Lecture 12 - June 12

Object Equality

Use of ==: Primitive vs. Reference
equals*: Default Version from Object*
equals*: Overridden Version Phases 1, 2*

Announcements/Reminders

- Today's class: notes template posted
- ProgTest0 marks & feedback released
- ProgTest1 tomorrow JUN 13 (during enrolled lab session)
- Priorities:
  - + Lab1 solution, Lab2
  - + Slides on **Classes and Objects**
  - + Slides on **Exceptions**

Equality ==

Primitive

1. ==
2. equals X

i.equals(j) X

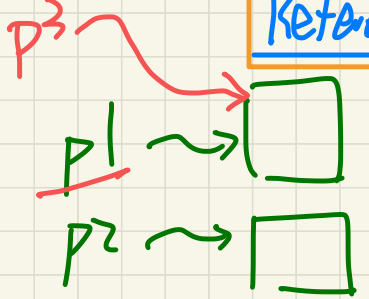
```
int i = 23;
int j = 46;
```

i == j

false

Reference

1. ==
2. equals.




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

p1 == p2 → false

↳ values of different addresses

p3 == p1 → true

Object Equality: First Example

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

```
class PointTester {
  … main(…) {
    PointV1 p1 = new PointV1(3, 4);
    PointV1 p2 = new PointV1(3, 4);
    System.out.println(p1.equals(p2));
  }
}
```

Q1:

Does PointTester compile, given that **equals** is **not** declared in PointV1?

YES.

Q2:

Where does **equals** come from?

