Lecture 15 - Oct 29

Object Equality

Overridden equals: Phase 4
Short-Circuit, Logical Implication
JUnit: assertSame vs assertEquals

Announcements/Reminders

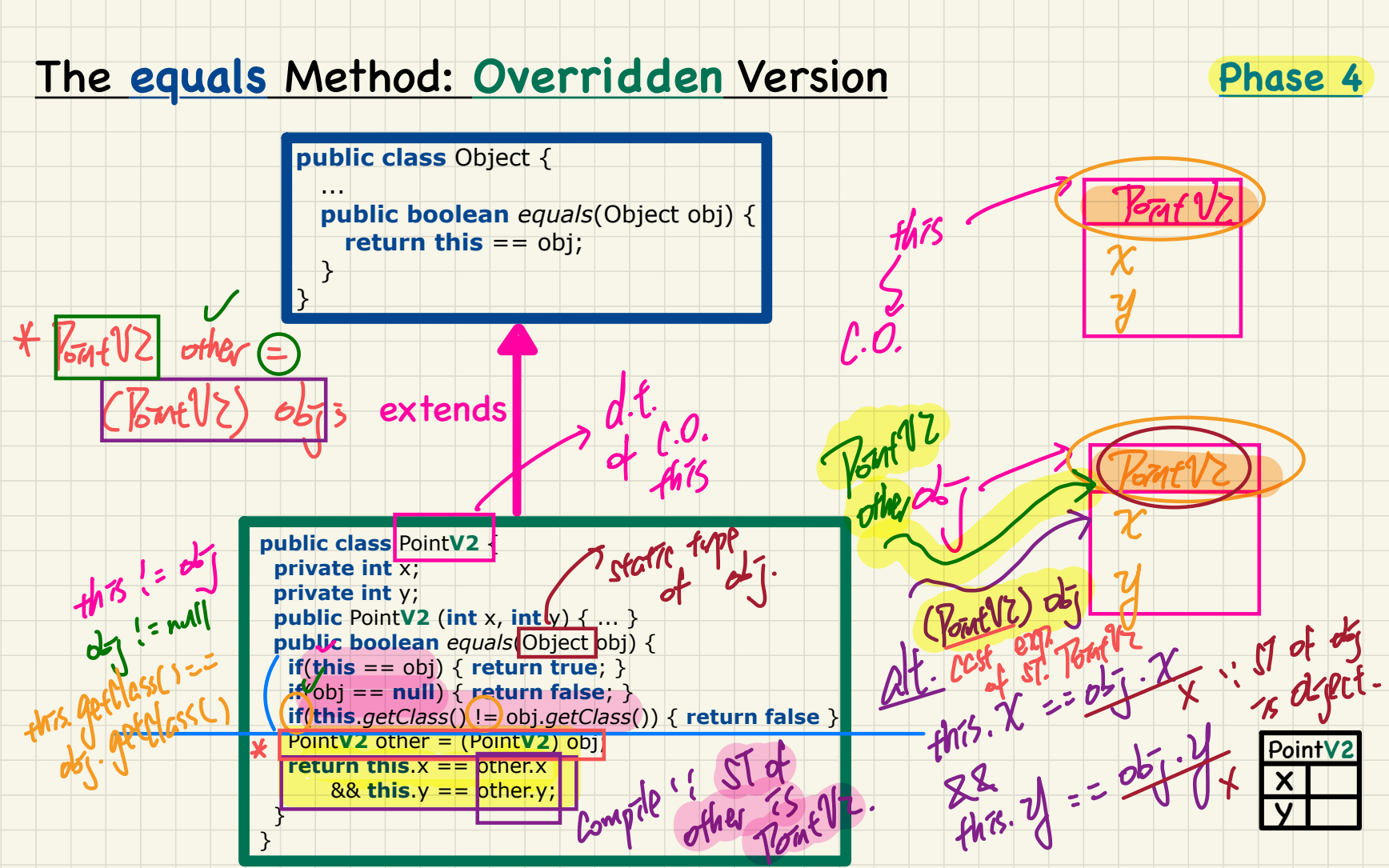
- Today's class: notes template posted
- ProgTest0 and ProgTest1 marks and feedback released
- Lab3 released

