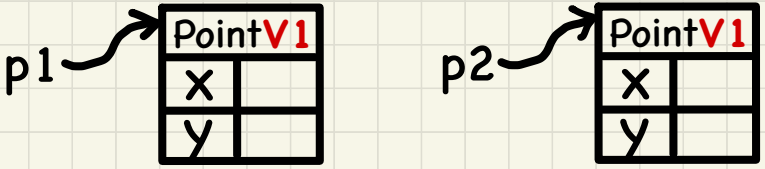
Testing Default Equality of Points in JUnit

```
@Test
public void testEqualityOfPointV1() {
  PointV1 p1 = new PointV1(3, 4); PointV1 p2 = new PointV1(3, 4);
  assertFalse(p1 == p2); assertFalse(p2 == p1);
  /* assertSame(p1, p2); assertSame(p2, p1); */ /* both fail */
  assertFalse(p1.equals(p2)); assertFalse(p2.equals(p1));
  assertTrue(p1.getX() == p2.getX() && p2.getY() == p2.getY());
}
```




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

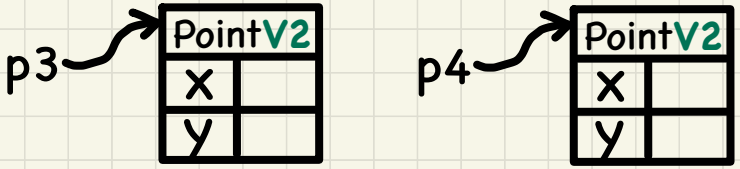
extends

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

Testing Overridden Equality of Points in JUnit

```java
@Test
public void testEqualityOfPointV2() {
  PointV2 p3 = new PointV2(3, 4); PointV2 p4 = new PointV2(3, 4);
  assertFalse(p3 == p4); assertFalse(p4 == p3);
  /* assertSame(p3, p4); assertSame(p4, p4); */ /* both fail */
  assertTrue(p3.equals(p4)); assertTrue(p4.equals(p3));
  assertEquals(p3, p4); assertEquals(p4, p3);
}
```

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



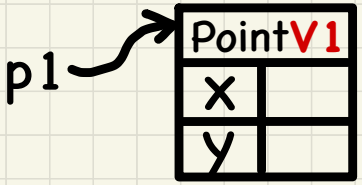
extends

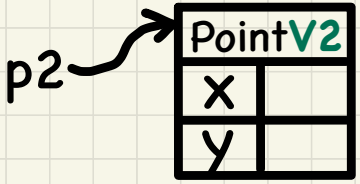
```java
public class PointV2 {
  private int x;
  private int y;
  public PointV2 (int x, int y) { ... }
  public boolean equals(Object obj) {
    if(this == obj) { return true; }
    if(obj == null) { return false; }
    if(this.getClass() != obj.getClass()) { return false }
    Point other = (PointV2)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

Testing Equality of Points in JUnit: Default vs. Overridden

```java
@Test
public void testEqualityOfPointV1andPointv2() {
  PointV1 p1 = new PointV1(3, 4); PointV2 p2 = new PointV2(3, 4);
  /* These two assertions do not compile because p1 and p2 are of different types. */
  /* assertFalse(p1 == p2); assertFalse(p2 == p1); */
  /* assertSame can take objects of different types and fail. */
  /* assertSame(p1, p2); */ /* compiles, but fails */
  /* assertSame(p2, p1); */ /* compiles, but fails */
  /* version of equals from Object is called */
  assertFalse(p1.equals(p2));
  /* version of equals from PointP2 is called */
  assertFalse(p2.equals(p1));
}
```

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





extends

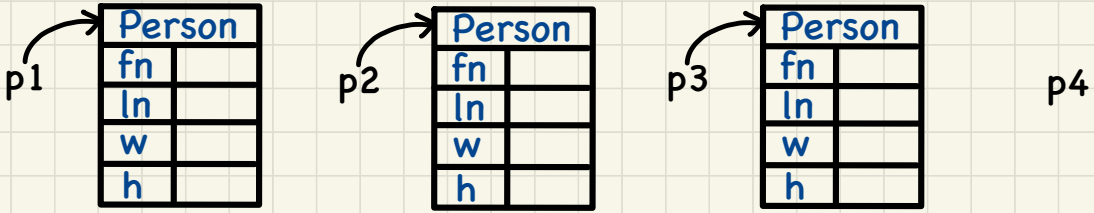
extends

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

