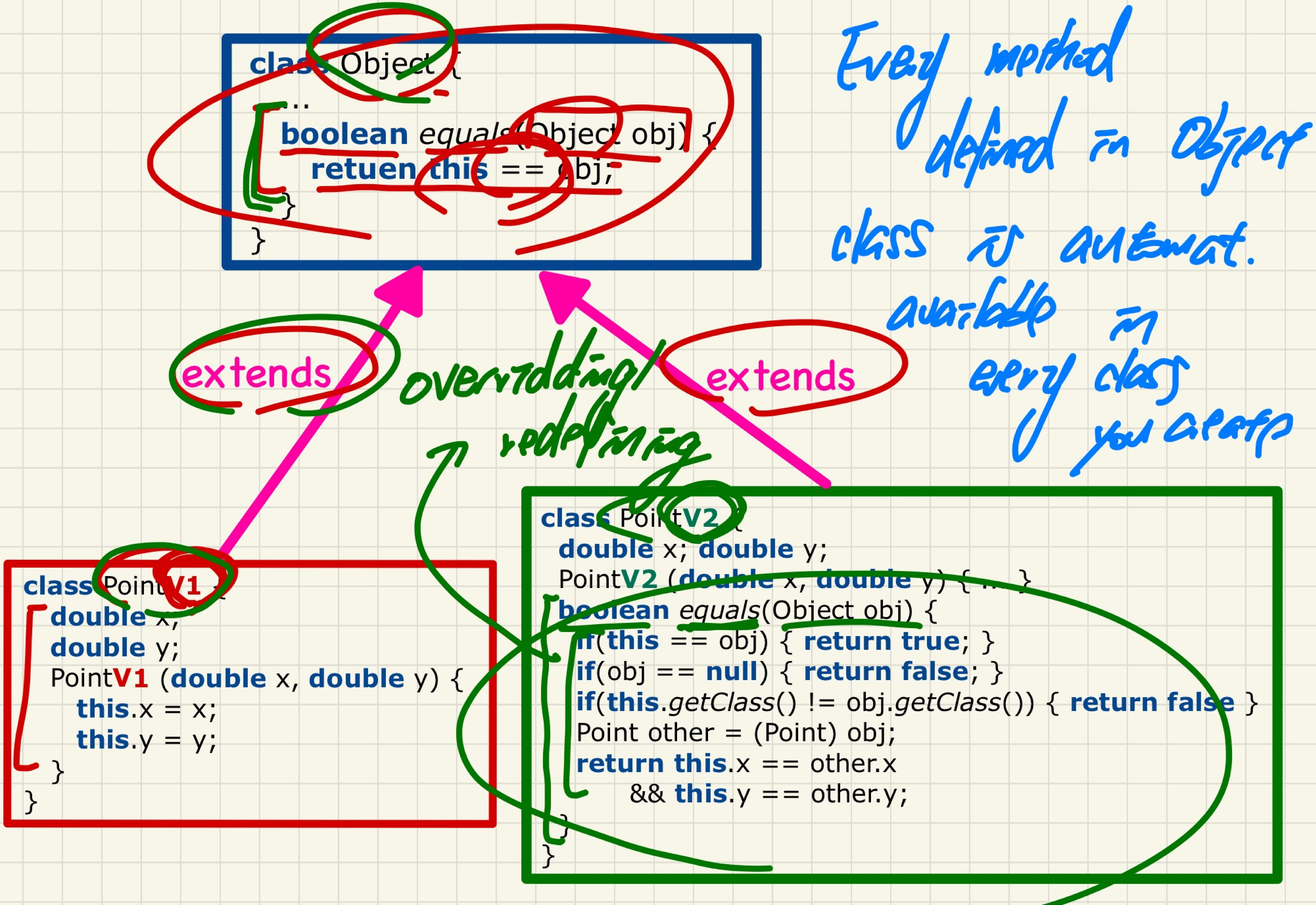
LECTURE 8

MONDAY SEPTEMBER 30

The equals Method: To Override or Not?



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

```
class PointV1 {
  double x;
  double y;
  PointV1 (double x, double y) {
    this.x = x;
    this.y = y;
  }
}
```

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

Every method defined in Object class is automat. available in every class you create

The equals Method: Default Version

[redacted]

