CS2030 Midterm Discussion

17/18 Semester 2

To ask for regrade: Please submit your script back to your TA with a note. We will collect and get back.

Question 2. Recall that we can override the method `equals` in the class `Point` (as defined in CS2030) so that two `Point` objects are equal if they have the same x and y coordinates. Overriding `equals` may or may not violate LSP. It depends on what the specified properties of `equals` in the class Object are.

Which of the following property of `equals`, **if specified**, would cause `Point` to violate the LSP?

For two variables of type `Object`, `o1` and `o2`, `o1.equals(o2)` is true if and only `o1 == o2`

This would cause `Point` to violate LSP because the behavior of `Point` is now different.

For two variables of type `Object`, `o1` and `o2`, `o1.equals(o2)` is true if and only `o2.equals(o1)`.

This is the symmetric property.

The equals() method in `Point` does not violate this property, so no violation of LSP.

For two variables of type `Object`, `o1` and `o2`, `o1.equals(o2)` is true if and only if `o1.equals(o3) => o3.equals(o2)` for some `o3`

The `equals()` method in `Point` does not violate this property, so no violation of LSP.

Answer: A (i) only.

Question 5.

```
class Out {
  int x;
  class In {
    int y;
  }
  void foo(int z) {
    x = 1; // (A)
    z = 1; // (B)
    class Local extends In {
      void bar() {
        int w;
        w = x; // (C)
        w = y; // (D)
        w = z; // (E)
      }
    }
  }
}
```

Question 5.

```
class Out {
  int x;
  class In {
    int y;
  }
  void foo(int z) {
    x = 1; // (A)
    z = 1; // (B)
    class Local extends In {
      void bar() {
        int w;
        w = x; // (C)
        w = y; // (D)
        w = z; // (E)
      }
    }
  }
}
```

Inner class

z is local variable

Local class

Question 5.

Answer: A

```
class Out {
  int x;
  class In {
    int y;
  }
  void foo(int z) {
    x = 1; // x is not a local variable. OK
    z = 1; // not ok - captured so must be final.
    class Local extends In {
      void bar() {
        int w;
        w = x; // ok to access enclosing class
        w = y; // ok to access parent class
        w = z; // not ok – z is not final
      }
    }
  }
}
```

Question 5.

Answer: A

```
class Out {
  int x;
  class In {
    int y;
  }
  void foo(int z) {
    x = 1; // x is not a local variable. OK
    z = 1; // not ok - captured so must be final.
    class Local extends In {
      void bar() {
        int w;
        w = x; // ok to access enclosing class
        w = y; // ok to access parent class
        w = z; // not ok – z is not final
      }
    }
  }
}
```

Question 9.

A module has multiple assessments. There are three types of assessments: lab assignment, test, and project, each to be graded in a different way.

The nouns

A **module** has multiple **assessments**. There are three types of **assessments**: **lab assignment**, **test**, and **project**, each to be graded in a different way.

