x, int y) { ... }
  public boolean equals(Object obj) {
    if(this == obj) { return true; }
    if(obj == null) { return false; }
    if(this.getClass() != obj.getClass()) { return false }
    Point other = (PointV2) obj;
    return this.x == other.x
        && this.y == other.y;
  }
}
```

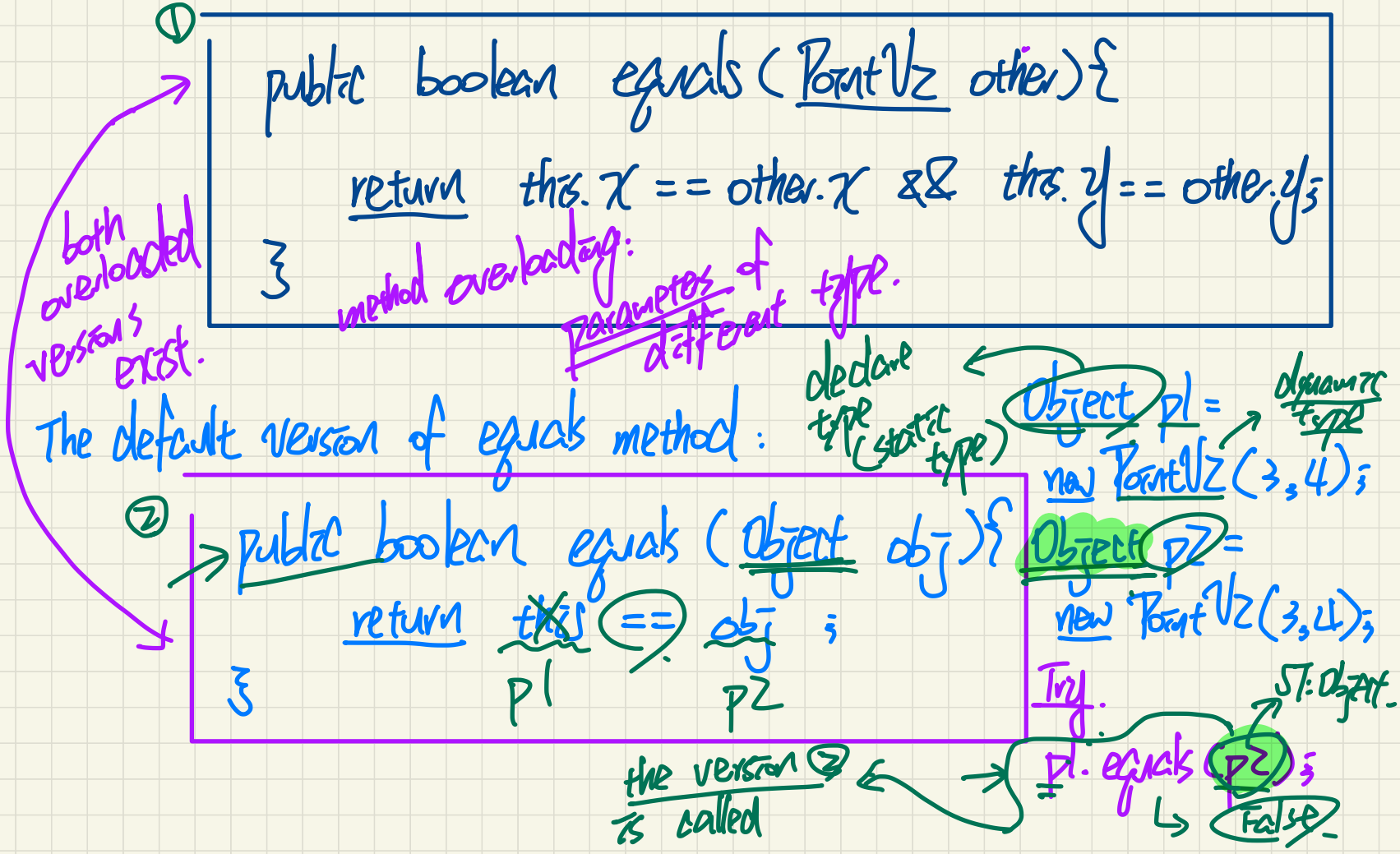
what if obj is null?

Single-Return Version

```
boolean equal;
if(①) { equal = true; }
else if(② || ③) { equal = false; }
else {
  PointV2 other = (PointV2) obj;
  equal = this.x == other.x
      && this.y == other.y;
}
return equal;
```

obj.x ✗ obj.y ✗

single return


①
public boolean equals(PointVz other){
return this.x == other.x && this.y == other.y;
}
method overloading: parameters of different type.
both overloaded versions exist.
The default version of equals method:
declare type (static type)
Object p1 = new PointVz(3,4);
dynamic type
②
public boolean equals(Object obj){
return this == obj;
}
p1
p2
Object p2 = new PointVz(3,4);
Try.
p1.equals(p2);
ST: Object.
the version ② is called
false

declared type (you would not call it static type)