```java
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;
  }
}
```

Testing Equality of Person/PersonCollector in JUnit (1)

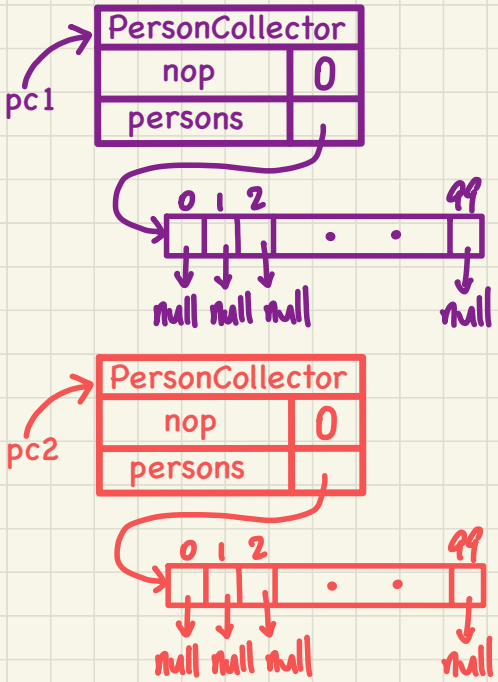
```
@Test
public void testPersonCollector() {
  Person p1 = new Person("A", "a", 180, 1.8);
  Person p2 = new Person("A", "a", 180, 1.8);
  Person p3 = new Person("B", "b", 200, 2.1);
  Person p4 = p3;
  assertFalse(p1 == p2); assertTrue(p1.equals(p2));
  assertTrue(p3 == p4); assertTrue(p3.equals(p4));
```




```
public class Person {
  private String firstName; private String lastName;
  private double weight; private double height;
  public boolean equals(Object obj) {
    if(this == obj) { return true; }
    if(obj == null || this.getClass() != obj.getClass()) { return false; }
    Person other = (Person) obj;
    return
        this.weight == other.weight
     && this.height == other.height
     && this.firstName.equals(other.firstName)
     && this.lastName.equals(other.lastName);
  }
}
```

Testing Equality of Person/PersonCollector in JUnit (2) (continued from testPersonCollector)

```
PersonCollector pc1 = new PersonCollector();
PersonCollector pc2 = new PersonCollector();
assertFalse(pc1 == pc2); assertTrue(pc1.equals(pc2));
```



Q: How about **assertTrue**(pc2.**equals**(pc1))?

```
class PersonCollector {
  private Person[] persons;
  private int nop; /* number of persons */
  public PersonCollector() { ... }
  public void addPerson(Person p) { ... }
  public int getNop() { return this.nop; }
  public Person[] getPersons() { ... }
}
```

