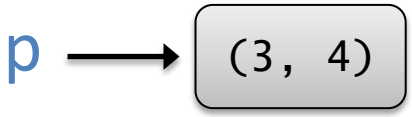
References and Vectors of Objects

References

- A reference is a variable that points to na object
  - The referenced object may not be defined. In that case the reference is null
  - Has a type (class)
    - Can only point to objects of that class

```
Point p = new Point(3, 4)
```

type *name* *Object creation*



Reference types and value types

- Reference types
  - Identity relevant
  - Equality usually irrelevant

Classes Java.

- Value types
  - Identity irrelevant
  - Equality relevant

Tipos primitivos em Java.

- Value types: two objects are the same in the content is the same even if they have different references

Primitive types vs. classes

- Varibles of primitive types (int, double, etc)
  - Keep the value
  - Attribution changes value

- Class varables
  - Keep a reference to an object
  - Attribution changes the reference, not the referenced object

References and objects: declaration and construction

Reference construction

```
Class var;
```

Declation of reference *var*, not initialized, may reference object of class *Class*.

```
var = null;
```

Inicialization of reference with special value `null`.

Reference construction

```
Class anotherVar = null;
```

Construction of reference with null initial value.

```
Class yetAnotherVar = new Class(…);
```

Construction of new object and a reference to it.

Object construction

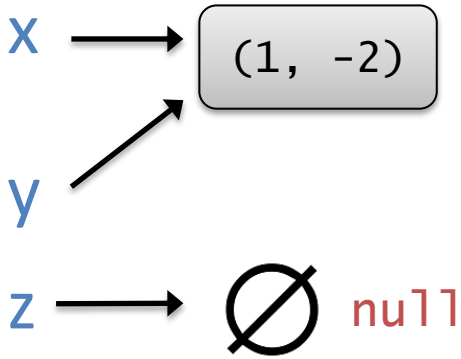
Attribution: value vs. reference

Primitive types (int, boolean, etc)

```
int a = 7;

int b = a;

int c;
```

Reference types (Classes and vectors)

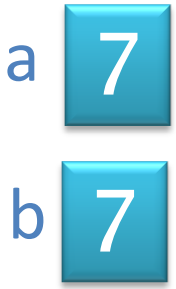
```
Point x = new Point(1, -2);

Point y = x;

Point z;
```



Equality: value vs. reference

Primitive types (int, boolean, etc)	Reference types (Classes and vectors)

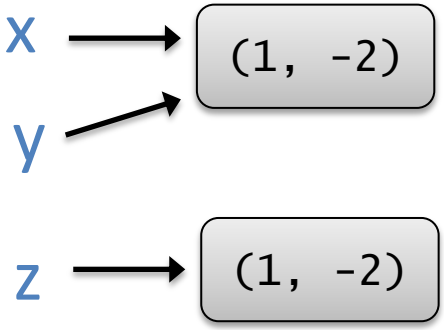
```
int a = 7;

int b = 7;
```



a == b

true

```
Point x = new Point(1, -2);

Point y = x;

Point z = new Point(1, -2);
```



x == y

true

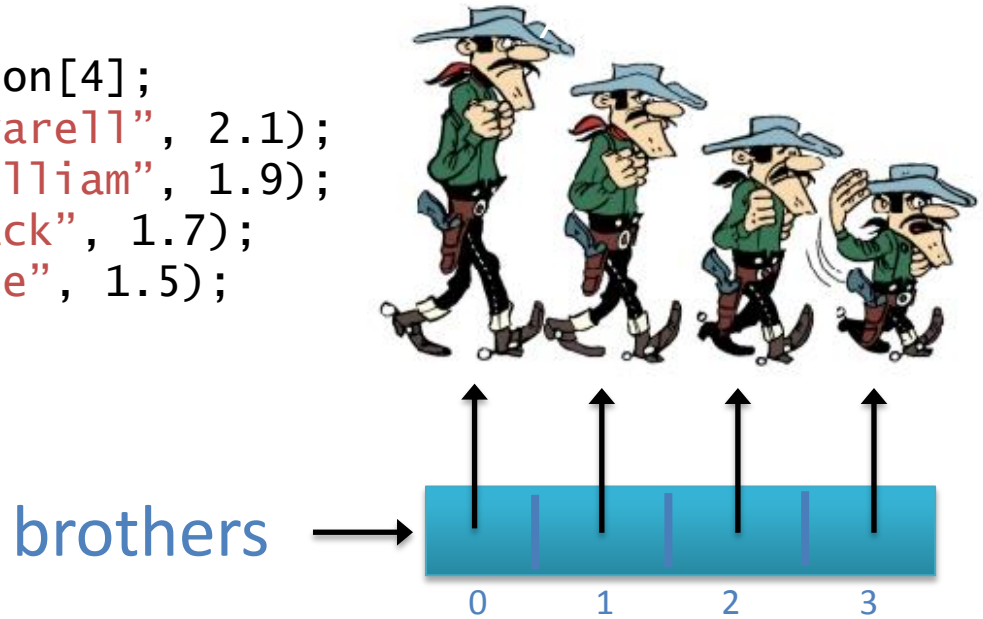
x == z

false

Object Vectors

- It is possible to create vectors of objects of a class

```
Person[] brothers = new Person[4];
brothers[0] = new Person("Avarell", 2.1);
brothers[1] = new Person("William", 1.9);
brothers[2] = new Person("Jack", 1.7);
brothers[3] = new Person("Joe", 1.5);
```



