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

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

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

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

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;
  }
}
```

The equals Method: Overridden Version

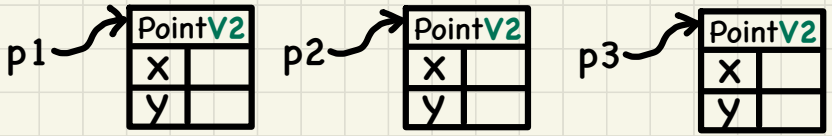
Example 1: Trace L9

```
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;
  }
}
```

```
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] */
```



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;
  }
}
```

The equals Method: Overridden Version

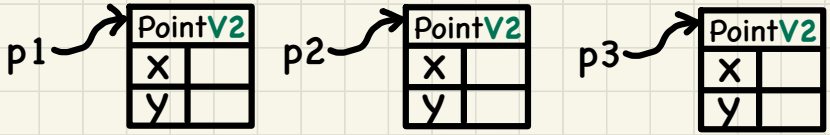
Example 1: Trace L10

```
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] */
```



The equals Method: Overridden Version

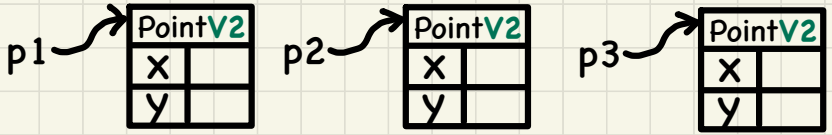
Example 1: Trace L11

```
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] */
```



The equals Method:

To Override or Not?

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

extends

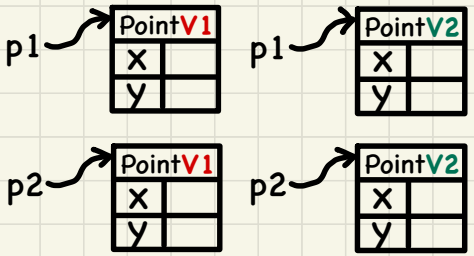
extends

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

```
public class PointV2 {
   private int x; double y;
   public 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;
   }
}
```

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

```
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));   /* true */
8  System.out.println(p1.equals(null));   /* false */
9  System.out.println(p1.equals(s));   /* false */
10 System.out.println(p1.equals(p2));   /* true */
11 System.out.println(p2.equals(p3));   /* false */
```



The equals Method: Overridden Version

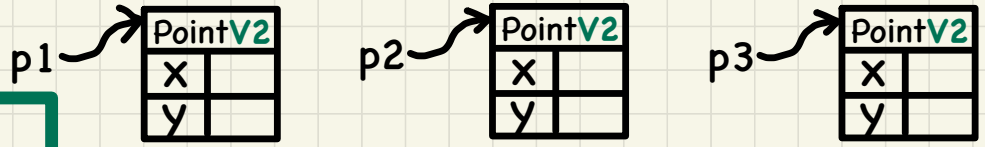
Example 2

```
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;
  }
}
```

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



(A) Two objects are reference-equal.
(B) Two objects are contents-equal.

- If (A) is true, then (B) is true.
- If (B) is true, then (A) is true.

Short-Circuit Evaluation: &&

Left Operand op1	Right Operand op2	op1 && op2
true	true	true
true	false	false
false	true	false
false	false	false

Test Inputs:

x = 0, y = 10

x = 5, y = 10

```
System.out.println("Enter x:");
int x = input.nextInt();
System.out.println("Enter y:");
int y = input.nextInt();
if(x != 0 && y / x > 2) {
    System.out.println("y / x is greater than 2");
}
else { /* !(x != 0 && y / x > 2) == (x == 0 || y / x <= 2) */
    if(x == 0) {
        System.out.println("Error: Division by Zero");
    }
    else {
        System.out.println("y / x is not greater than 2");
    }
}
```