EECS2030 Advanced Object-Oriented Programming (Fall 2021)

Q&A - Lecture 3

Wednesday, October 20

Announcement

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

2. Copy Constructor & Composition

2pm EST Monday.

1. equal

So the default equals method only compares the id numbers? YES. (object addresses)

If id numbers match then it is true? (Part B; timestamp 9:00)

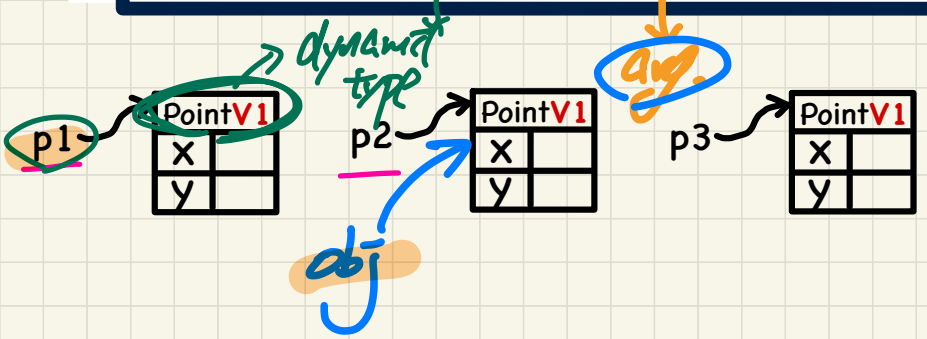
```
public class Object {
    public boolean equals(Object obj) {
        return this == obj;
    }
}
```

```
1  String s = "(2, 3)";
2  PointV1 p1 = new PointV1(2, 3);
3  PointV1 p2 = new PointV1(2, 3);
4  PointV1 p3 = new PointV1(4, 6);
5  System.out.println(p1 == p2);   /* false */
6  System.out.println(p2 == p3);   /* false */
7  System.out.println(p1.equals(p1));   /* true */
8  System.out.println(p1.equals(null));   /* false */
9  System.out.println(p1.equals(s));   /* false */
10 System.out.println(p1.equals(p2));   /* false */
11 System.out.println(p2.equals(p3));   /* false */
```

call by value: obj = p2

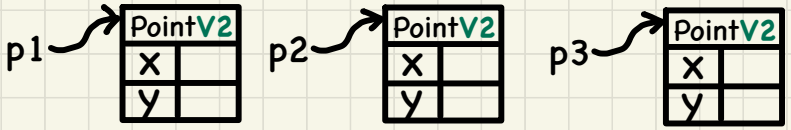
extends

```
public class PointV1 {
    private int x;
    private int y;
    public PointV1 (int x, int y) {
        this.x = x;
        this.y = y;
    }
}
```



Why wouldn't you use .equals to figure out dynamic.
if this and obj are objects of the same type?
Wouldn't != only show you if the objects are not of the same type?
(Part C2; timestamp 9:00)

```
1  String s = "(2, 3)";
2  PointV2 p1 = new PointV2(2, 3);
3  PointV2 p2 = new PointV2(2, 3);
4  PointV2 p3 = new PointV2(4, 6);
5  System.out.println(p1 == p2);   /* false */
6  System.out.println(p2 == p3);   /* false */
7  System.out.println(p1.equals(p1));   /* true */
8  System.out.println(p1.equals(null));   /* false */
9  System.out.println(p1.equals(s));   /* false */
10 System.out.println(p1.equals(p2));   /* true */
11 System.out.println(p2.equals(p3));   /* false */
```




```
public class Object {
  ...
  public boolean equals(Object obj) {
    return this == obj;
  }
}
```

