Lecture 12 - Oct 20

Object Equality

Call by Value: Prim. vs. Ref. Types
Equality: == vs. equals
Default Version of equals in Object Class

Announcements/Reminders

- Today's class: notes template posted
- ProgTest0, ProgTest1 grading process:
  + TAs are due to return results to me Tuesday afternoon.
  + I will take one day or two to finalize the results.
- WrittenTest1 next Wednesday (October 29)
  + Some practice questions on eClass
  + Lecture 12 and Lecture 13 this week **are** covered.
  + Zoom Review Session: 4 PM on Saturday, October 25
- Lab3 to be released (and deadline shifted accordingly)

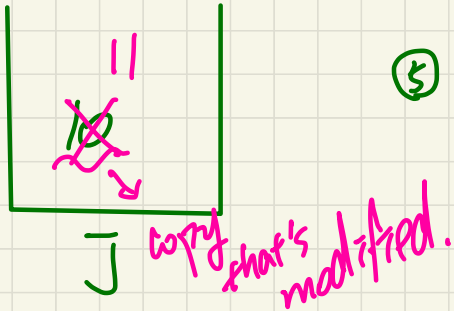
Call by Value: Re-Assigning Primitive Parameter

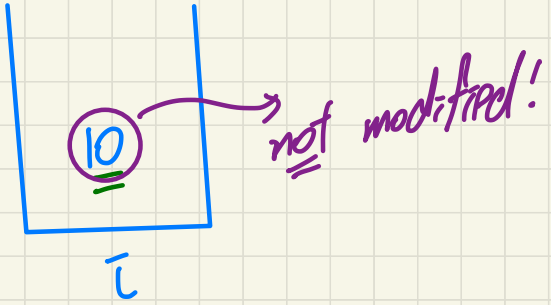
```
public class Util {
  void reassignInt(int j) {
  6 j = j + 1; }
  void reassignRef(Point q) {
    Point np = new Point(6, 8);
    q = np; }
  void changeViaRef(Point q) {
    q.moveHorizontally(3);
    q.moveVertically(4); } }
```

```
1  @Test
2  public void testCallByVal() {
3  1 Util u = new Util();
4  2 int i = 10;
5  3 assertTrue(i == 10);
6  4 u.reassignInt(i);
7  7 assertTrue(i == 10);
8  }
```

Call by value:

5 j = i





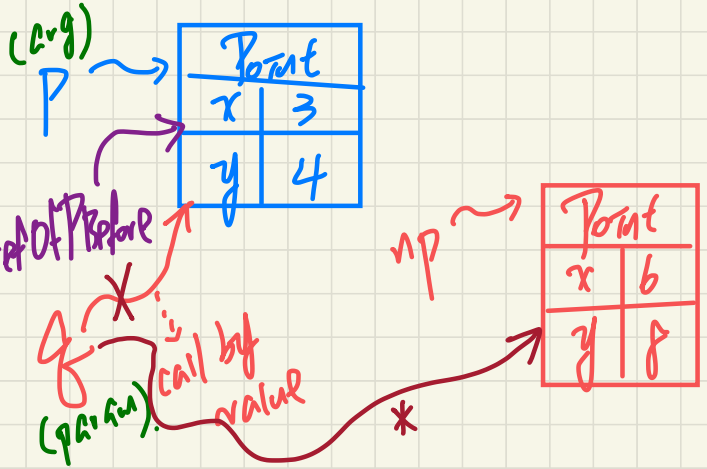
* only q, a copy of p, is modified. p itself is not modified.

Call by Value: Re-Assigning Reference Parameter

```
public class Util {
  void reassignInt(int j) {
    j = j + 1; }
  void reassignRef(Point q) {
    Point np = new Point(6, 8);
    q = np; }
  void changeViaRef(Point q) {
    q.moveHorizontally(3);
    q.moveVertically(4); } }
```

```
1  @Test
2  public void testCallByRef_1() {
3    Util u = new Util();
4    Point p = new Point(3, 4);
5    Point refOfPBefore = p;
6    u.reassignRef(p);
7    assertTrue(p == refOfPBefore);
8    assertTrue(p.getX() == 3);
9    assertTrue(p.getY() == 4);
10 }
```

Incorrect to expect that p is re-assigned to np.

