Lecture 11 - Oct. 10

Object Equality

equals Method: Default vs. Overridden
Overriding equals Method: Phases 1 - 2
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)

```
class PointV1 {
    int x;
    int y;
    PointV1(int x, int y) { ... }
}
```

```
PointV1 p1 = new ...
PointV1 p2 = new ...
println( p1.equals(p2) )
```

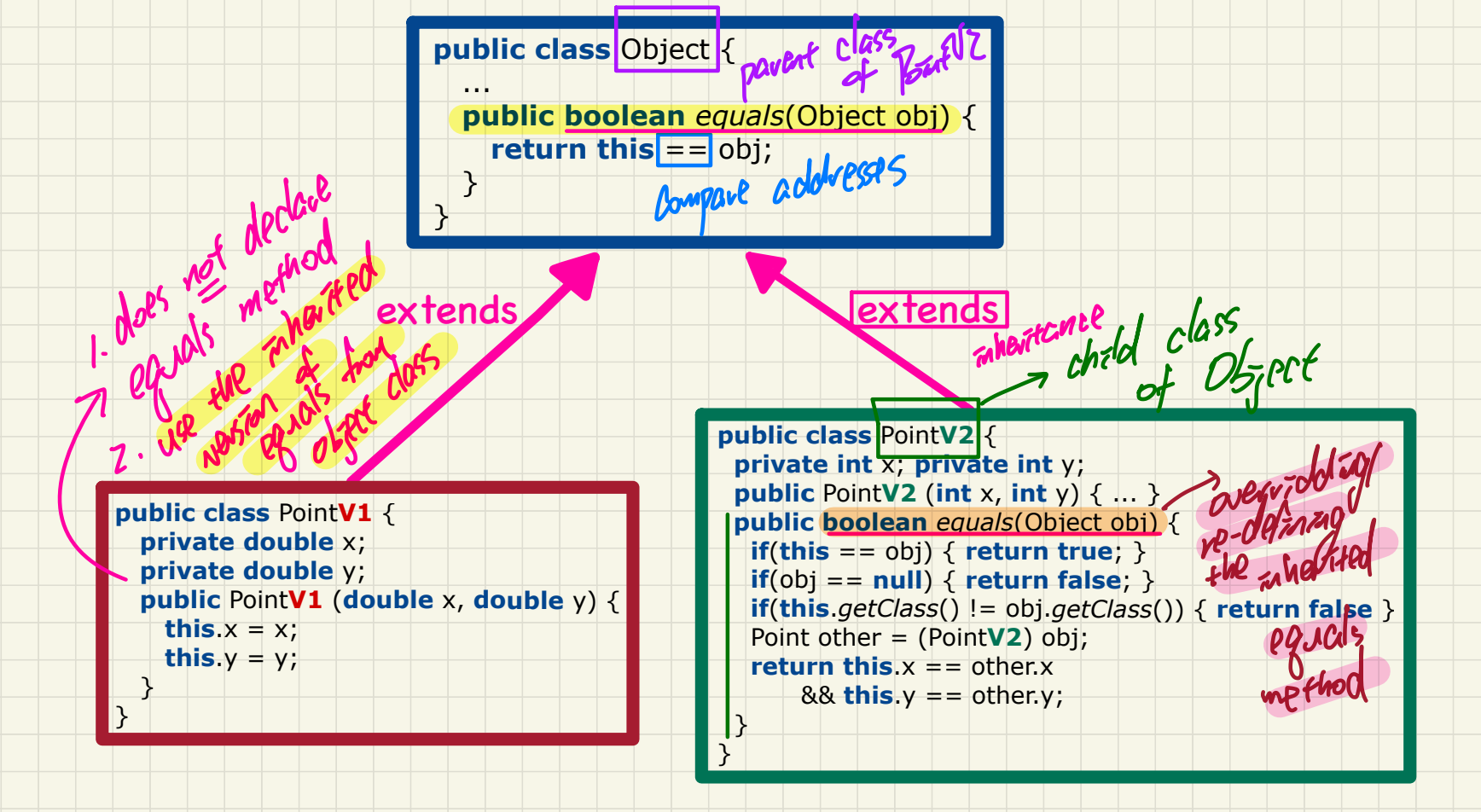
Q1. Does this compile, given that equals is not declared in PointV1?