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

null

p1

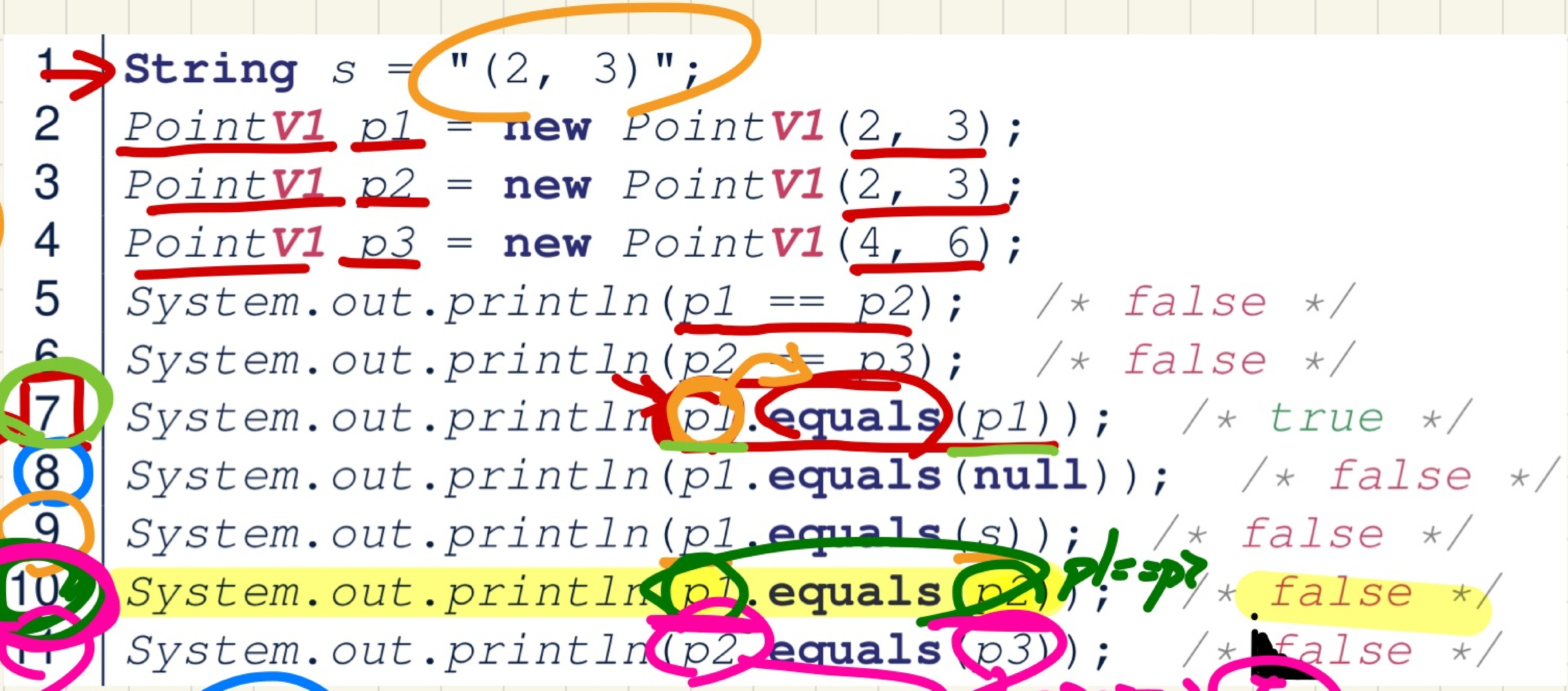
p1

p1

p1 == null

extends

```
class PointV1 {
  double x;
  double y;
  PointV1 (double x, double y) {
    this.x = x;
    this.y = y;
  }
}
```

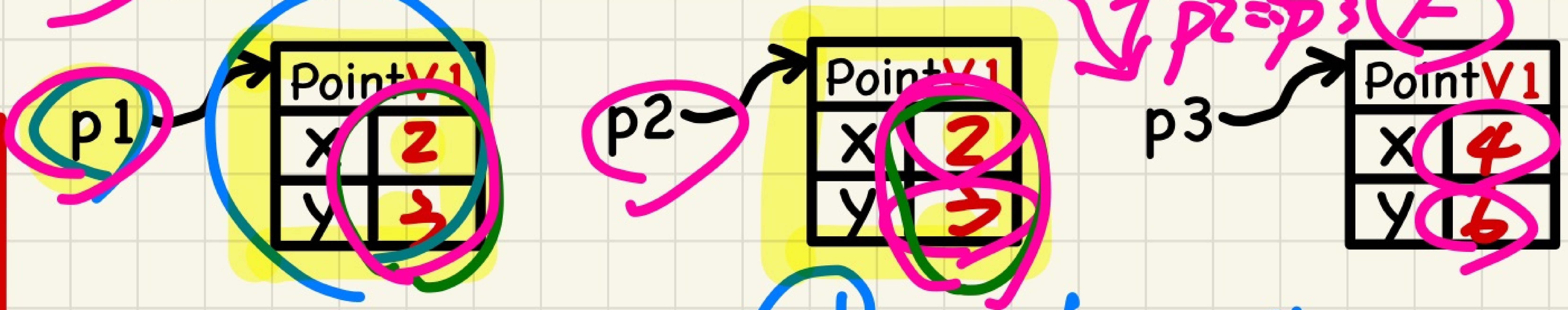


```
String s = "(2, 3)";
PointV1 p1 = new PointV1(2, 3);
PointV1 p2 = new PointV1(2, 3);
PointV1 p3 = new PointV1(4, 6);
System.out.println(p1 == p2);   /* false */
System.out.println(p2 == p3);   /* false */
System.out.println(p1.equals(p1));   /* true */
System.out.println(p1.equals(null));   /* false */
System.out.println(p1.equals(s));   /* false */
System.out.println(p1.equals(p2));   /* false */
System.out.println(p2.equals(p3));   /* false */
```

p1!=p2

p2==p3

F



p1.equals(null) F

↳

C.O.

