Lecture 14 - Oct 27

Object Equality

Overridden equals: Phases 1, 2, 3
Static vs. Dynamic Types, Type Casting

Announcements/Reminders

- Today's class: notes template posted
- ProgTest0 and ProgTest1 marks and feedback released
- WrittenTest1 this Wednesday (October 29)
  + Some practice questions on eClass
  + Past review session materials released
  + Lecture 12 and Lecture 13 this week are covered.
- Lab3 released

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

anything is equal to itself.

p1.equals(p2)

c.o. arg

(arg.) p2

obj (param.)

(c.o.) p1

this

PointV2	
x	
y	

The equals Method: Overridden Version

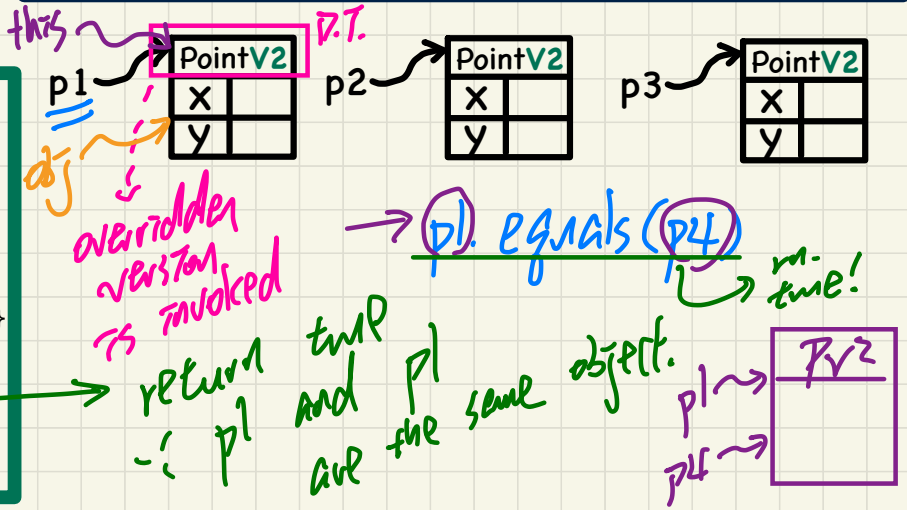
Example 1: Trace L7

```
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(p1 == 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: Overridden Version

Phase 2

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

extends

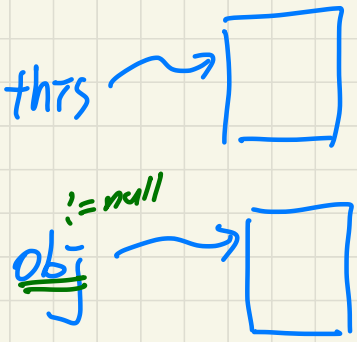
```
public class PointV2 {
  private int x;
  private int y;
  public PointV2 (int x, int y) { ... }
  public boolean equals(Object obj) {
    if(this == obj) { return true; }
    if(obj == null) { return false; }
    if(this.getClass() != obj.getClass()) { return false }
    PointV2 other = (PointV2)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

reaching this line: this != obj.

p1.equals(p2) p2 != null

↳ if p1 == null

↳ NPE even before we step into the body of equals method



not reachable, ∴ NPE would've occurred!!

* alt

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

PointV2	
x	
y	

The equals Method: Overridden Version

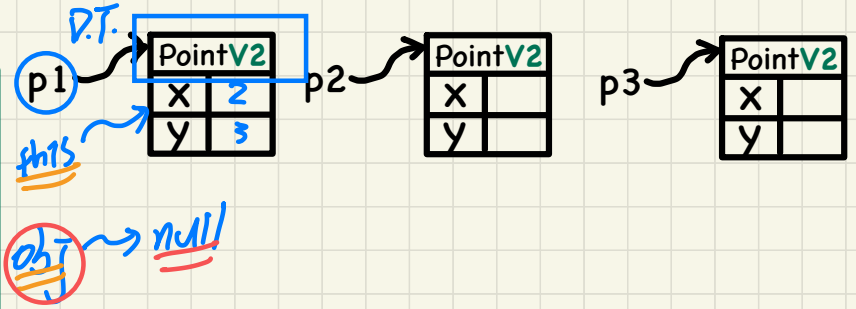
Example 1: Trace L8

```
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 }
    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(p2.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

may be changed

runtime type: determines the version of method called.

fixed

declared type

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

aliases

var.	ST	exp
o1	C1	m1, eg
o2	C2	m2, eg
o3	Object	equals.

determines the range of methods that can be involved on a ref. variable

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

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

(C1) o3 — cast type