Object Vectors

```
Person[] brothers = new Person[4];
brothers[0] = new Person("Avarell", 2.1);
brothers[1] = new Person("William", 1.9);
brothers[2] = new Person("Jack", 1.7);
brothers[3] = new Person("Joe", 1.5);
```

```
boolean b = brothers[3].isTall();
String s = brothers[1].getFirstName();
```

b  false

s → "William"

Objects containing vectors

```
public class Family {
   private String surname;
   private Person[] members;

   public Family(String surname, Person[] members) {
      this.surname = surname;
      this.members = members;
   }

   public String toString() {
      String newline = System.getProperty("line.separator");
      String text = " ";
      for(int i = 0; i < members.length; i++) {
          text = text + members[i].getFirstName() + " " + surname + newline;
      }

      return text;
   }

   ...
}
```

Objects containing vectors

```
Person[] brothers = new Person[4];
brothers[0] = new Person("Avarell", 2.1);
brothers[1] = new Person("William", 1.9);
brothers[2] = new Person("Jack", 1.7);
brothers[3] = new Person("Joe", 1.5);
Family daltons = new Family("Dalton", brothers);
System.out.println(daltons);
```

```
> Avarell Dalton
> William Dalton
> Jack Dalton
> Joe Dalton
```

Objects containing vectors

```
public class Family {
   private String surname;
   private Person[] members;

   ...

   public double averageHeight() {
      double heightsSum = 0;

      for(int i = 0; i < members.length; i++) {
          heightsSum = heightsSum + members[i].getHeight();
      }

      return heightsSum / members.length;
   }

}
```

Objects containing vectors

```
Person[] brothers = new Person[4];
brothers[0] = new Person("Avarell", 2.1);
brothers[1] = new Person("William", 1.9);
brothers[2] = new Person("Jack", 1.7);
brothers[3] = new Person("Joe", 1.5);
Family daltons = new Family("Dalton", brothers);

double d = daltons.averageHeight();
```

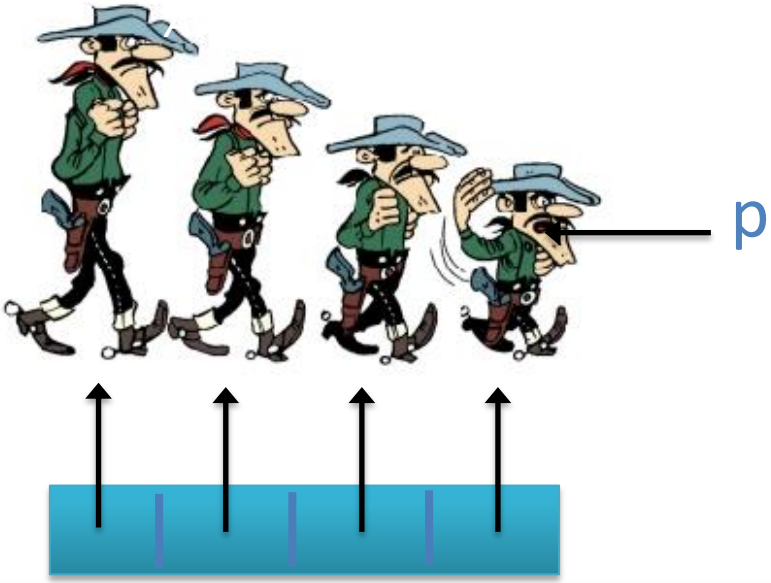
d 1.8

Objects containing vectors

```
public class Family {
   private String surname;
   private Person[] members;

   ...

   public Person smallest() {
      Person smallest = members[0];
      double lowestHeight = members[0].getHeight();

      for(int i = 1; i < members.length; i++) {
          double h = members[i].getHeight();
          if(h < lowestHeight) {
             lowestHeight = h;
             smallest = members[i];
          }
      }

      return smallest;
   }
}
```

Objects containing vectors

```
Person[] brothers = new Person[4];
brothers[0] = new Person("Avarell", 2.1);
brothers[1] = new Person("William", 1.9);
brothers[2] = new Person("Jack", 1.7);
brothers[3] = new Person("Joe", 1.5);
Family daltons = new Family("Dalton", brothers);

Person p = daltons.smallest();
```



More information / References

- Y. Daniel Liang, "Introduction to Java Programming" 7th Ed. Prentice-Hall, 2010.

Summary

- References
- Vectors of Objects