Lecture 12 - Oct. 22

Object Equality

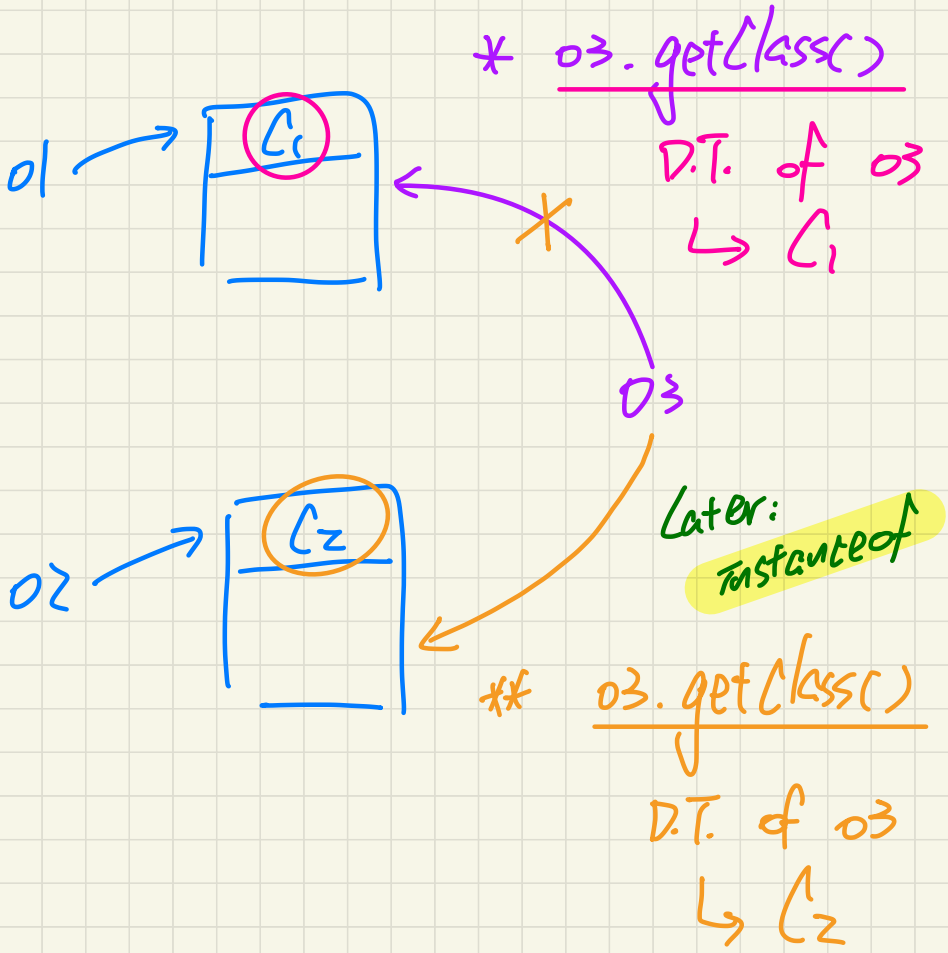
Reference Equality vs. Object Equality
JUnit: assertSame vs. assertEquals

Announcements/Reminders

- ProgTest1 results to be released by next Monday
- Lab2 due this Friday
- ProgTest2 on Wednesday, October 30

  + PDF Guide released

```
C1 o1 = new C1();
C2 o2 = new C2();
o1.m1();
o1.m2();
o2.m1();
o2.m2();
Object o3;
o3 = o1; *
o3.m1;
o3.m2;
((C1) o3).m1();
((C1) o3).m2();
o3 = o2; **
((C2) o3).m1();
((C2) o3).m2();
```



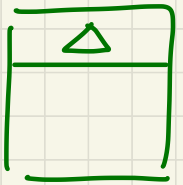
The equals Method: Overridden Version

Phase 3

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

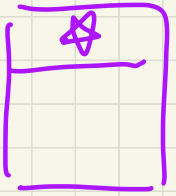
extends

this →



```
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;
  }
}
```

obj →



if the two objects have distinct D.T.s, no comparison → false.