```
C1 o1 = new C1();
C2 o2 = new C2();
o1.m1();
o1.m2();   X : m2 not part of o1's ST's exp.
o2.m1();   X
o2.m2();
Object o3;
o3 = o1;
o3.m1;     do not compile : ST of o3 remains Object
o3.m2;
((C1) o3).m1();
((C1) o3).m2();   X
o3 = o2;
((C2) o3).m1();   X : cast type is C1.
((C2) o3).m2();
```

D.T.

S.T. C1

S.T. C2

create a new copy of address with a different expectation

C1 o1 → C1

C2 o2 → C2

o3 → Object

(C1) o3

(C2) o3

a new alias that:
1. points to the same object as o3 does **
2. this new alias has ST. same as cast type

Given a line of method call:

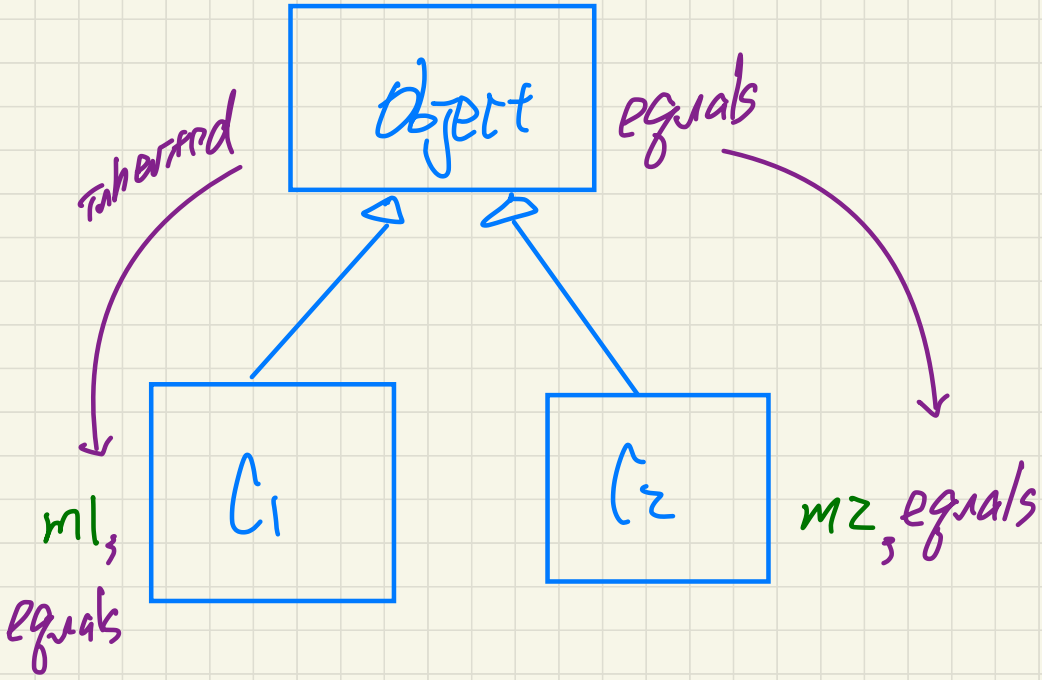
obj.m(...)

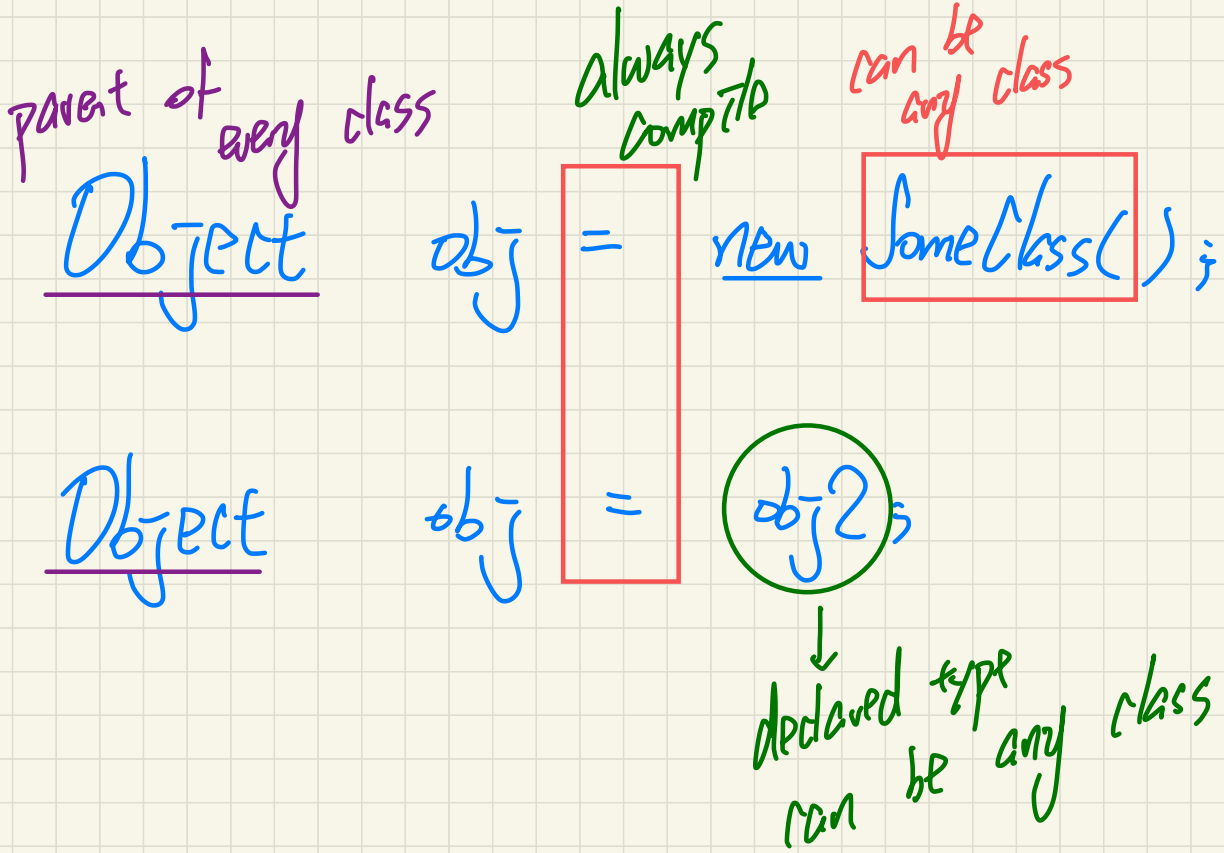
(1) Does the line compile?

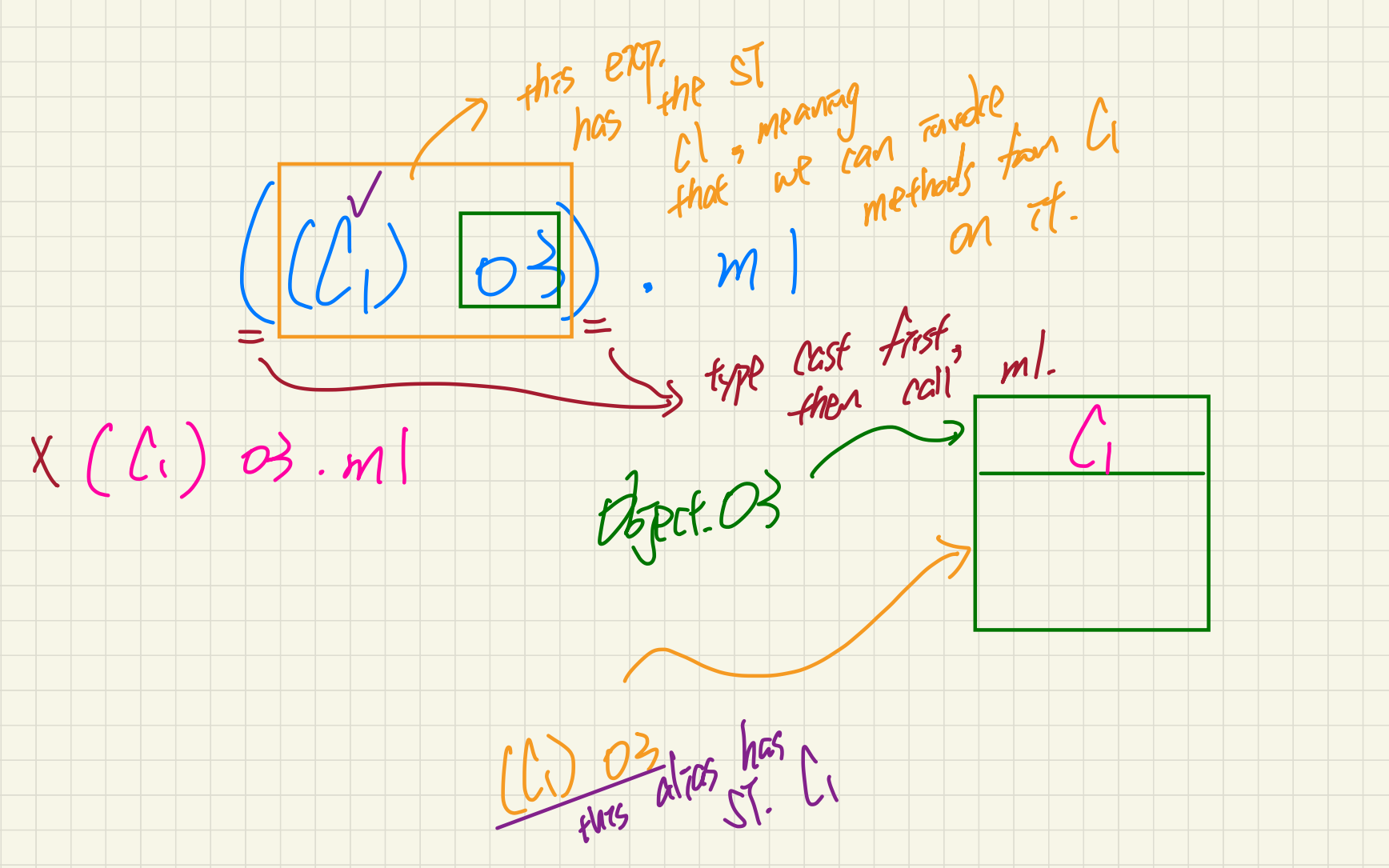
↳ look at S.T. of C.O. obj.

(2) If it compiles, which version of m is called?

↳ look at R.T. of C.O. obj


Object
inherited
equals
C1
C2
m1, equals
m2, equals


parent of every class
Object
Object
always compile
can be any class
obj = new SomeClass();
obj = obj2;
declared type
can be any class


this exp. has the ST C1, meaning that we can invoke methods from C1 on it.
((C1) o3).m1
type cast first, then call m1.
