Lecture 11 - Oct. 10

Object Equality

equals Method: Default vs. Overridden
Overriding equals Method: Phases 1 - 3
Static Type, Dynamic Type, Type Casting

Announcements/Reminders

Office Hours during Reading Week TBA:

- Help on Lab2
- Go over WrittenTest1 answers
- Questions on course materials, e.g., OOP reviews

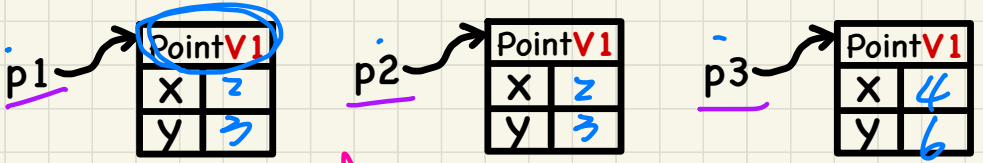
Bonus Opportunity: Midterm Course Survey (eClass)

The equals Method: Default Version

```
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); /* [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] */
```

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



The equals Method: Overridden Version

Phase 1

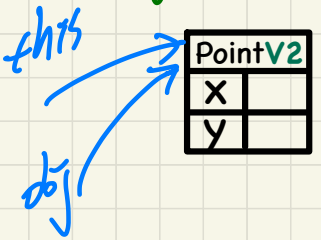
```
public class Object {
   ...
   public boolean equals(Object obj) {
      return this == obj;
   }
}
```

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

overridden/redefined equals.

if C.O. "this" is the same object as the input param "obj" → they're equal.



The equals Method: Overridden Version

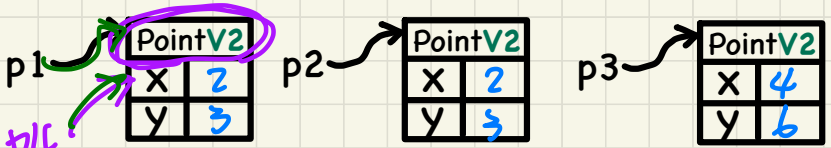
Example 1: Trace L7

```
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); /* [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] */
```

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



Context object pointing to a PointV2 object → overridden version of equals in PointV2 indeed

method overloading

vs.

method overridding

(re-definition)

The equals Method: Overridden Version

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

reaching this phase means !(this == obj)

* logically unnecessary
if this == null (T)
→ NPE would have occurred.

p1.equals(x); → null → false

** Q. What if this == null is also true?

① if(obj == null){
② if(this == null) return T;
③ else return F;
}

The equals Method: Overridden Version

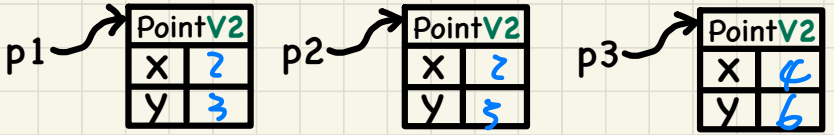
Example 1: Trace L8

```
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 }
    Point 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] */
```



Static Type, Dynamic Type, Type Casting

Static Type: a ref variable's **declared** type

Dynamic Type: type of address **currently** stored in a ref variable

Type **Casting**: creating an expression of certain **static** type

determines the method that's applicable to the C.O.

```
class C1 {
  ...
  public void m1() {...}
}

class C2 {
  ...
  public void m2() {...}
}
```

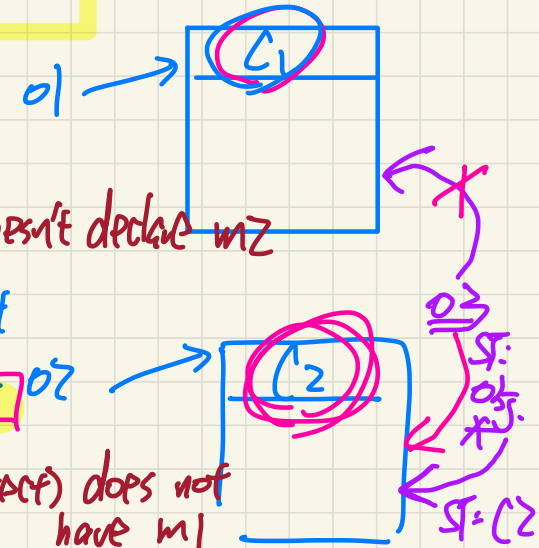
```
① C1 o1 = new C1();      S.T. D.T.
② C2 o2 = new C2();
o1.m1(); ✓
o1.m2(); X ∵ ST of o1 doesn't declare m2
o2.m1(); X
o2.m2(); ✓
Object o3;                ST: Object
o3 = o1;                  ST of o3: Object, DT of o3: C1
o3.m1; X ∵ ST of o3 (Object) does not have m1
o3.m2; X
((C1) o3).m1(); ✓
((C1) o3).m2(); X
o3 = o2;                  ST of o3: Object, DT of o3: C2
* ((C2) o3).m1(); X
((C2) o3).m2(); ✓
```

Object class does not declare m1 and m2.

exp of ST: C1

Q. ((C2) o3).m2(); don't worry for now!

To cast an obj, always cast it to its D.T. D.T.



The equals Method: Overridden Version

Phase 3

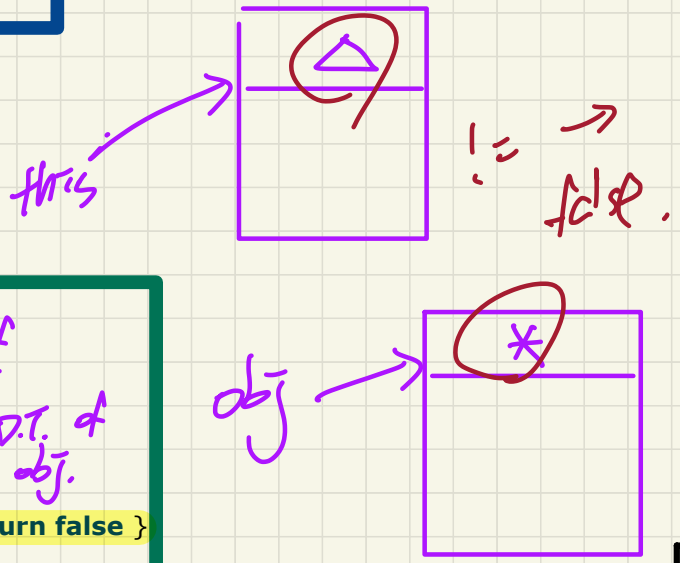
```
public class Object {
   ...
   public boolean equals(Object obj) {
      return this == obj;
   }
}
```

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

D.T. of this

D.T. of obj.



PointV2	
x	
y	

The equals Method: Overridden Version

Example 1: Trace L9

```
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; }
    Point 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] */
```