p1.equals(p1)

p1.equals(p2) F

p1.equals(s)

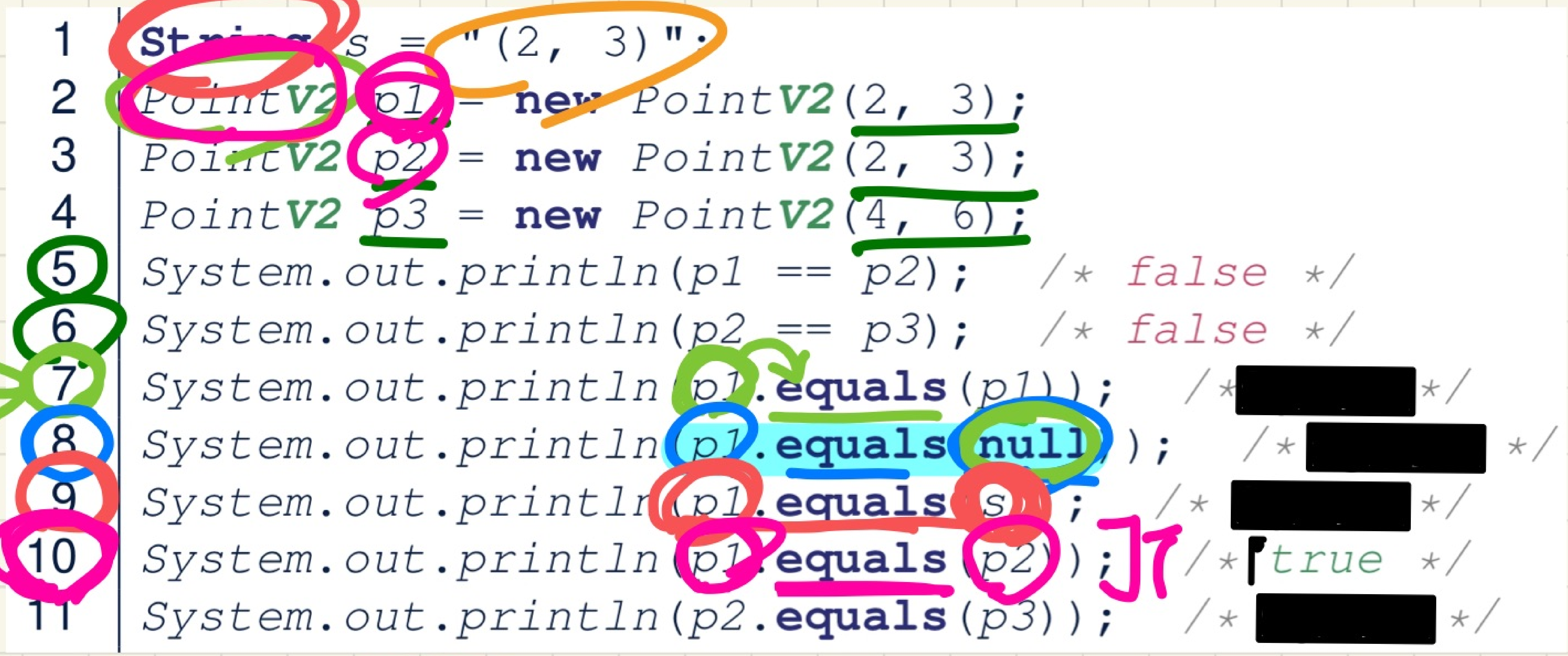
↳ p1 == s F

The equals Method: Overridden Version Example 1

[redacted]