X ((C1) o3).m1
Object.o3
C1
((C1) o3) alias has ST. C1

The equals Method: Overridden Version

Phase 3

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

extends

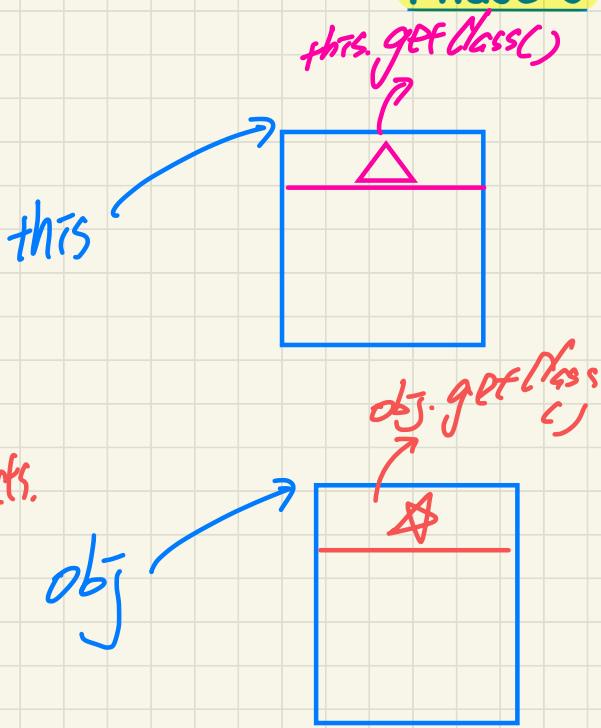
```
public class PointV2 {
  private int x;
  private int y;
  public PointV2 (int x, int y) { ... }
  public boolean equals(Object obj) {
    if(this == obj) { return true; }
    if(obj == null) { return false; }
    if(this.getClass() != obj.getClass()) { return false }
    PointV2 other = (PointV2)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