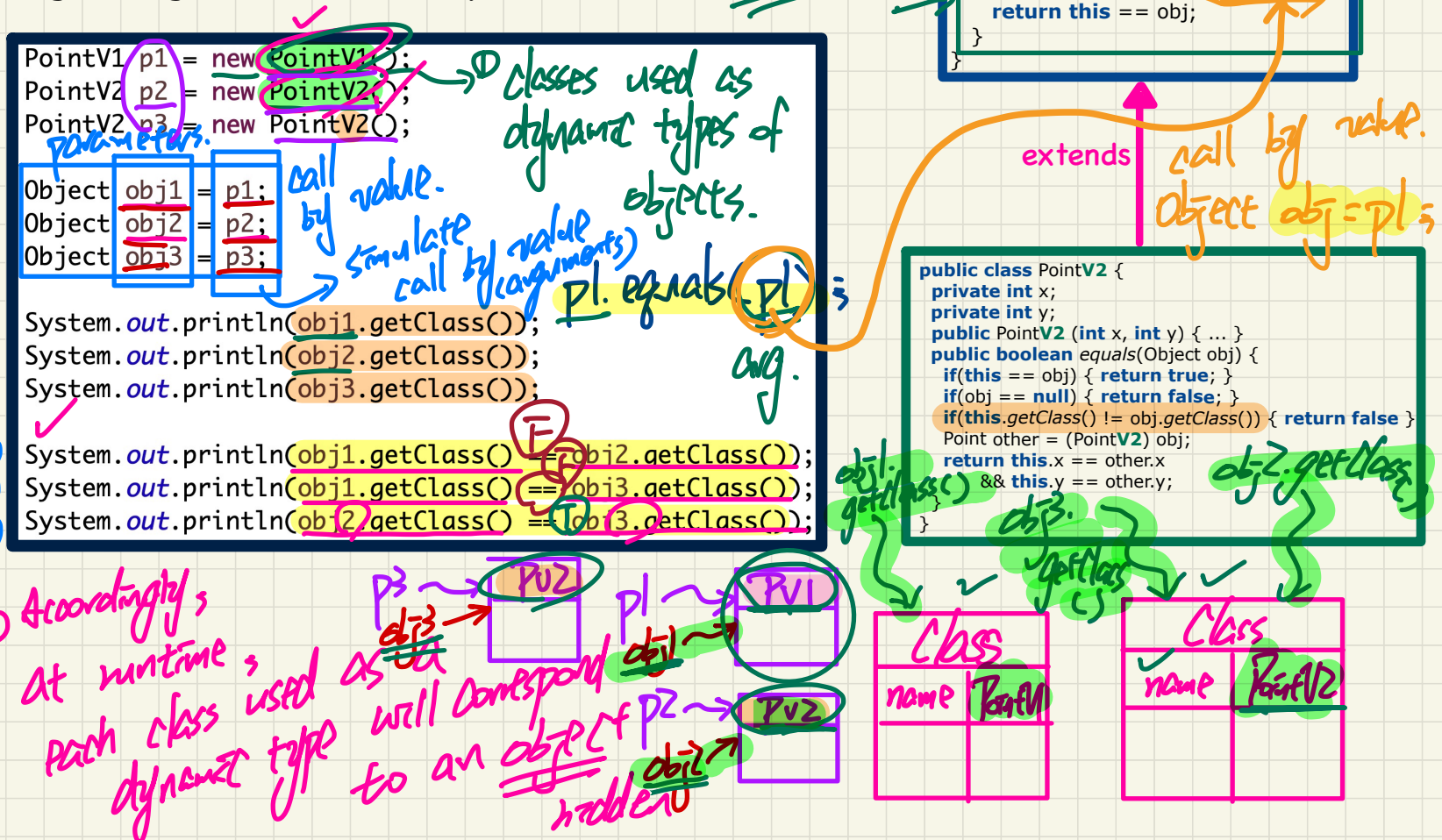
if this boolean exp evaluates to true, then

extends

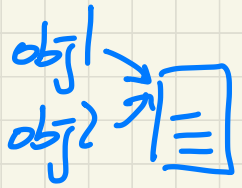
this and obj do not have the same dynamic type ⇒ they cannot be equal

```
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 }
    PointV2 other = (PointV2) obj;
    return this.x == other.x
        && this.y == other.y;
  }
}
```

Figuring Dynamic Types



At Part C5; 12:23, doesn't reference equal just mean that they are both pointing at an object with the same dynamic type, (e.g. Pointv2) or is that a different term/idea?