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

extends



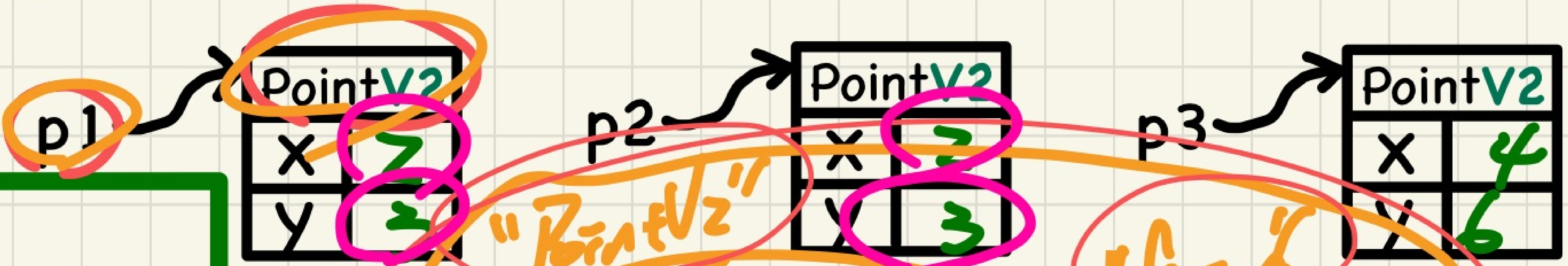
```
1  String s = "(2, 3)";
2  PointV2 p1 = new PointV2(2, 3);
3  PointV2 p2 = new PointV2(2, 3);
4  PointV2 p3 = new PointV2(4, 6);
5  System.out.println(p1 == p2);   /* false */
6  System.out.println(p2 == p3);   /* false */
7  System.out.println(p1.equals(p1));   /* [redacted] */
8  System.out.println(p1.equals(null));   /* [redacted] */
9  System.out.println(p1.equals(s));   /* [redacted] */
10 System.out.println(p1.equals(p2));  /* true */
11 System.out.println(p2.equals(p3));  /* [redacted] */
```

L7: p1 == p1

L8: null == null T

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

p1.x == p2.x && p1.y == p2.y



"PointV2" "String"

L9: p1.getClass() != s.getClass() T

dynamic type

p1.equals(p1)

↳ type of c.o. p1: PointV2

∴ equals redefined ∴ redefined version called

PointV1 p1 = new . _ _

PointV2 p2 = new . _

p1.equals(p2)

c.o. "PointV1" "PointV2"

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

F

F

F

~~Point~~ PointV2

*

this != obj

obj != null

this.getClass() == obj.getClass()

all true if you can reach *

```
class PointV2 {
  double x; double y;
  PointV2 (double x, double y) { ... }
  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;
  }
}
```