The equals Method: Overridden Version

Phase 4

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

```
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;
    }
}
```



The equals Method: Overridden Version

Example 1: Trace L10

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

extends

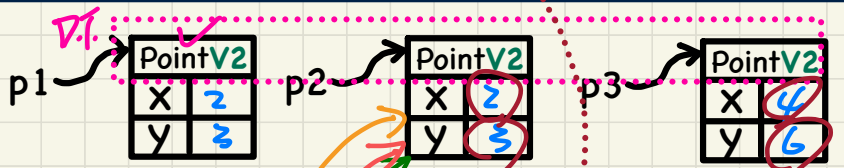
```
public class PointV2 {
   private int x;
   private int y;
   public PointV2 (int x, int y) { ... }
   public boolean equals(Object obj) {
      if(this == obj) { return true; }
      if(obj == null) { return false; }
      if(this.getClass() != obj.getClass()) { return false }
      PointV2 other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

bypassed

```
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);   /* [redacted] */
6  System.out.println(p2 == p3);   /* [redacted] */
7  System.out.println(p1.equals(p1));   /* [redacted] */
8  System.out.println(p1.equals(null));   /* [redacted] */
9  System.out.println(p1.equals(s));   /* [redacted] */
10 System.out.println(p1.equals(p2));   /* [redacted] */
11 System.out.println(p2.equals(p3));   /* [redacted] */
```

D.T.

tmp.



Object obj

PointV2 other

(PointV2) obj

return false

Exercise

The equals Method: Overridden Version

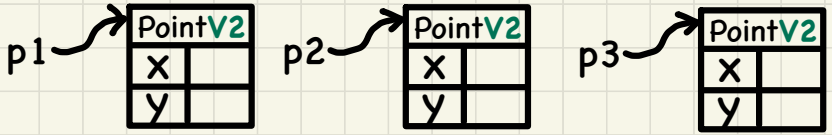
Example 1: Trace L11

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

extends

```
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;
  }
}
```

```
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);   /* [redacted] */
6  System.out.println(p2 == p3);   /* [redacted] */
7  System.out.println(p1.equals(p1));   /* [redacted] */
8  System.out.println(p1.equals(null));   /* [redacted] */
9  System.out.println(p1.equals(s));   /* [redacted] */
10 System.out.println(p1.equals(p2));   /* [redacted] */
11 System.out.println(p2.equals(p3));   /* [redacted] */
```



The equals Method:

To Override or Not?

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

extends

extends

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

```
public class PointV2 {
   private int x; double y;
   public PointV2 (double x, double y) { ... }
   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;
   }
}
```

```
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 */
```

D.T.

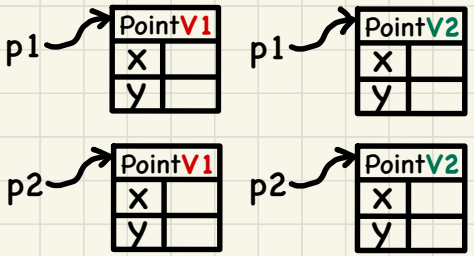
default version from Object is called

⇒ p1 == p2

```
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 */
```

D.T.

overridden version from PointV2 is called



```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;
  }
}
```

① ② ③

risk of NPE

a ☐ if (① && ②) { return false; }

b ☐ if (② && ①) { rn. false; }

c ☑ if (① || ②) { rn. false; }

↳ if true, skip eval. ②

d ☐ if (② || ①) { rn. false; }

if obj is null NPE would've occurred before evaluating ①.

The equals Method: Overridden Version

Example 2

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

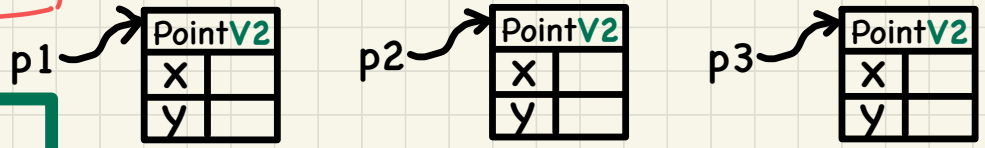
```
1  PointV2 p1 = new PointV2(3, 4);
2  PointV2 p2 = new PointV2(3, 4);
3  PointV2 p3 = new PointV2(4, 5);
4  System.out.println(p1 == p1);   /* [redacted] */
5  System.out.println(p1.equals(p1));   /* [redacted] */
6  System.out.println(p1 == p2);   /* [redacted] */
7  System.out.println(p1.equals(p2));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