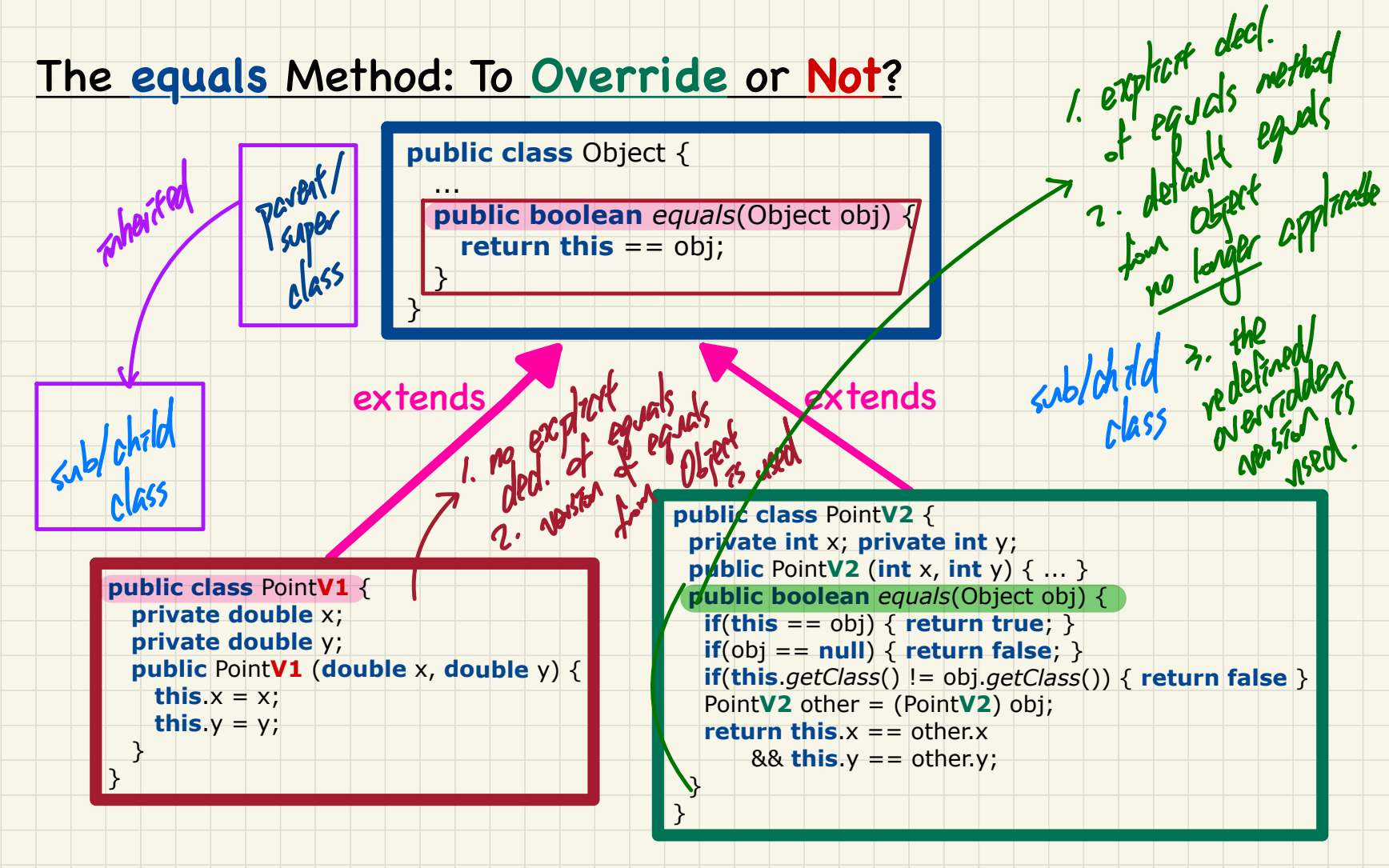
Object

step into: PointV1 (Object). equals

p1 == p2

def. for equals method in Object class

The equals Method: To Override or Not?



The equals Method: Default Version

* 1. C.O. p1 has type PointV1
2. PointV2 has no overridden eq.
3. Version of equals from Object is invoked.

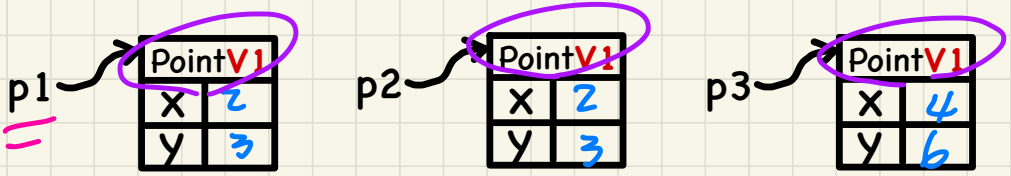
```
public class Object {
   ...
   public boolean equals(Object obj) {
      return this == obj;
   }
}
```

extends

```
public class PointV1 {
   private int x;
   private int y;
   public PointV1 (int x, int y) {
      this.x = x;
      this.y = 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); F /* [redacted] */
6  System.out.println(p2 == p3); F /* [redacted] */
7  System.out.println(p1.equals(p1)); T /* [redacted] */
8  System.out.println(p1.equals(null)); F /* [redacted] */
9  System.out.println(p1.equals(s)); F /* [redacted] */
10 System.out.println(p1.equals(p2)); /* [redacted] */
11 System.out.println(p2.equals(p3)); /* [redacted] */
```

*



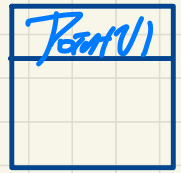
s ~> "(2, 3)"

7. -> p1 == p1 true
8. -> p1 == null
9. -> "p1 == s" F

PointV1 p1 = new ...;

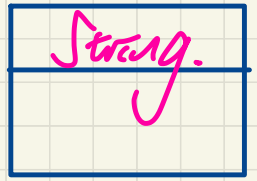
String s = "...";

① p1 == s → compilation error :(p1 and s have different types.



② p1.equals(s) → Object version

↳ "p1 == s"



Call by Value: Invoking the Overridden equals

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

extends

default

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

call by value: obj = p3

```
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;
  }
}
```

param

p2 → PointV2 { x: 3, y: 4 }

arg.

p3 → PointV2 { x: 4, y: 5 }

obj

overridden version.

overridden version from PointV2 class

The equals Method: Overridden Version

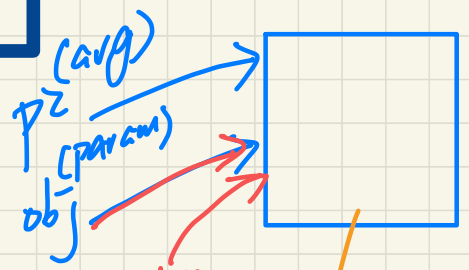
Phase 1

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

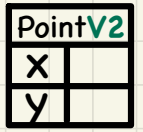
extends

```
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;
  }
}
```

p1.equals(p2)



an object should be equal to itself.



The equals Method: Overridden Version

Example 1: Trace L7

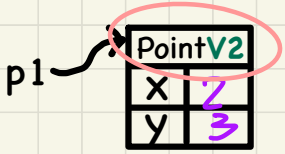
```
public class Object {
   ...
   public boolean equals(Object obj) {
      return this == obj;
   }
}
```

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(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] */
```

true.



an object of type PointV2

↳ overridden version of equals is called.

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 point means: this != obj

p1.equals(p2)

alt.

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

impossible to reach.

this != null

∴ if this is null, the equals method call would've already thrown NPE.

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(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] */
```