Object

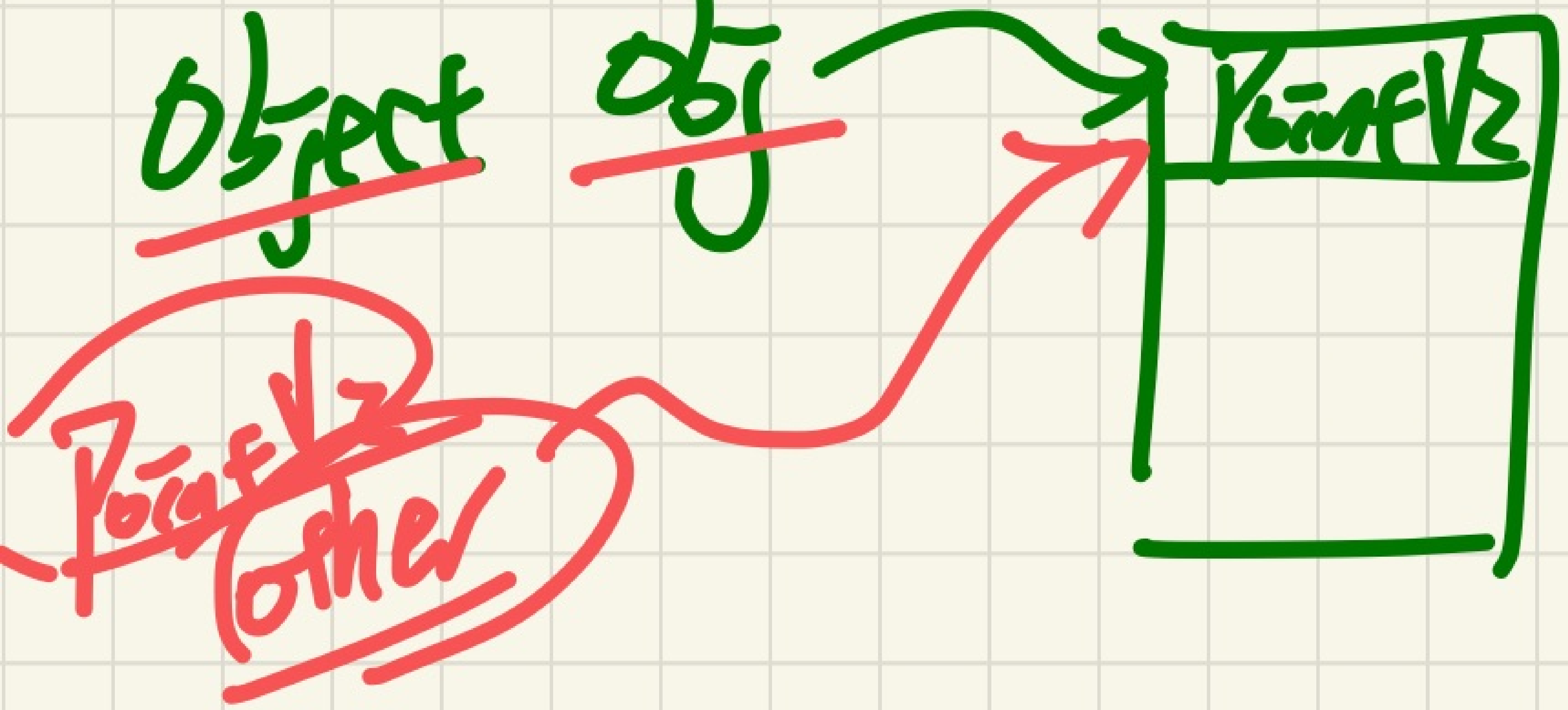
obj

obj has the declare type Object.

this and obj are same type

V1

V2 [return this.x == obj.x && this.y == obj.y



Java compiler only allows attributes/methods defined in the declared type of obj.

Object

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

V2 V2

p1 → null

PointV2 p1 = new - -

p1 = null;

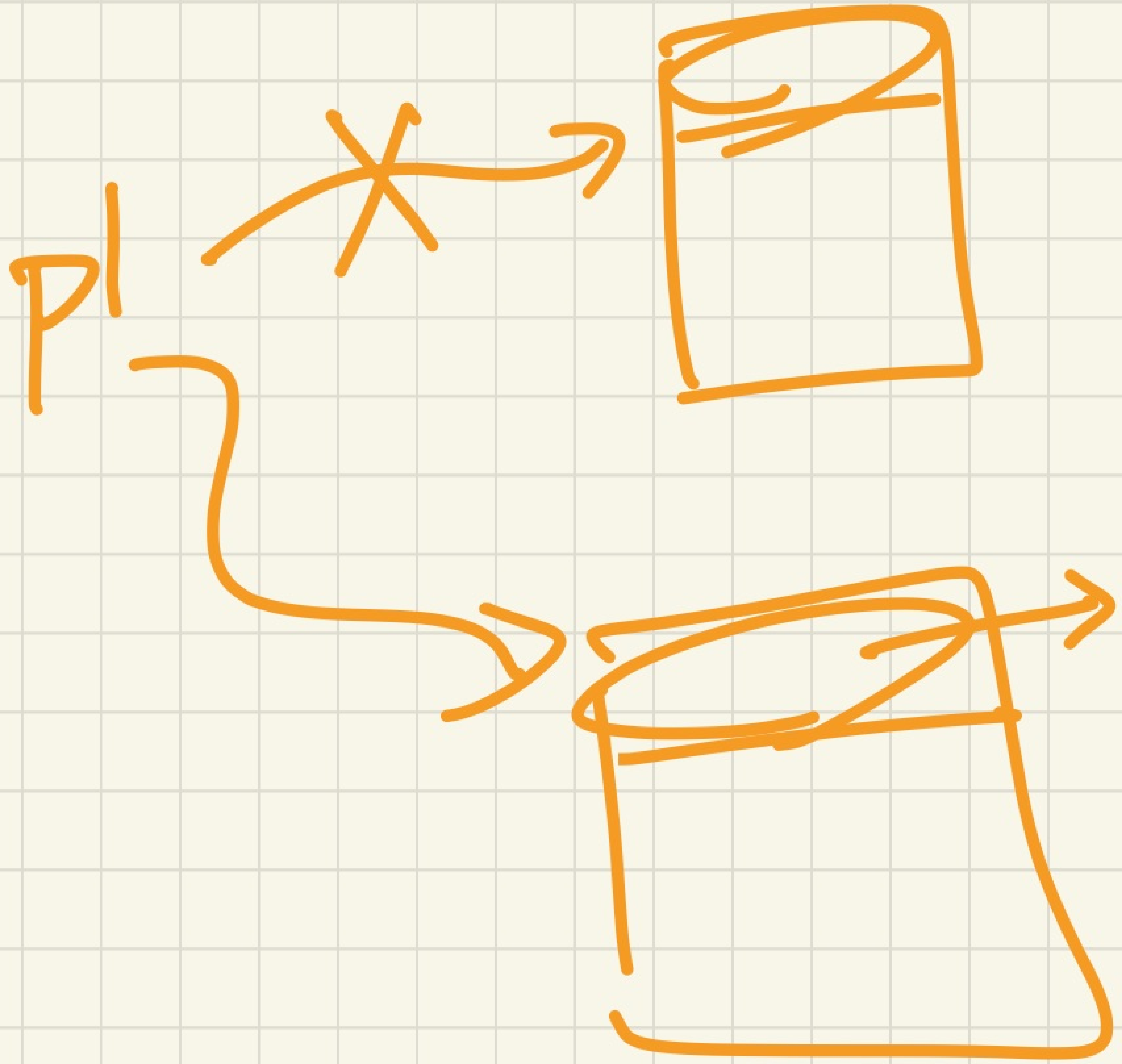
p1.equals("junk")

look up where the object pointed by p1 is.

p1

if (this == null) { return ? ; }

↳ redundat ∵ a NPE would've occured already.



o. getClass()

returns the type of object pointed by o currently.