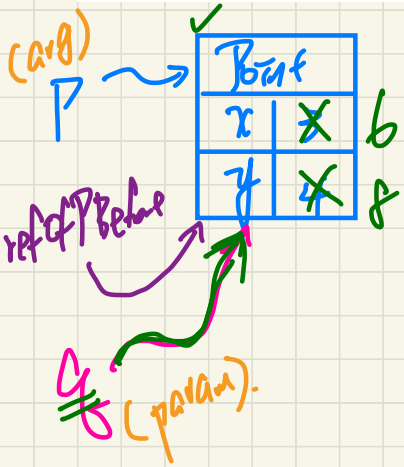



```
public class Point {
  private int x;
  private int y;
  public Point(int x, int y) {
    this.x = x;
    this.y = y;
  }
  public int getX() { return this.x; }
  public int getY() { return this.y; }
  public void moveVertically(int y){ this.y += y; }
  public void moveHorizontally(int x){ this.x += x; }
}
```

Call by Value: Calling Mutator on Reference Parameter

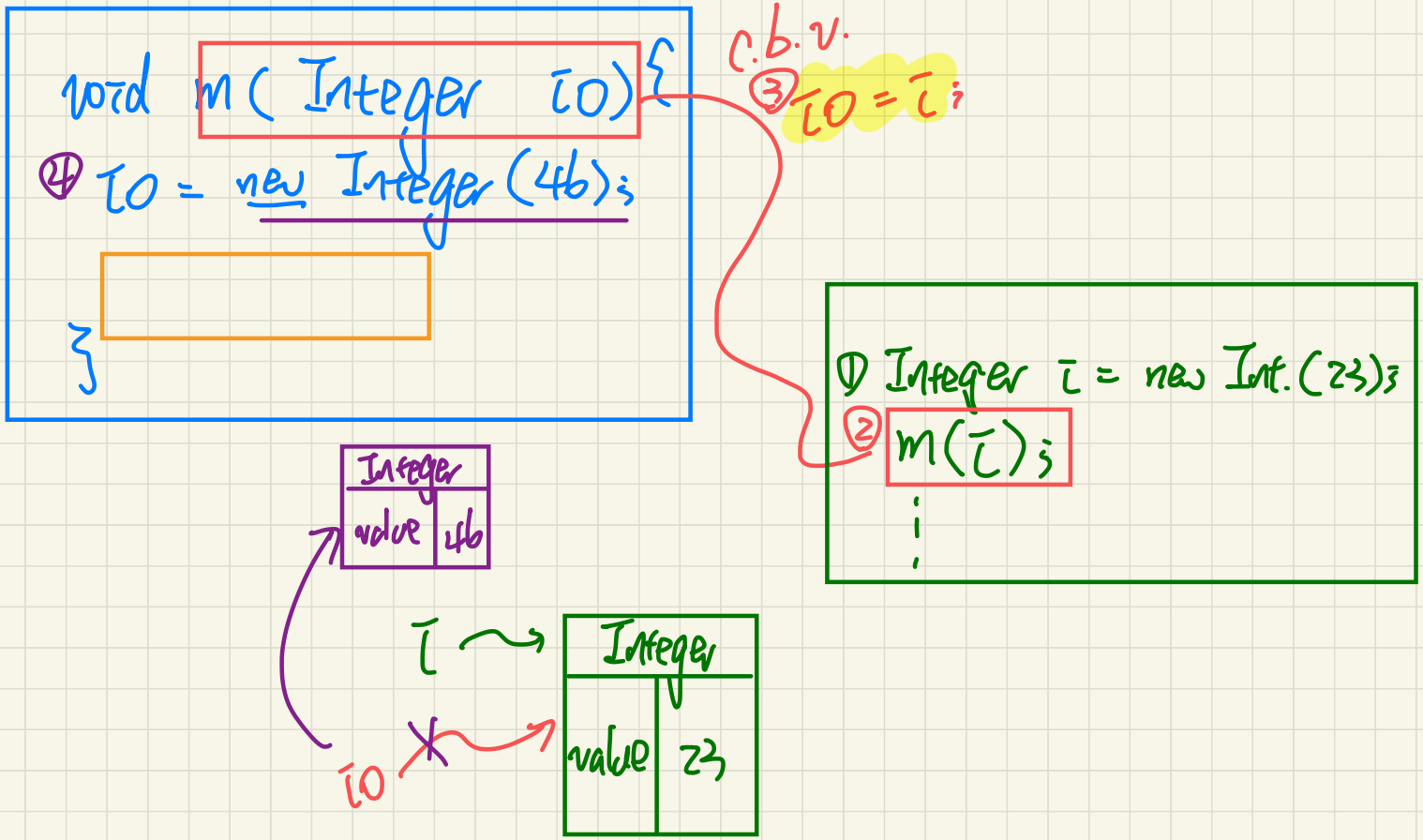
```
public class Util {
  void reassignInt(int j) {
    j = j + 1; }
  void reassignRef(Point q) {
    Point np = new Point(6, 8);
    q = np; }
  void changeViaRef(Point q) {
    q.moveHorizontally(3);
    q.moveVertically(4); } }
```