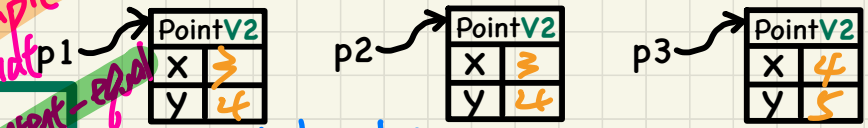

```java
public class Object {
  ...
  public boolean equals(Object obj) {
    return this == obj;
  }
}
```

extends

```java
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;
  }
}
```

a counter example shows that two contents-equal objects may have diff. addresses

```java
1 PointV2 p1 = new PointV2(3, 4);
2 PointV2 p2 = new PointV2(3, 4);
3 PointV2 p3 = new PointV2(4, 5);
4 System.out.println(p1 == p1);      /* true */
5 System.out.println(p1.equals(p1)); /* true */
6 System.out.println(p1 == p2);      /* false */
7 System.out.println(p1.equals(p2)); /* true */
8 System.out.println(p2 == p3);      /* false */
9 System.out.println(p2.equals(p3)); /* false */
```



obj1 obj2

(A) Two objects are reference-equal.

(B) Two objects are contents-equal.

obj1 == obj2

obj1.equals(obj2)

customized.

- If (A) is true, then (B) is true.
- If (B) is true, then (A) is true.

always true.

not always true.

Why do you need a cast (point) on obj?

Is it because you can't know for sure what the dynamic type of obj is?

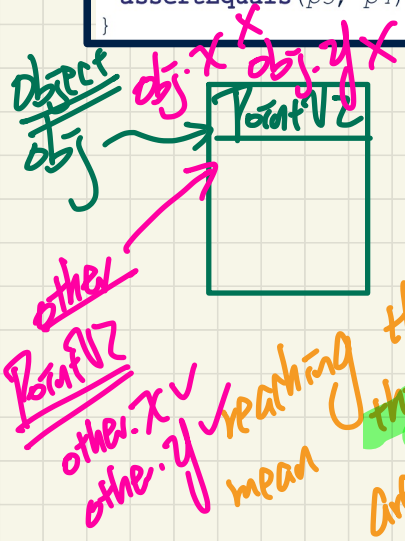
(Part E; timestamp 8:50)

```java
@Test
public void testEqualityOfPointV2() {
  PointV2 p3 = new PointV2(3, 4); PointV2 p4 = new PointV2(3, 4);
  assertFalse(p3 == p4); assertFalse(p4 == p3);
  /* assertSame(p3, p4); assertSame(p4, p4); */ /* both fail */
  assertTrue(p3.equals(p4)); assertTrue(p4.equals(p3));
  assertEquals(p3, p4); assertEquals(p4, p3);
}
```

```java
public class Object {
  ...
  public boolean equals(Object obj) {
    return this == obj;
  }
}
```

extends

```java
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 }
    PointV2 other = (PointV2) obj;
    return this.x == other.x
        && this.y == other.y;
  }
}
```



Static vs Dynamic Types

Account arc = new Account();

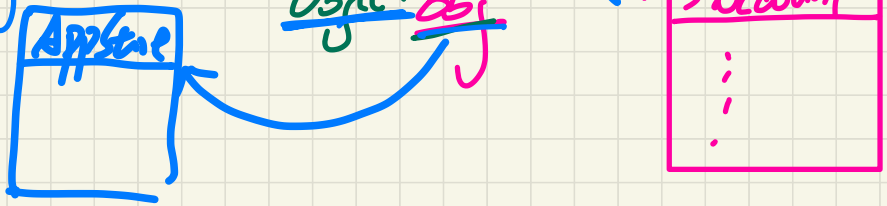
Account → static type (never changes)