The equals Method:

To Override or Not?

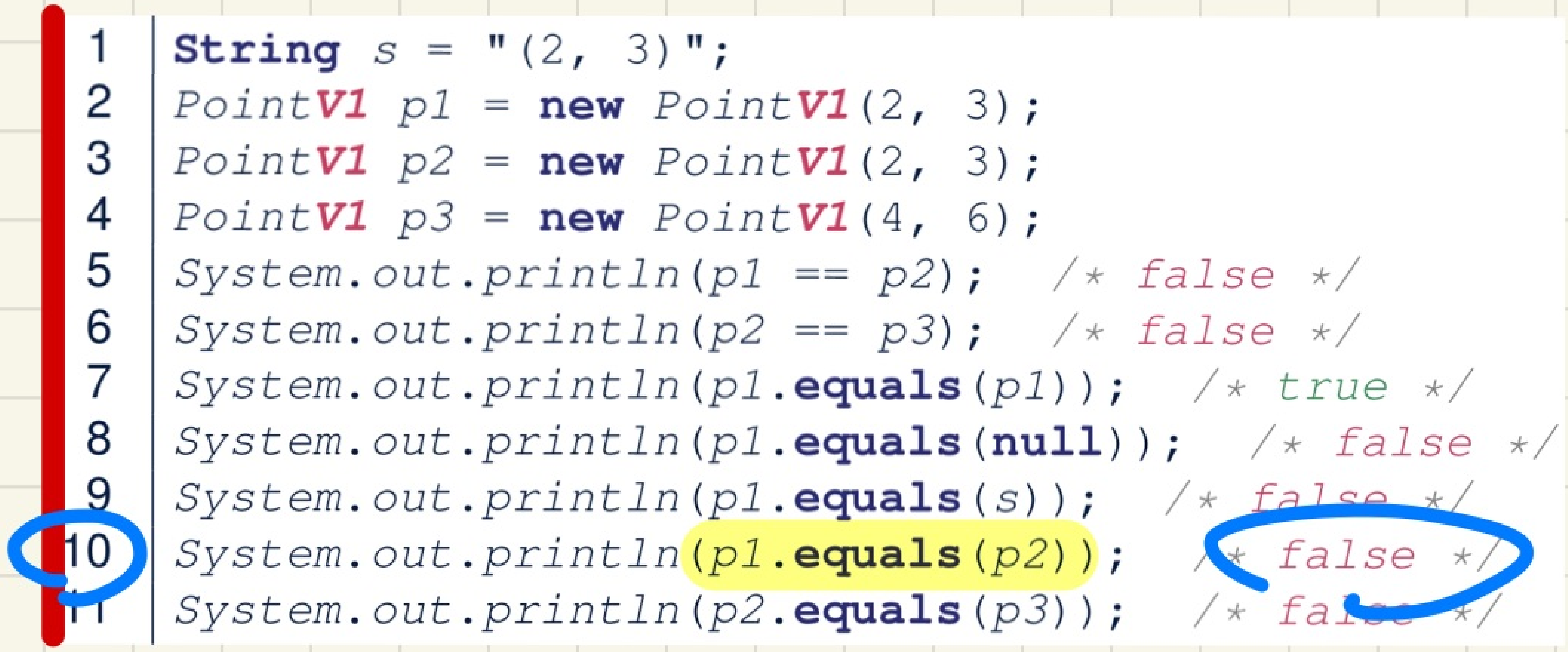
```
class Object {
  ...
  boolean equals(Object obj) {
    retuen this == obj;
  }
}
```