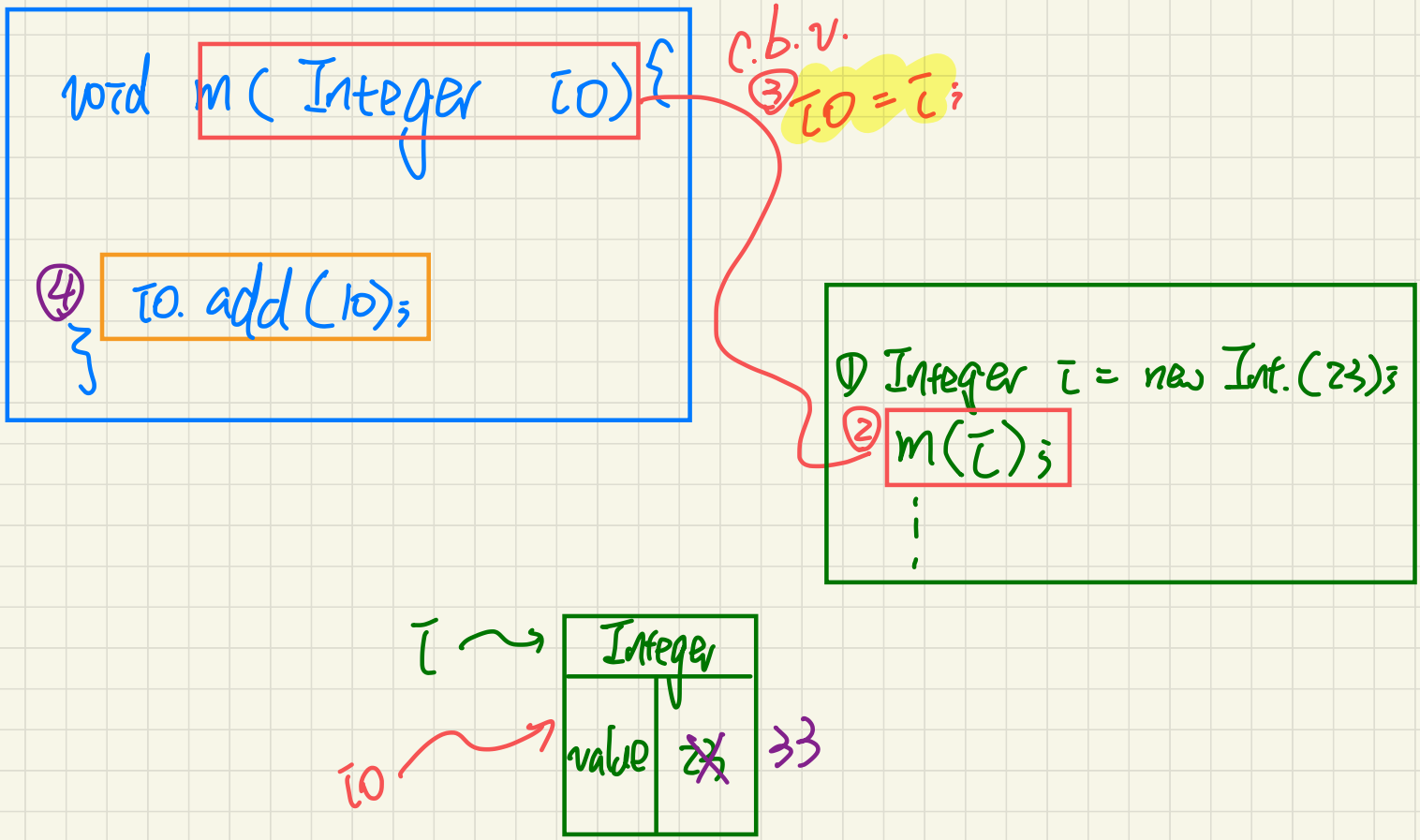
```
1  @Test
2  public void testCallByRef_2() {
3    Util u = new Util();
4    Point p = new Point(3, 4);
5    Point refOfPBefore = p;
6    u.changeViaRef(p);
7    assertTrue(p == refOfPBefore);
8    assertTrue(p.getX() == 6);
9    assertTrue(p.getY() == 8);
10 }
```