Module

Assessment

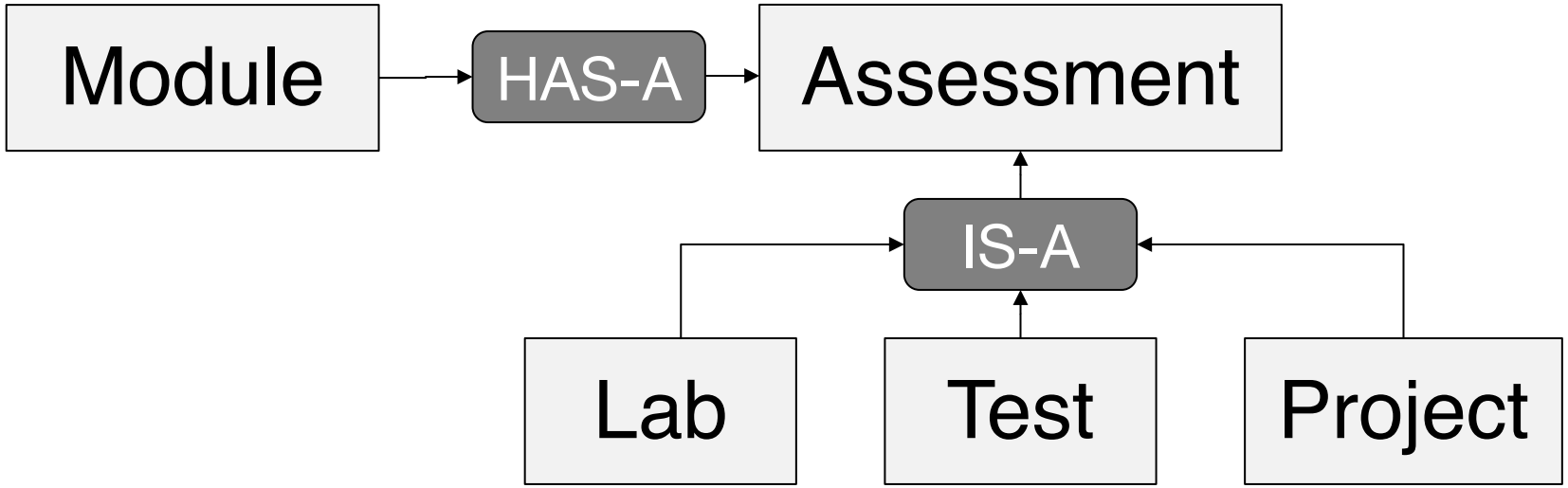
Lab

Test

Project

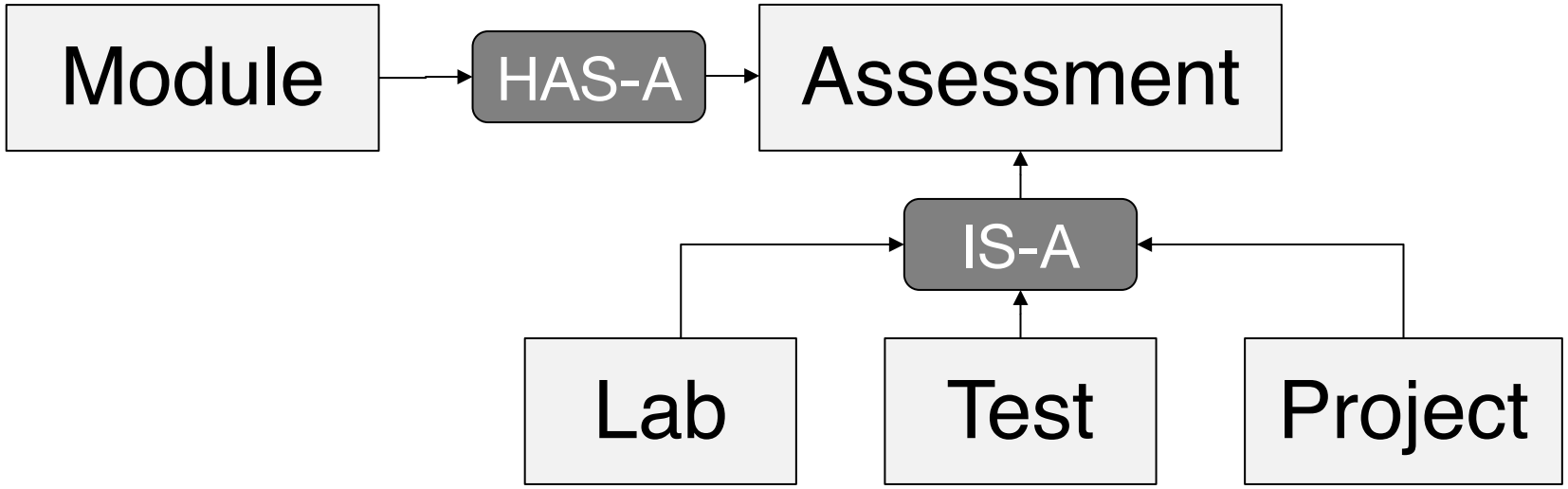
Question 10.

A module has multiple assessments. There are three types of assessments: lab assignment, test, and project, each to be graded in a different way.