extends

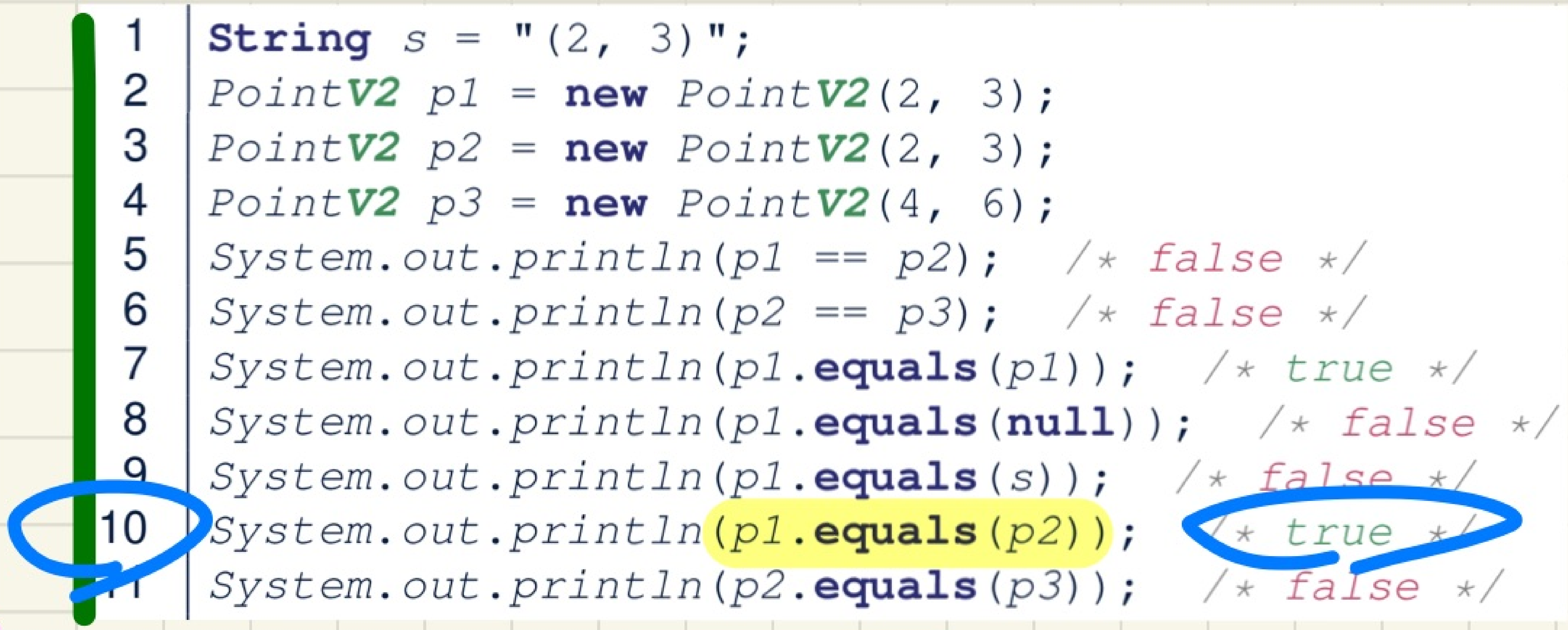
extends

```
class PointV1 {
  double x;
  double y;
  PointV1 (double x, double y) {
    this.x = x;
    this.y = y;
  }
}
```

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



```
String s = "(2, 3)";
PointV1 p1 = new PointV1(2, 3);
PointV1 p2 = new PointV1(2, 3);
PointV1 p3 = new PointV1(4, 6);
System.out.println(p1 == p2);  /* false */
System.out.println(p2 == p3);  /* false */
System.out.println(p1.equals(p1));  /* true */
System.out.println(p1.equals(null));  /* false */
System.out.println(p1.equals(s));  /* false */
System.out.println(p1.equals(p2));  /* false */
System.out.println(p2.equals(p3));  /* false */
```



```
String s = "(2, 3)";
PointV2 p1 = new PointV2(2, 3);
PointV2 p2 = new PointV2(2, 3);
PointV2 p3 = new PointV2(4, 6);
System.out.println(p1 == p2);  /* false */
System.out.println(p2 == p3);  /* false */
System.out.println(p1.equals(p1));  /* true */
System.out.println(p1.equals(null));  /* false */
System.out.println(p1.equals(s));  /* false */
System.out.println(p1.equals(p2));  /* true */
System.out.println(p2.equals(p3));  /* false */
```

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

p2.equals(p3)