C.b.V.
⑤ q = p

```
public class Point {
  private int x;
  private int y;
  public Point(int x, int y) {
    this.x = x;
    this.y = y;
  }
  public int getX() { return this.x; }
  public int getY() { return this.y; }
  public void moveVertically(int y){ this.y += y; }
  public void moveHorizontally(int x){ this.x += x; }
}
```


void m(Integer io){
④ io = new Integer(46);
}
c.b.v.
③ io = i;
① Integer i = new Int.(23);
② m(i);
Integer
value 46
i
io
Integer
value 23


void m(Integer io){
④
io.add(10);
}
c.b.v.
③ io = i;
① Integer i = new Int.(23);
② m(i);
i
Integer
io
value
23
33

Equality < == < equals

Primitive

1. == (value).
2. equals ✗ ↳ compilation error

```
int i = 23;
int j = 45;
i == j
```

Reference

1. == false
2. equals
   1. either a version from Object class
   2. or a version that is defined in Point class

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

i.equals(j) ✗ compilation error.

p1 == p2 false ↳ different add. values.

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);
①   System.out.println(p1.equals(p2));
  }
}②println(p1==p2);
```

effectively the same ⇒ no explicit def. of equals method in PointV1 class.

p1 == p2
this obj.

Q1:

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

Yes ∵ equals method is implicitly defined in every class.

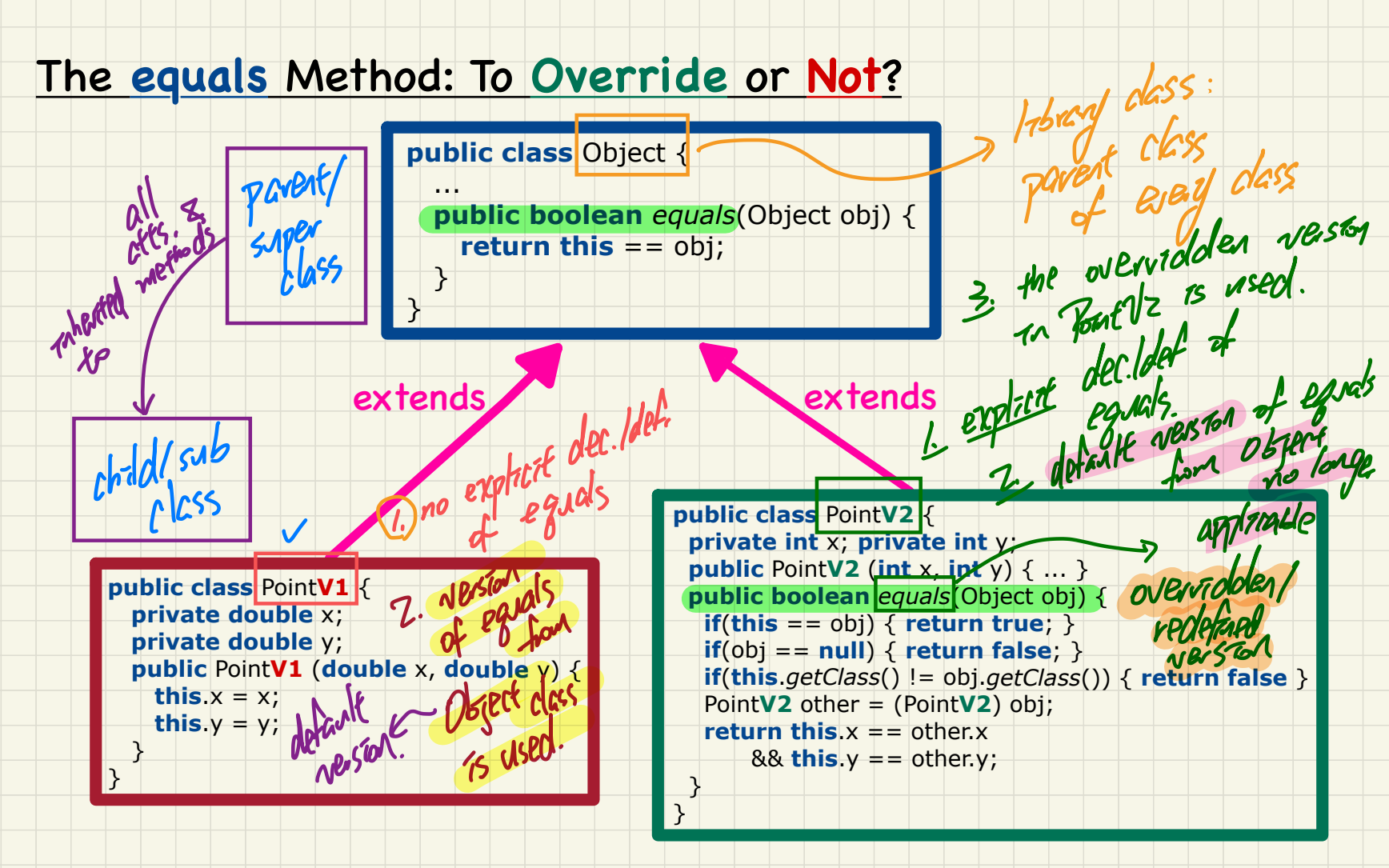
Q2:

Where does **equals** come from?

Eclipse: you may have to attach the source of your JDK installation

step into: Object class' equals method being called

The equals Method: To Override or Not?



The equals Method: Default Version

* inherited from Object class *

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

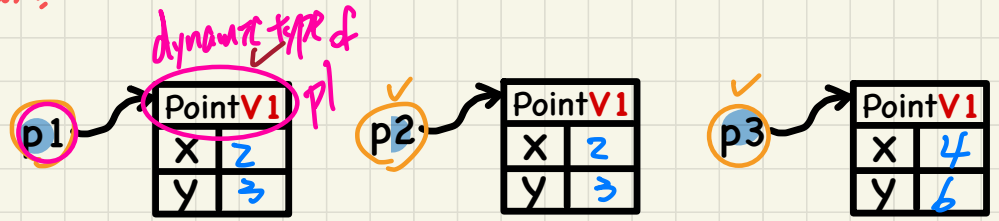
inherited to PointV1 class.

extends

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

no explicit redefinition/overridding of equals.

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



L7: C.O. p1 points to an object of type PointV1

dynamic type of p1 is PointV1

↳ the equals method comes from the one *