return the D.T. of the context objects.

this != obj

obj != null

objects with different D.T.s are trivially not equal to each other.



The equals Method: Overridden Version

Example 1: Trace L9

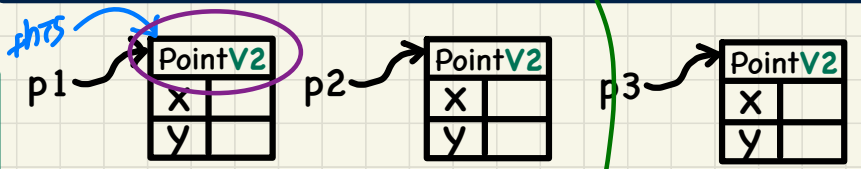
```
public class Object {
   ...
   public boolean equals(Object obj) {
      return this == obj;
   }
}
```

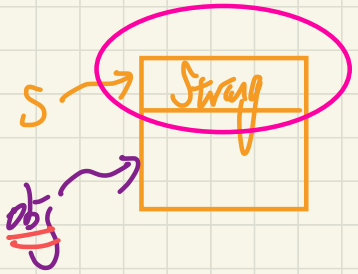
extends

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

PointV2 != String ①

```
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] */
```





rn. false
∵ p1 and s have different R.T.s.

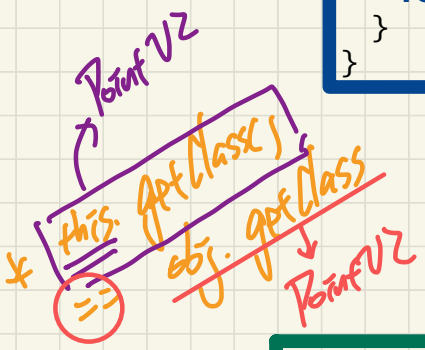
The equals Method: Overridden Version

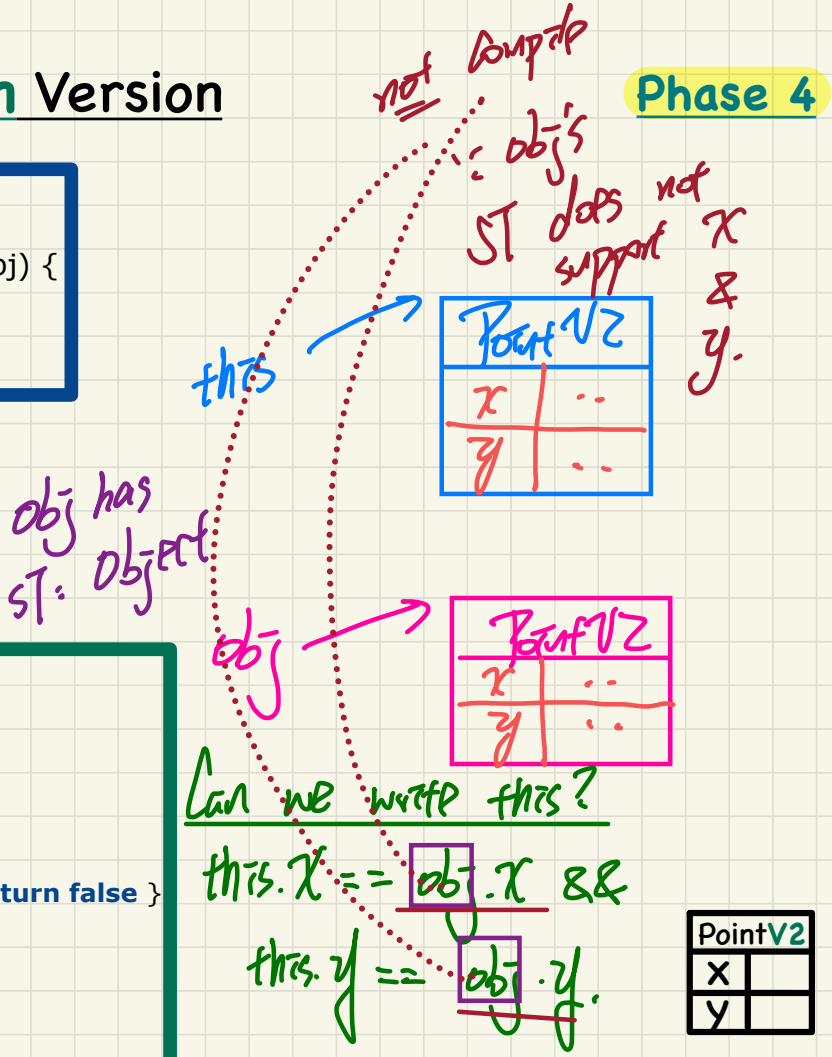
Phase 4

```
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;
  }
}
```





obj has ST: Object

1. this != obj
2. obj != null
3. *

PointV2	
x	
y	