p2.x p3.x

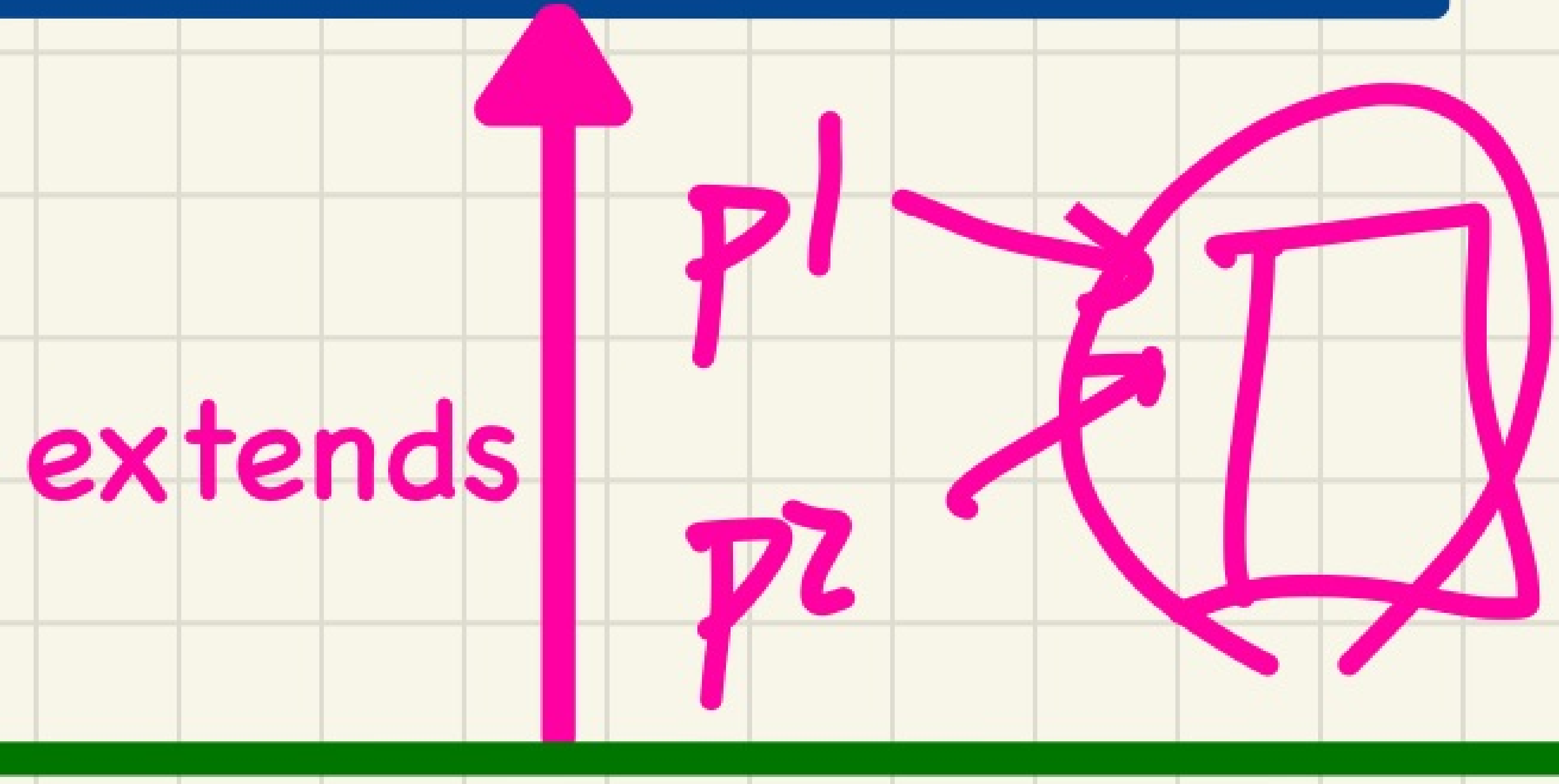
The equals Method: Overridden Version

Example 2

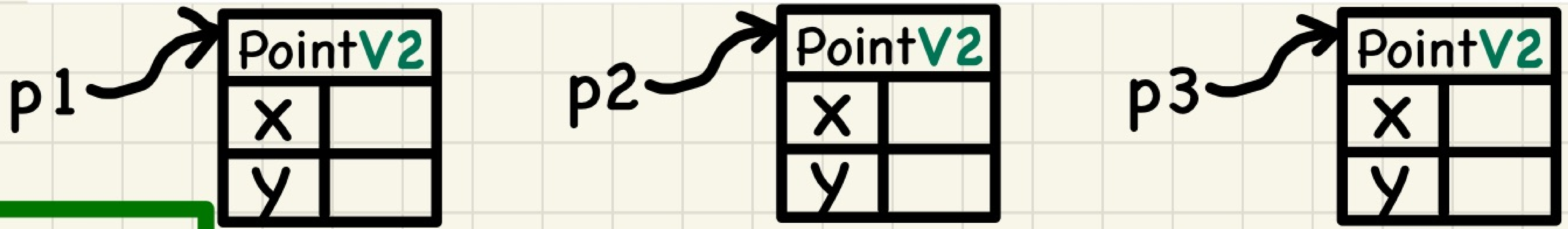
[redacted]

```
PointV2 p1 = new PointV2(3, 4);
PointV2 p2 = new PointV2(3, 4);
PointV2 p3 = new PointV2(4, 5);
System.out.println(p1 == p1);   /* [redacted] */
System.out.println(p1.equals(p1));   /* [redacted] */
System.out.println(p1 == p2);   /* [redacted] */
System.out.println(p1.equals(p2));   /* [redacted] */
System.out.println(p2 == p3);   /* [redacted] */
System.out.println(p2.equals(p3));   /* [redacted] */
```

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



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



(A) Two objects are reference-equal.

(B) Two objects are contents-equal.

- If (A) is true, then (B) is true.
  p1==p2   p1.equals(p2)
- If (B) is true, then (A) is true.
  p1.equals(p2)   p1==p2