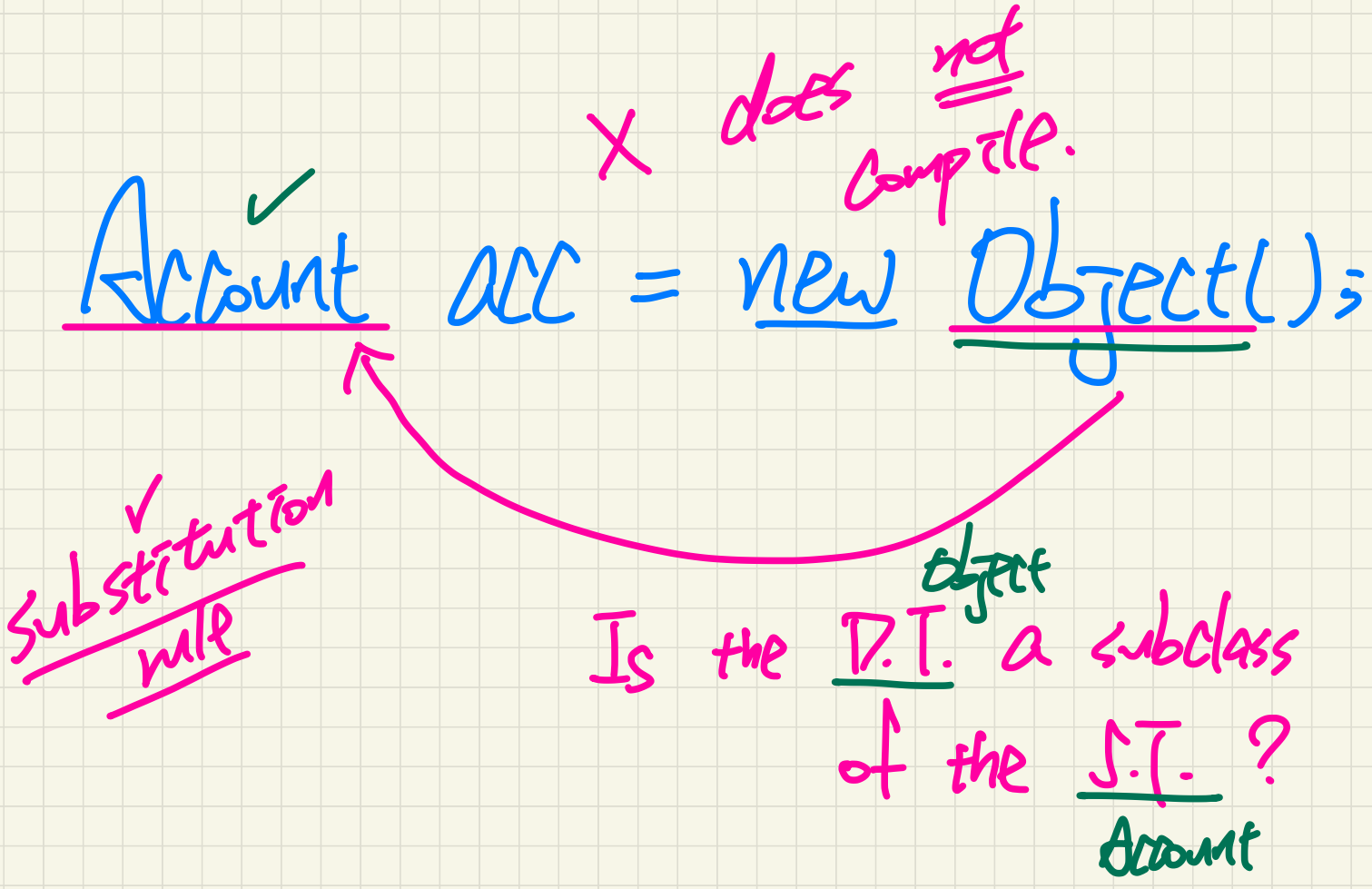
Account() → dynamic type. (may change at any time)

Object obj = new Account();

obj = new AppStore();

Object → parent/super class of every class




X does not compile.
Account acc = new Object();
Substitution rule
Is the D.T. Object a subclass of the S.T. ?
Account