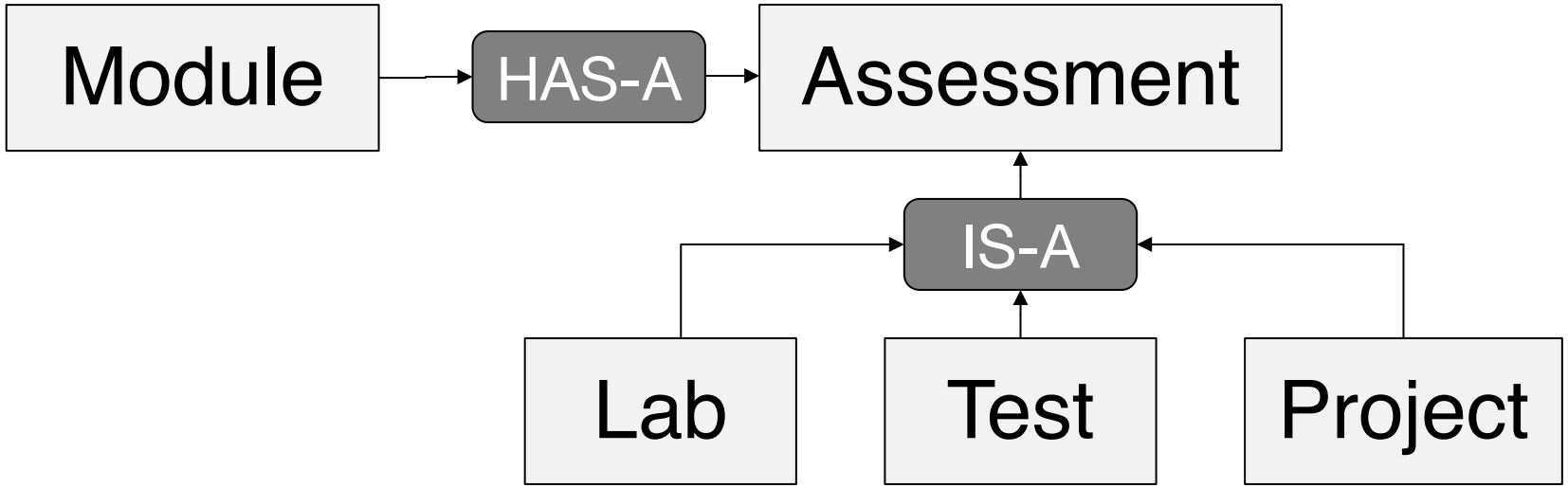
The relationship

A module **has** multiple assessments. There are **three types of assessments**: lab assignment, test, and project, each to be graded in a different way.



Overriding / polymorphism

A module has multiple assessments. There are three types of assessments: lab assignment, test, and project, each to be **graded in a different way**.



Question 10

```
class A {
      :
   @Override
   int hashCode() {
      return 8888;
   }
}
```

(b) Why is it a bad practice?

- **Violate LSP?**
  No. In fact, part (a) says that it still satisfies the required property of

- **Shouldn’t use magic number**
  Yes. But not the most glaring error (0.5 marks)

- **All objects will be equal!**
  No. Note that Property P does not say

```
x.hashCode() == y.hashCode()
=> x.equals(y)
```

- **HashMap / HashSet will stop working**
  No. Still can hash into buckets.
  Can still search for equality.

- **HashMap / HashSet will be slower**
  Yes. All elements will be in the same bucket now. Have to search and compare each one to find a match.

- **equals() cannot use hashCode() to filter out non-equal objects.**
  Yes.

```
boolean equals(Object o) {
  if (hashCode() != o.hashCode()) {
   return false;
  }
  // do usual comparison
}
```

Question 11

Already discussed in lecture.

Question 12

```
void printInteger(List<Integer> list) {
  for (Integer i : list) {
    if (i.byteValue() > 0) {
      // print
    }
  }
}
```

Question 12

```
void printDouble(List<Double> list) {
  for (Double i : list) {
    if (i.byteValue() > 0) {
      // print
    }
  }
}
```

Question 12

```
void printLong(List<Long> list) {
  for (Long i : list) {
    if (i.byteValue() > 0) {
      // print
    }
  }
}
```

Copy and paste?

Not if you have taken CS2030!

Write only one version.
Accept `List<Integer>`,
`List<Double>`, `List<Long>`, etc.

Question 12

```
void printNumber(List<Number> list) {
  for (Number i : list) {
    if (i.byteValue() > 0) {
      // print
    }
  }
}
```

No. Can only accept List<Number>, not List<Integer>, etc.

Question 12

```
void printNumber(List<?> list) {
  for (Object i : list) {
    if (i.byteValue() > 0) {
      // print
    }
  }
}
```

No. Too general. Cannot call `byteValue()`

Question 12

```
void print (List<? extends Number> list)
{
   for (Number i : list) {
      if (i.byteValue() > 0) {
         // print
      }
   }
}
```

Question 12

```
void print (List<T extends Number> list)
{
  for (T i : list) {
    if (i.byteValue() > 0) {
      // print
    }
  }
}
```

Close. What is T? `<T extends Number>` won't compile

Question 12

```
<T extends Number> void print (List<T> list)
{
  for (T i : list) {
    if (i.byteValue() > 0) {
      // print
    }
  }
}
```

What's the diff between `List<?>` and `List<T>`?

List<?>

List of something (but I don’t know what). Can only read Object from it. Can’t add anything (since I don’t know what is stored)

List<T>

List of some type T, to be determined by the argument passed in. Can read T from it. Can add T to it (what ever T is).