PointV2	
x	
y	

The equals Method: Overridden Version

Example 1: Trace L9

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

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

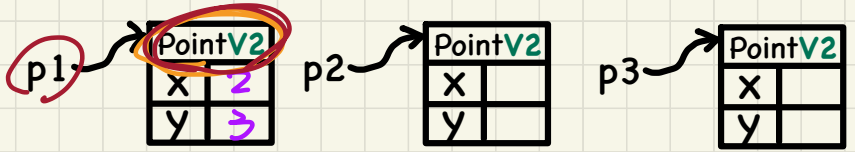
PointV2 extends Object

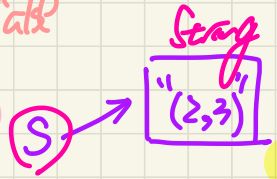
p1.getClass() s.getClass() String

```
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;
  }
}
```

Exercise: if inputs equal. to false

"PointV2" != String

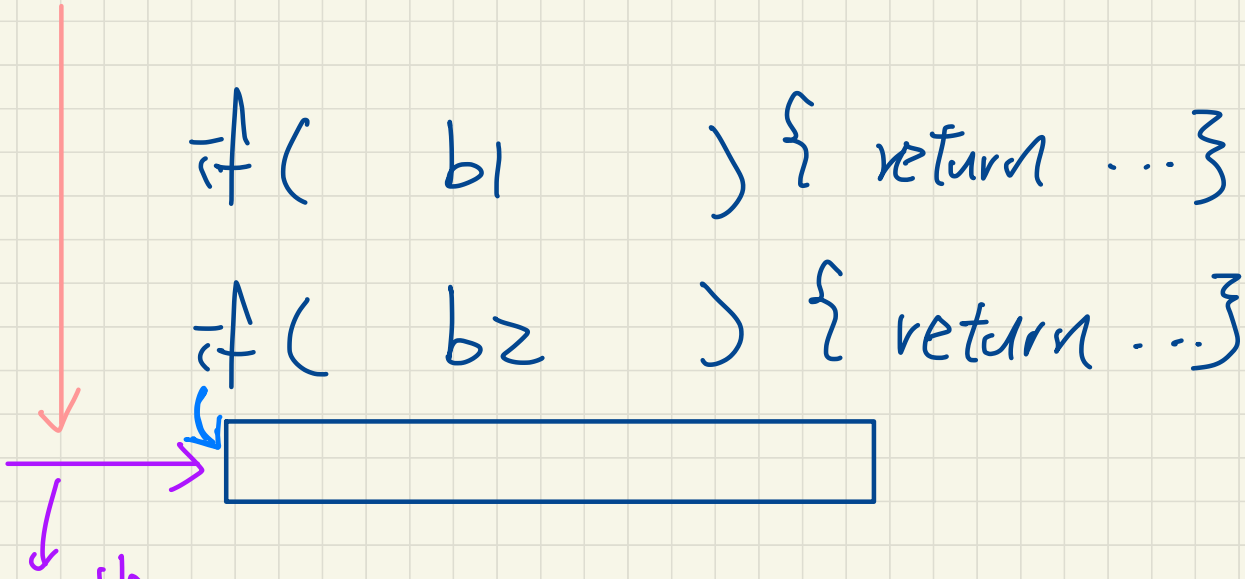




p1.equals(s)

D.T. of C.O. is PointV2

↳ invoke version of equals from PointV2


if (b1) { return ... }
if (b2) { return ... }
being able to reach this line:
!b1 && !b2
!(b1 || b2)

The equals Method: Overridden Version

Phase 4

```
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;
  }
}
```

ST of obj: Object
↳ can only call Object attributes on obj.

① this != obj
② obj != null
③ this and obj have same D.T.

** (PointV2) obj).x == this.x &&

* return this.x == obj.x && this.y == obj.y;

ST: Object

this → PointV2

obj, other → PointV2 (x, y)

other: PointV2 ST

PointV2
x	
y	

The equals Method: Overridden Version

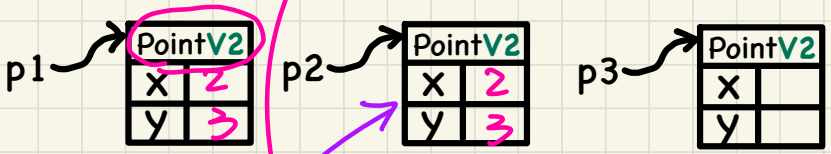
Example 1: Trace L10

```
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;
  }
}
```

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



D.T. of p1 : PointV2

other
ST: PointV2

The equals Method: Overridden Version

Example 1: Trace L11

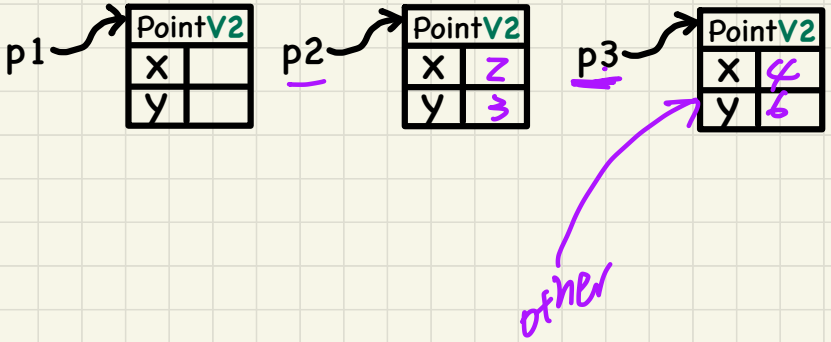
```
public class Object {
   ...
   public boolean equals(Object obj) {
      return this == obj;
   }
}
```