F
T

(A)
(B)

extends

obj1
obj2




```
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;
  }
}
```

obj1 == obj2

(A) Two objects are reference-equal.

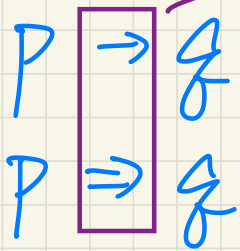
(B) Two objects are contents-equal.

obj1.eq.(obj2)

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

not valid.

Logical Implication



false when:
p is true
q is false.

it rains → outdoor ground wet

it rains (T)
outdoor ground wet (F)
→ implication is false.

assertSame vs. assertEquals

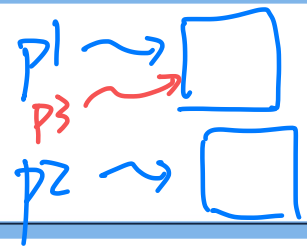
pass if: exp1 == exp2
fail if: exp1 != exp2

assertSame(exp1, exp2) — any objects

- Passes if exp1 and exp2 are references to the same object
  - ≈ **assertTrue**(exp1 == exp2)
  - ≈ **assertFalse**(exp1 != exp2)

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

p1 == p3 → pass
p2 == p3 → fail



assertEquals(exp1, exp2)

- ≈ `exp1 == exp2` if exp1 and exp2 are **primitive** type

```
int i = 10;
int j = 20;
assertEquals(i, j);
```

i == j → fail

ref types : any type

assertEquals(expr1 , expr2)

↳ assertTrue(expr1.equals(expr2))

① assertEquals(expr1 , expr2)

↳ assertTrue(expr1.equals(expr2))

version in expr1's D.T.

② assertEquals(expr2 , expr1)

↳ assertTrue(expr2.equals(expr1))

↳ version in expr2's D.T.

assertEquals: Reference Comparison or Not

* assertT(p1.equals(p2))
** assertT(p2.equals(p3))

assertEquals(exp1, exp2)

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

p1 == p2

" p2 == p3 "

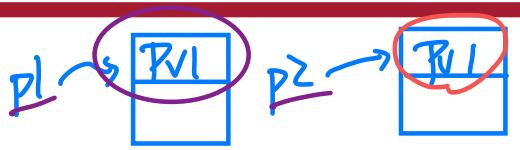
Case 1: If `equals` is ***not*** explicitly overridden in exp1's declared type

≈ ***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);
```

false

false



Case 2: If `equals` is explicitly ***overridden*** in exp1's declared type

≈ `exp1.equals(exp2)`

** phase 3

both rn. false but for diff reasons