YES.

Q2. Where does equals come from?

Object.

The equals Method: To Override or Not?

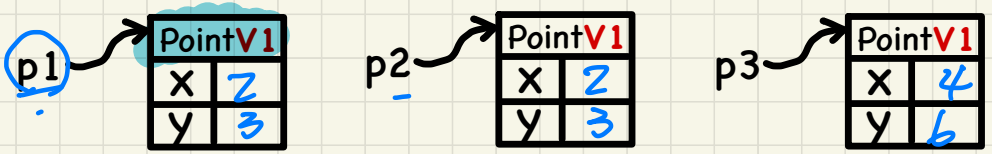


The equals Method: Default Version

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

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



* p1.equals(p1) C.O.

** p1.equals(s)
↳ equals from Object : "p1 == s" F

Note. writing p1 == s → compilation errors

The equals Method: Overridden Version

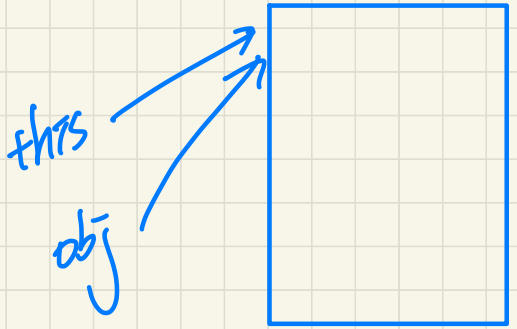
Phase 1

```
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;
  }
}
```

p1.equals(..);



①

PointV2	
x	
y	

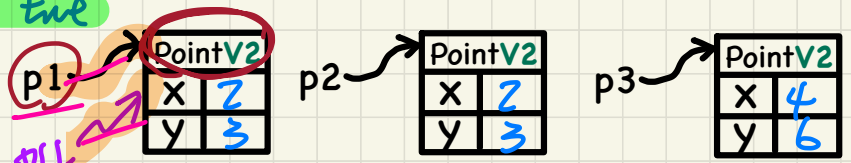
The equals Method: Overridden Version

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] */
```

```
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;
  }
}
```



The equals Method: Overridden Version

Phase 2

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

extends

sensible?

alt

```
if(obj == null){
   if(this == null){
      return true; }
   else return false;
}
```

```
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;
  }
}
```

PointV2 p5 = null;
p5.getX();
NPE

p1.equals(..)

this is null
NPE

reaching this means this != obj

this != null

this → null

obj → null

this →

obj →

The equals Method: Overridden Version

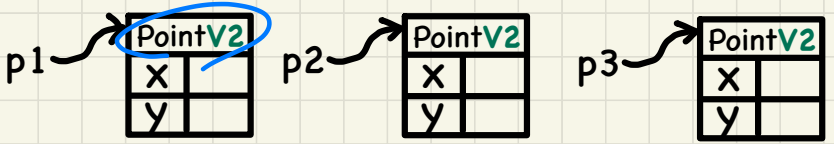
Example 1: Trace L8

```
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] */
```



obj → null

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

For now, always cast to the same D.T.

ST: of a C.O. is of certain ST, it determine the range of methods applicable that on C.O.

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

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

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

ST: C1 / ST: C2

o1 ST: C1 DT: C1 — o2 ST: C2 DT: C2

o1 ST: C1 does not contain m2.

o3 ST: Object — o3 ST: o3: object — DT o3: C1

exp of static type C1

* ST of o3: object — DT of o3: C2

Object class has no declaration of m1 and m2.