```
public boolean equals(Object obj) {
  if(this == obj) { return true; }
  if(obj == null || this.getClass() != obj.getClass()) { return false; }
  PersonCollector other = (PersonCollector) obj;
  boolean equal = false;
  if(this.nop == other.nop) {
    equal = true;
    for(int i = 0; equal && i < this.nop; i ++) {
      equal = this.persons[i].equals(other.persons[i]);
    }
  }
  return equal;
}
```

Testing Equality of Person/PersonCollector in JUnit (3)

(continued from testPersonCollector)

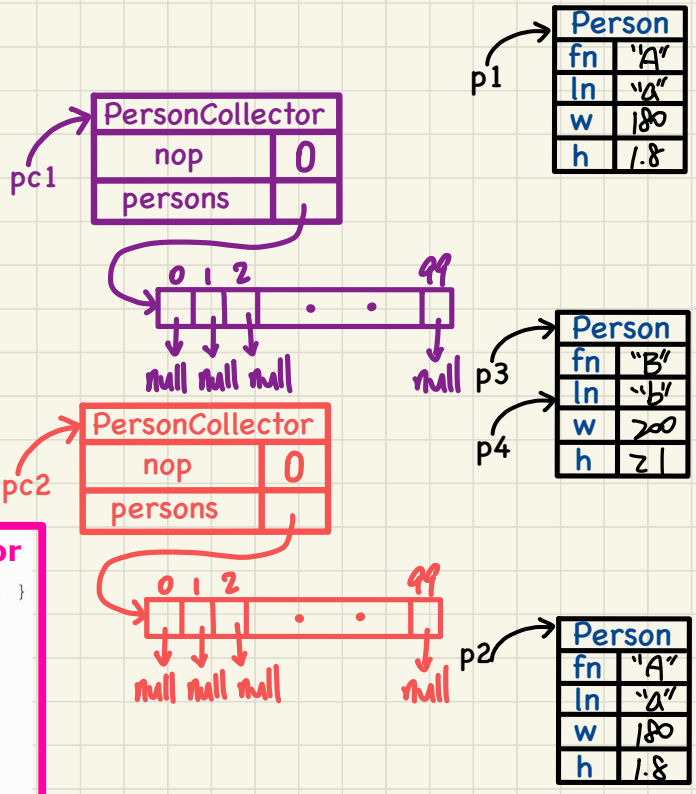
```
pc1.addPerson(p1);
assertFalse(pc1.equals(pc2));
pc2.addPerson(p2);
assertFalse(pc1.getPersons()[0] == pc2.getPersons()[0]);
assertTrue(pc1.getPersons()[0].equals(pc2.getPersons()[0]));
assertTrue(pc1.equals(pc2));
pc1.addPerson(p3);
pc2.addPerson(p4);
assertTrue(pc1.getPersons()[1] == pc2.getPersons()[1]);
assertTrue(pc1.getPersons()[1].equals(pc2.getPersons()[1]));
assertTrue(pc1.equals(pc2));
```

Person

```
public boolean equals(Object obj) {
  if(this == obj) { return true; }
  if(obj == null || this.getClass() != obj.getClass()) { return false; }
  Person other = (Person) obj;
  return
      this.weight == other.weight
   && this.height == other.height
   && this.firstName.equals(other.firstName)
   && this.lastName.equals(other.lastName);
}
```

PersonCollector

```
public boolean equals(Object obj) {
  if(this == obj) { return true; }
  if(obj == null || this.getClass() != obj.getClass()) { return false; }
  PersonCollector other = (PersonCollector) obj;
  boolean equal = false;
  if(this.nop == other.nop) {
    equal = true;
    for(int i = 0; equal && i < this.nop; i ++) {
      equal = this.persons[i].equals(other.persons[i]);
    }
  }
  return equal;
}
```



Testing Equality of Person/PersonCollector in JUnit (4)

(continued from testPersonCollector)

```
pc1.addPerson(new Person("A", "a", 175, 1.75));
pc2.addPerson(new Person("A", "a", 165, 1.55));
assertFalse(pc1.getPersons()[2] == pc2.getPersons()[2]);
assertFalse(pc1.getPersons()[2].equals(pc2.getPersons()[2]));
assertFalse(pc1.equals(pc2));
```

Person	
fn	
ln	
w	
h	

Person

```
public boolean equals(Object obj) {
  if(this == obj) { return true; }
  if(obj == null || this.getClass() != obj.getClass()) { return false; }
  Person other = (Person) obj;
  return
       this.weight == other.weight
    && this.height == other.height
    && this.firstName.equals(other.firstName)
    && this.lastName.equals(other.lastName);
}
```

PersonCollector

```
public boolean equals(Object obj) {
  if(this == obj) { return true; }
  if(obj == null || this.getClass() != obj.getClass()) { return false; }
  PersonCollector other = (PersonCollector) obj;
  boolean equal = false;
  if(this.nop == other.nop) {
    equal = true;
    for(int i = 0; equal && i < this.nop; i ++) {
      equal = this.persons[i].equals(other.persons[i]);
    }
  }
  return equal;
}
```