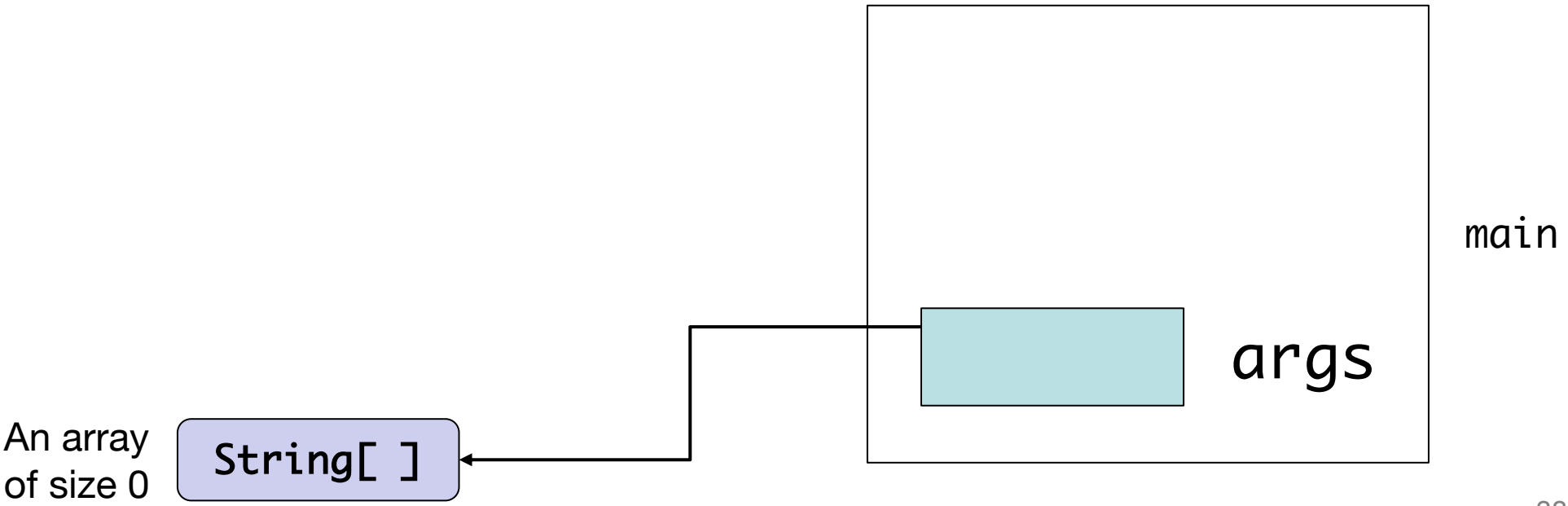
Question 13

Heap & Stack

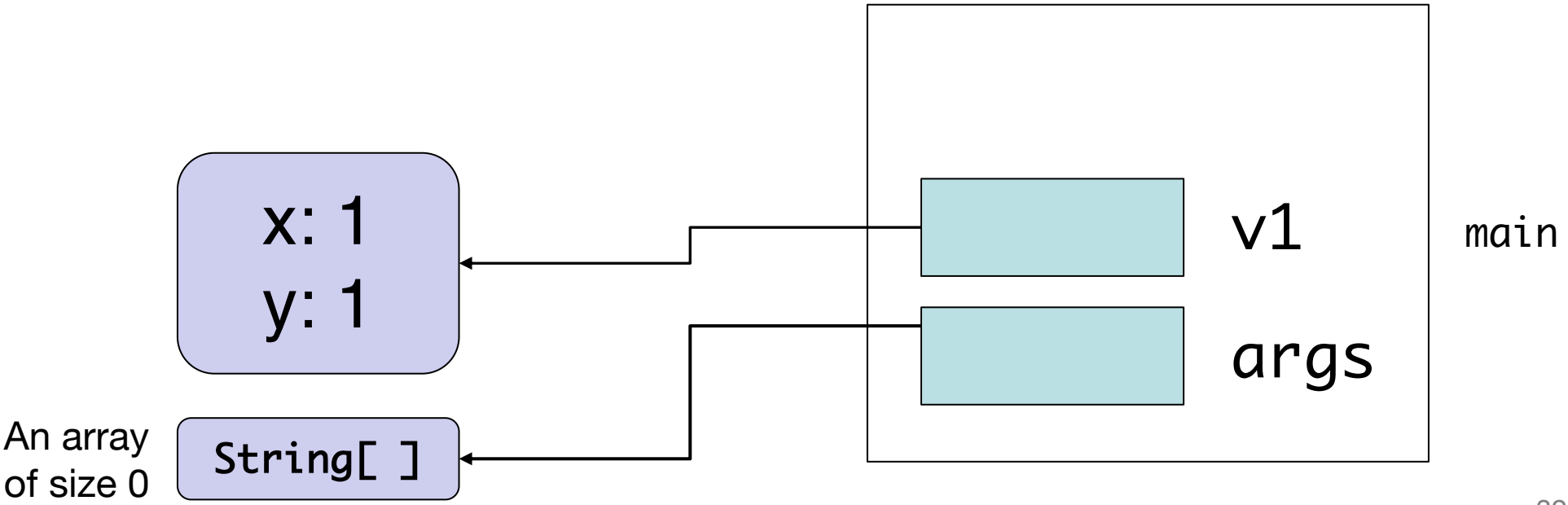
```
class Vector2D {

  private double x;
  private double y;

  Vector2D(double x, double y) {
    this.x = x;
    this.y = y;
  }

  void add(Vector2D v) {
    this.x += v.x;
    this.y += v.y;
    // line A
  }
}
```

```
public static void main(String[] args) {
  // no argument being passed in
}
```