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;
  }
}
```

false

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



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;
   }
}
```

* default equals invoked

↳ p1 == p2

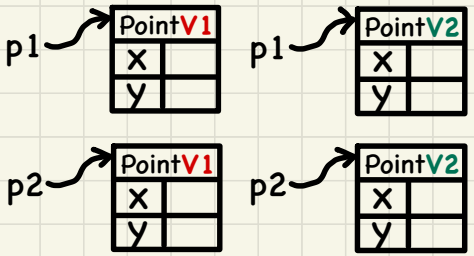
** overridden equals invoked

↳ p1.equals(p2)

↳ p1.x == p2.x && p1.y == p2.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 */
```

```
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 */
```



The equals Method: Overridden Version

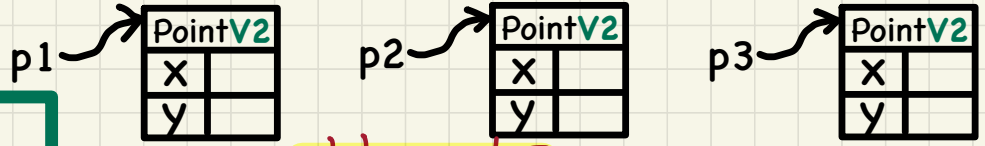
Example 2

```
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  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);          /*      */
5  System.out.println(p1.equals(p1));     /*      */
6  System.out.println(p1 == p2);          /*      */
7  System.out.println(p1.equals(p2));     /*      */
8  System.out.println(p2 == p3);          /*      */
9  System.out.println(p2.equals(p3));     /*      */
```



obj1 == obj2

(A) Two objects are reference-equal.

(B) Two objects are contents-equal.

obj1.equals(obj2)

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

ref. eq.

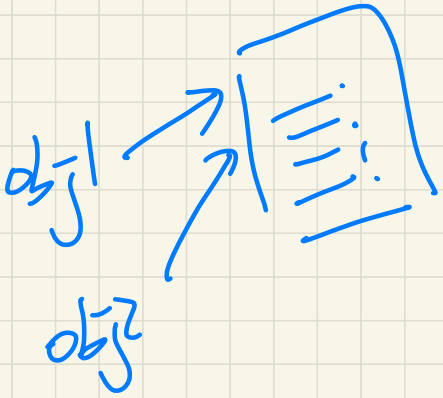
obj. eq.

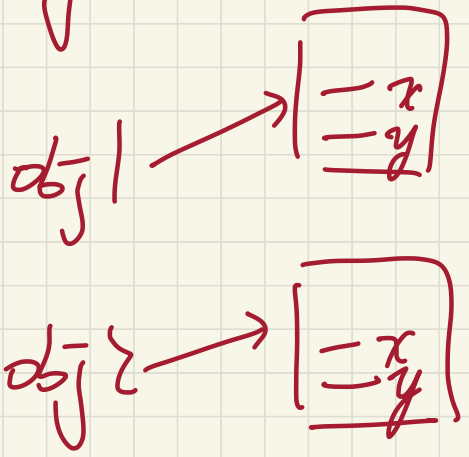
☑ (1) `obj1 == obj2` ⇒ `obj1.equals(obj2)`

☐ (2) `obj1.equals(obj2)` ⇏ `obj1 == obj2`

X

not in general





$\neg\, x.\text{equals}(\text{null})$

↳ x.equals(null) should always evaluate to **false**

∵ x cannot be null (otherwise NPE).

assertSame vs. assertEquals

address.

assertSame(exp1, exp2) → exp1 == exp2

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

exp1 → □ ← exp2

exp1 != exp2 → false

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

pass

fail

assertEquals(exp1, exp2)

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

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

assertFalse(i != j) → fail.

(i = 10, j = 20)

↳ assertTrue(i == j) fail.

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);
```

* Writing p2 == p3 directly causes compilation error.

ref types

assertEquals(exp1 , exp2)

↳ exp1.equals(exp2)

① assertEquals(exp1 , exp2)

↳ exp1.equals(exp2)

invoke the equals method defined in exp1's D.T.

② assertEquals(exp2 , exp1)

↳ exp2.equals(exp1)

invoke the equals method defined in exp2's D.T.