```
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);
```

false

false

p1.eq.(p2)
p2.eq.(p3)

* p3.eq.(p2)
↳ overridden ver.
↳ rn. false if diff. R.T. **

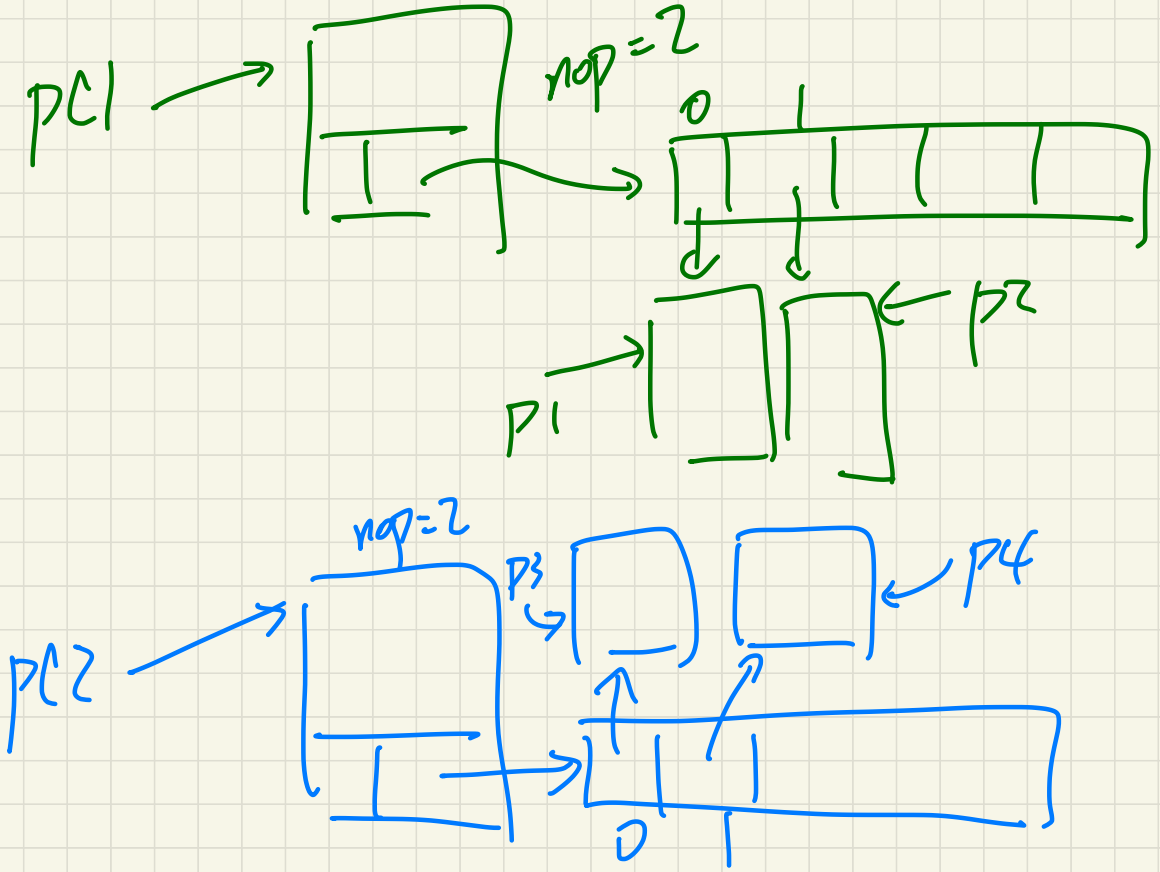
Exercise: PersonCollectors are equal if their arrays of persons are equal

```
class PersonCollector {
  private Person[] persons;
  private int nop; /* number of persons */
  public PersonCollector() { ... }
  public void addPerson(Person p) { ... }
  public int getNop() { return this.nop; }
  public Person[] getPersons() { ... }
}
```

```
1  public boolean equals(Object obj) {
2    if(this == obj) { return true; }
3    if(obj == null || this.getClass() != obj.getClass()) { return false; }
4    PersonCollector other = (PersonCollector) obj;
5    boolean equal = false;
6    if(this.nop == other.nop) {
7      equal = true;
8      for(int i = 0; equal && i < this.nop; i ++) {
9        equal = this.persons[i].equals(other.persons[i]);
10     }
11   }
12   return equal;
13 }
```

Q: At Line 9 of PersonCollector's equals method which version of the equals method is called?

```
1  public class Person {
2    private String firstName; private String lastName;
3    private double weight; private double height;
4    public boolean equals(Object obj) {
5      if(this == obj) { return true; }
6      if(obj == null || this.getClass() != obj.getClass()) { return false; }
7      Person other = (Person) obj;
8      return
9          this.weight == other.weight
10       && this.height == other.height
11       && this.firstName.equals(other.firstName)
12       && this.lastName.equals(other.lastName);
13   }
14 }
```


pc1
nop=2
0
1
p1
p2
pc2
nop=2
p3
p4
0
1