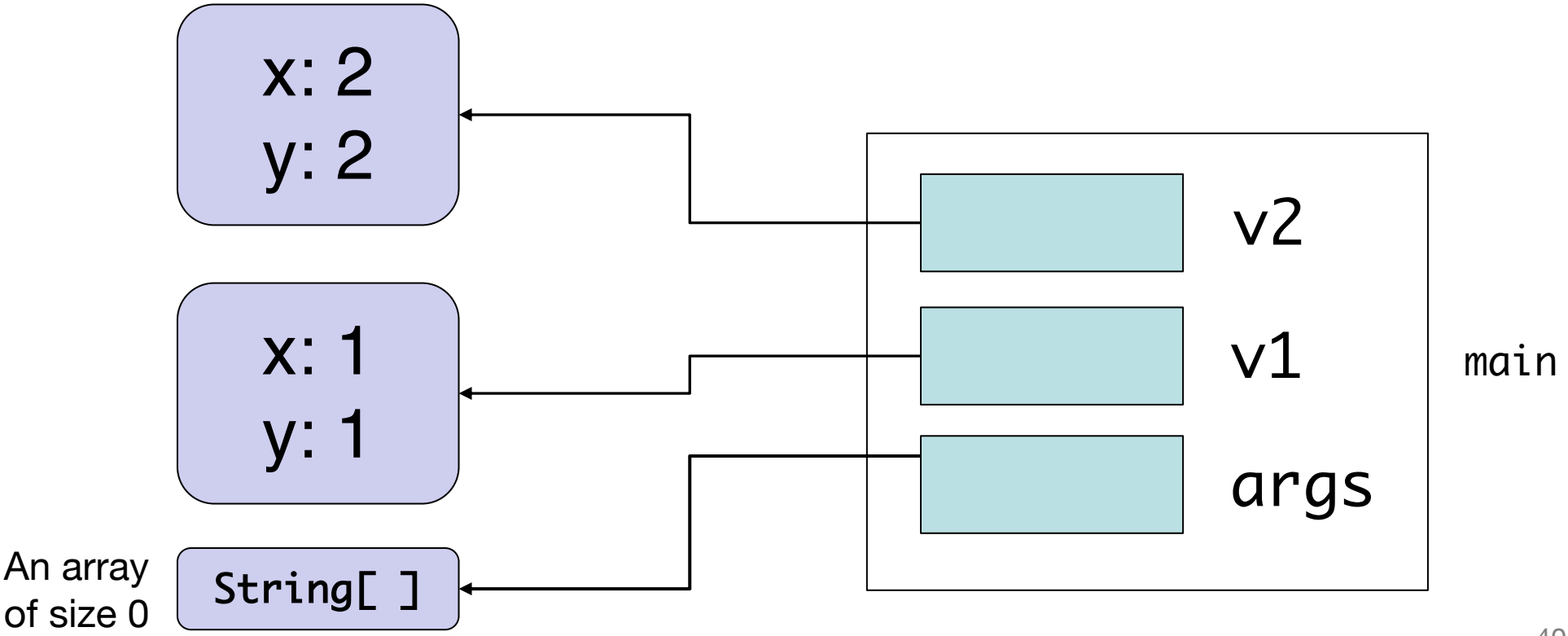
Showing null is also ok. But convention for JVM is to have 0-length String array.