```
int i = 3;
Product p = new Product(...);
```

Product → static type (never changed)

new Product(...) → dynamic type (changed for as many times as you like)

Object

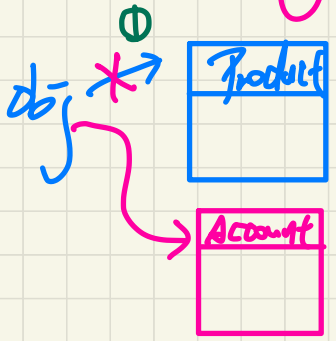
```
obj = new Product();
```

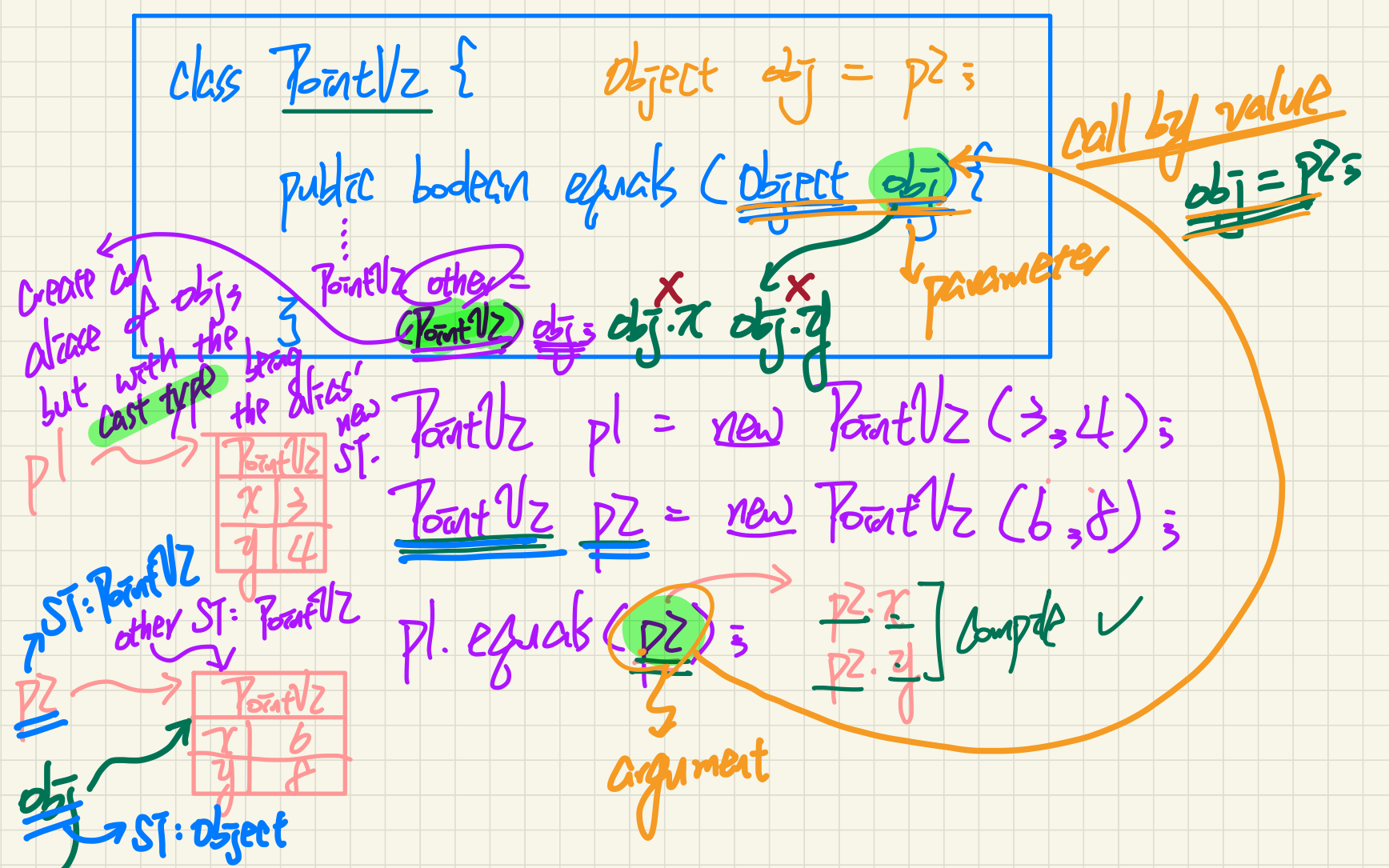
→ obj.getClass() → Product

```
obj = new Account();
```

→ obj.getClass() → Account

changing obj's dynamic type from Product to Account

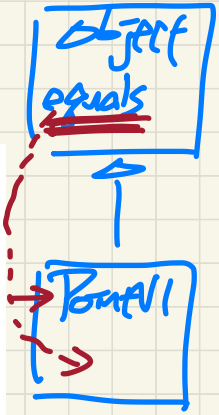



class PointVz {
Object obj = p2;
public boolean equals (Object obj) {
parameter
call by value
obj = p2;
PointVz other = (PointVz) obj;
Create an object but with the same type as the argument
cast type
obj.x
obj.y
p1
PointVz
x 3
y 4
new ST.
PointVz p1 = new PointVz (3,4);
PointVz p2 = new PointVz (6,8);
p1.equals (p2);
p2.x
p2.y
Compile ✓
argument
ST: PointVz
other ST: PointVz
p2
PointVz
x 6
y 8
obj
ST: Object

assertEquals: Reference Comparison or Not

assertEquals(exp1, exp2)

- ≈ `exp1.equals(exp2)` if exp1 and exp2 are ***reference*** type



Case 1: If `equals` is ***not*** explicitly overridden in `exp1`'s declared type
≈ ***assertSame***(exp1, exp2)

```
PointV1 p1 = new PointV1(3, 4);
PointV1 p2 = new PointV1(3, 4);
PointV2 p3 = new PointV2(3, 4);
assertEquals(p1, p2);
assertEquals(p2, p3);
```

Case 2: If `equals` is explicitly ***overridden*** in `exp1`'s declared type
≈ `exp1.equals(exp2)`

```
PointV1 p1 = new PointV1(3, 4);
PointV1 p2 = new PointV1(3, 4);
PointV2 p3 = new PointV2(3, 4);
assertEquals(p1, p2);
assertEquals(p2, p3);
assertEquals(p3, p2);
```

assertTrue(p1.equals(p2));
↳ p1 == p2