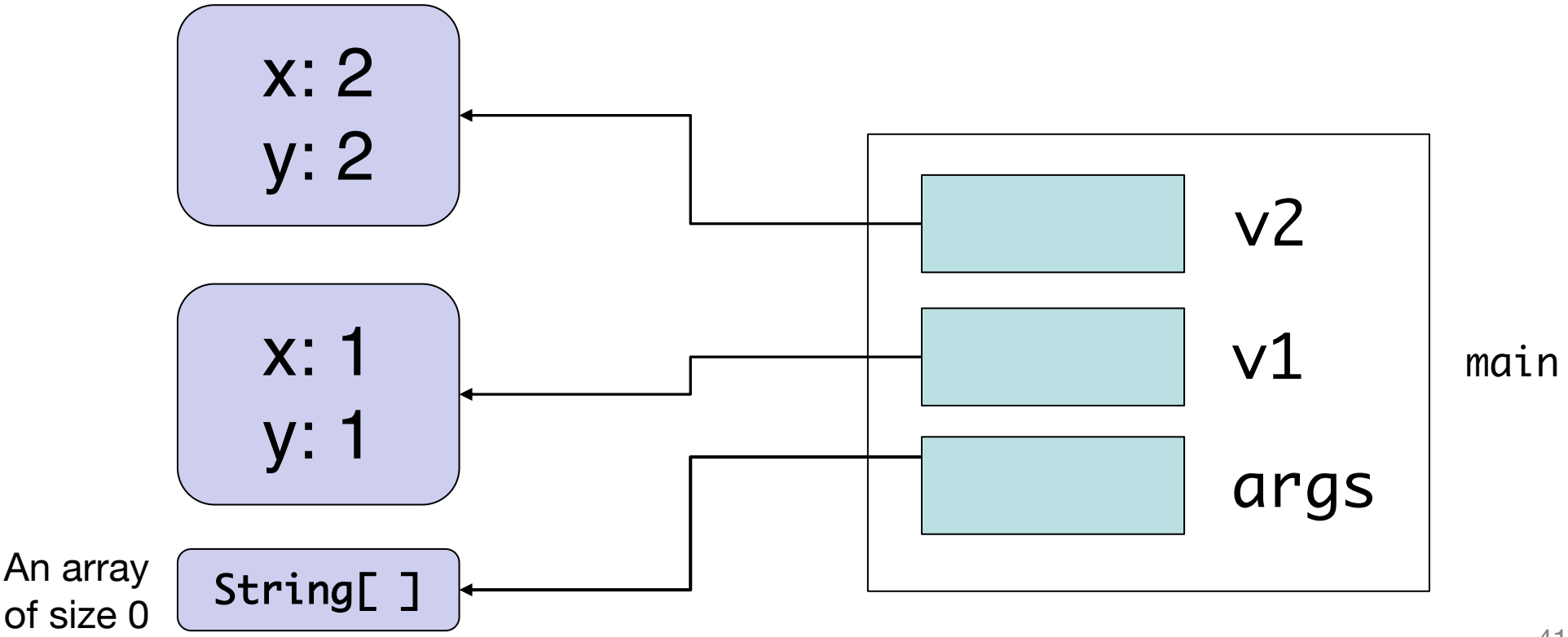
```
public static void main(String[] args) {
  Vector2d v1 = new Vector2D(1, 1);
}
```



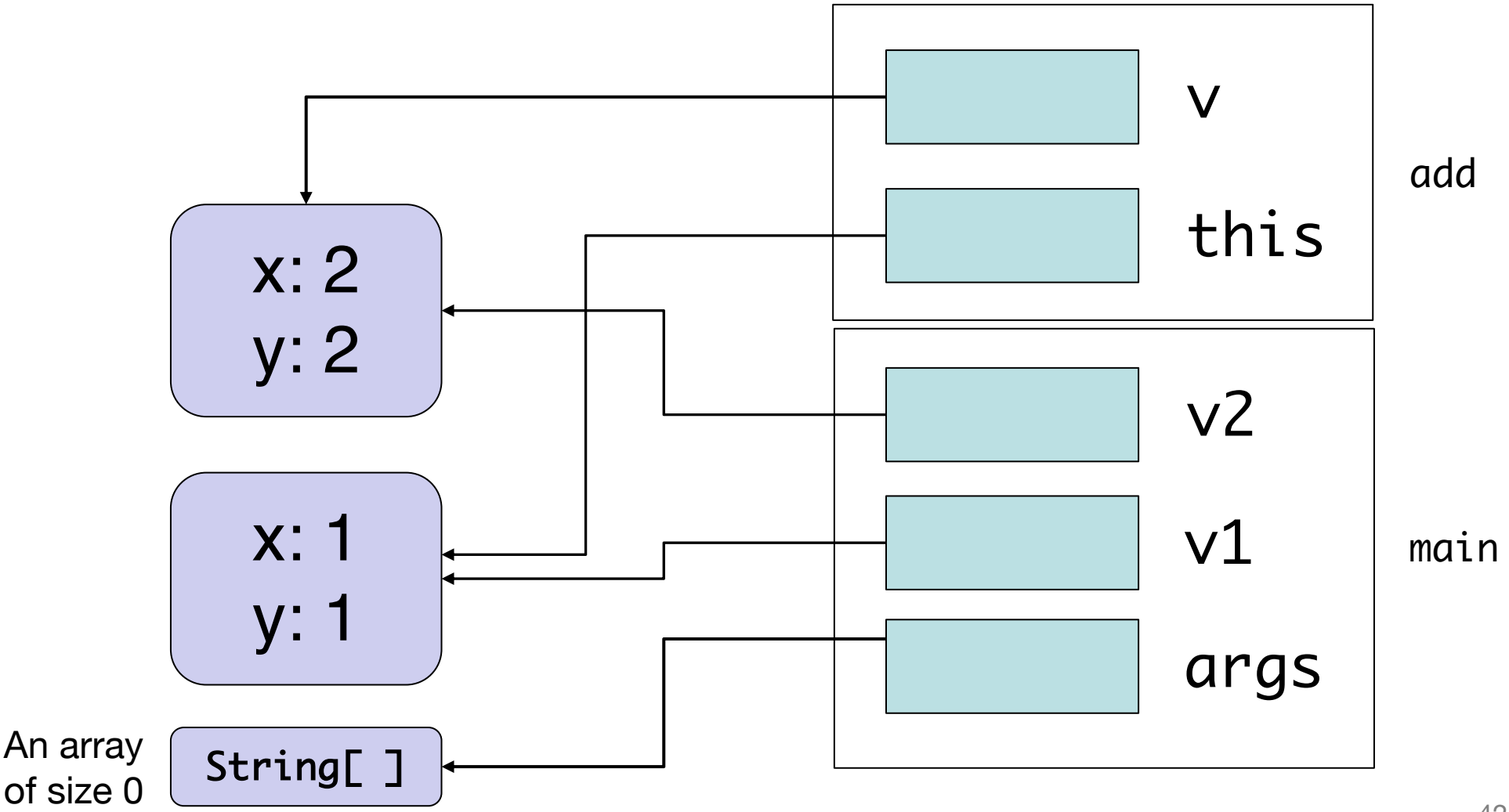
```
public static void main(String[] args) {
  Vector2D v1 = new Vector2D(1, 1);
  Vector2D v2 = new Vector2D(2, 2);
}
```



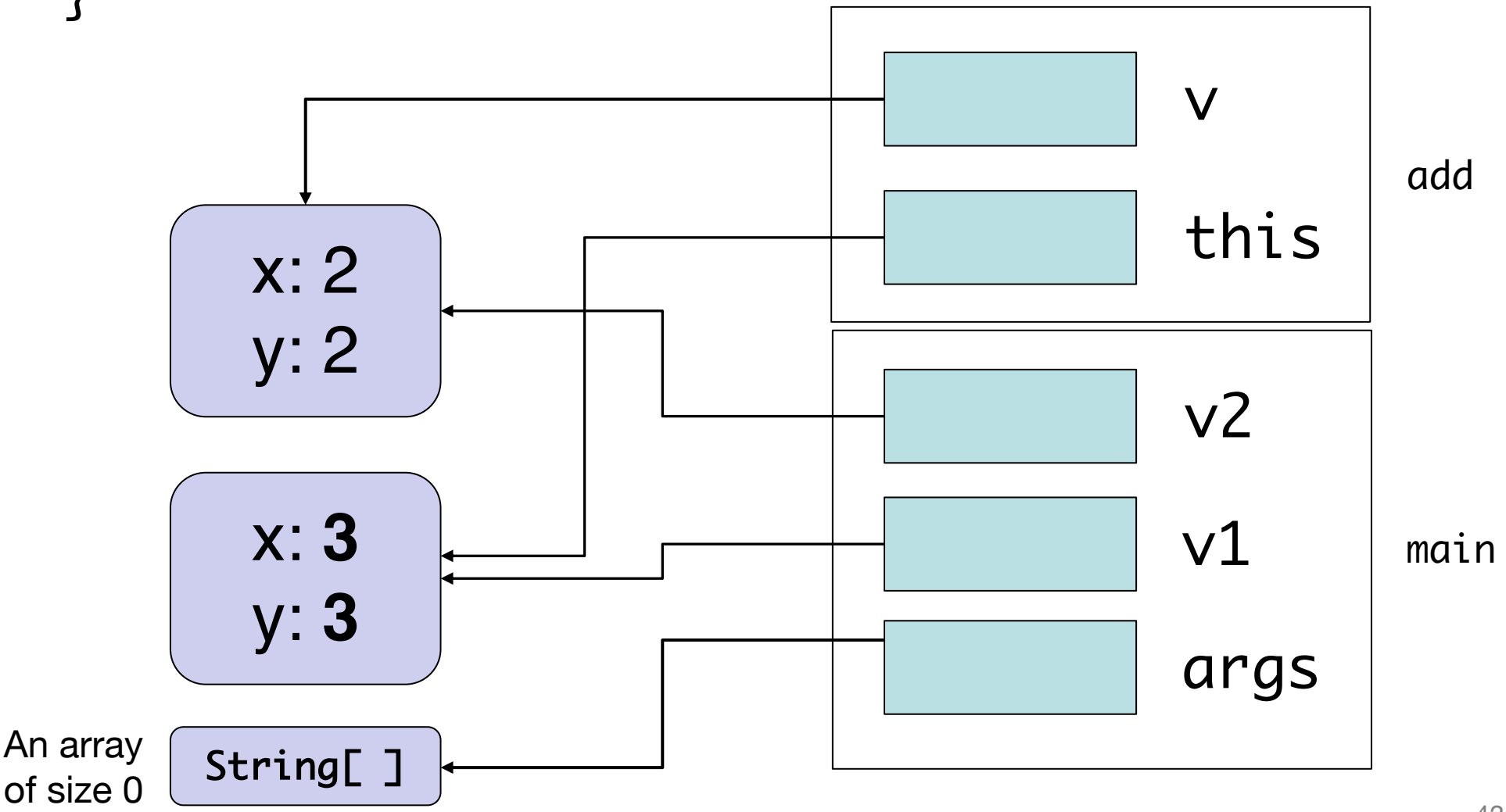
```
public static void main(String[] args) {
   Vector2D v1 = new Vector2D(1, 1);
   Vector2D v2 = new Vector2D(2, 2);
   v1.add(v2);
}
```

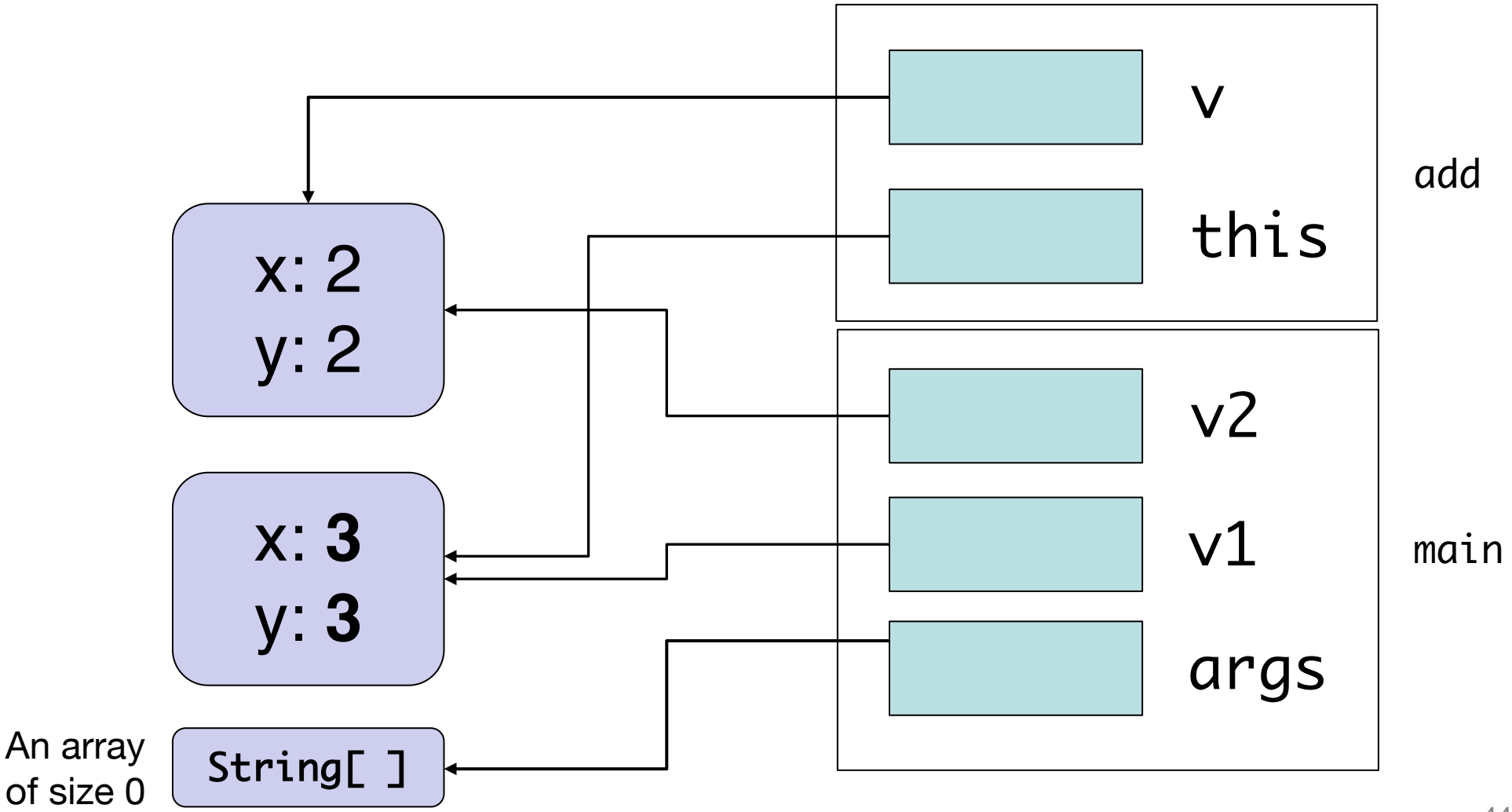


```
void add(Vector2D v) {
```



```
void add(Vector2D v) {
  this.x += v.x;
  this.y += v.y;
}
```




v
this
add
x: 2
y: 2
v2
x: 3
y: 3
v1
main
args
An array
of size 0
String[ ]