



Referências e Vectors de Objectos

Referências

- Uma **referência** é uma variável que aponta para um objecto
 - O objecto referenciado pode não estar definido, e neste caso a referência está a **null**
 - Tem um tipo associado (classe)
 - Só pode apontar para objectos desse tipo

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

tipo *nome* *criação do objecto*

Tipos de referência e tipos de valor

- Tipos de referência
 - Identidade é relevante
 - Igualdade usualmente não é relevante
- Tipos de valor
 - Igualdade é relevante
 - Identidade não é relevante
- Tipos de valor são aqueles em que dois objetos são considerados iguais se o conteúdo é igual, mesmo que tenham referências diferentes

Classes Java.

Tipos primitivos em Java.

Tipos primitivos vs. classes

- Variáveis de tipos primitivos (**int**, **double**, etc)
 - Guardam directamente o valor
 - Atribuição *altera o valor guardado*
- Variáveis de classes Java
 - Guardam uma **referência** para um objecto
 - Atribuição *altera a referência e não o objecto referenciado*

Referências e objectos: declaração e construção

Construção da referência

Classe *variável*;

Declaração da referência *variável*, não inicializada, capaz de referenciar objectos da classe *Classe*.

variável = null;

Inicialização da referência com o valor especial null, que indica que referência não referencia qualquer objecto.

Construção da referência

Classe *outraVariável* = null;

Construção de uma referência com valor inicial nulo.

Classe *aindaOutraVariável* =
 new *Classe*(...);

Construção de um novo objecto e de uma referência que o referencia.

Construção do novo objecto

Atribuição: valor vs. referência

Tipos primitivos (int, boolean, etc)

```
int a = 7;  
int b = a;  
int c;
```

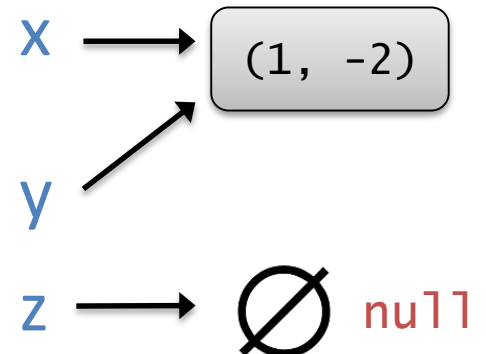
a 7
b 7
c 0

Tipos de referência (Classes e vectores)

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

```
Point y = x;
```

```
Point z;
```




Igualdade: valor vs. referência

Tipos primitivos (int, boolean, etc)

```
int a = 7;
```

```
int b = 7;
```

a 

a == b
 true

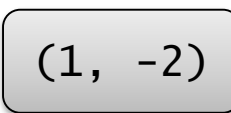
b 

Tipos de referência (Classes e vectores)

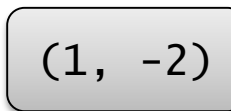
```
Point x = new Point(1, -2);
```

```
Point y = x;
```

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

x → 

y →

z → 

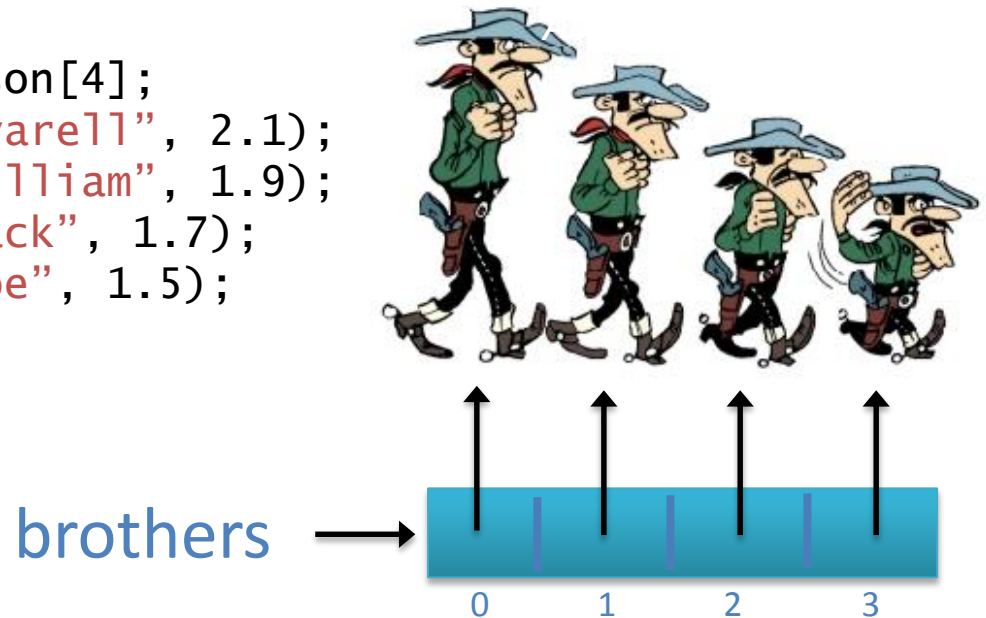
x == y
 true

x == z
 false

Vectores de objectos

- Uma vez definida uma classe, é possível criar vectores de objectos dessa classe



```
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);
```



Vectores de objetos

```
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 → 

Objectos com vectores

```
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;  
    }  
    . . .  
}
```

Objectos com vectores

```
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
```

Objectos com vectores

```
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;  
    }  
}
```

Objectos com vectores

```
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

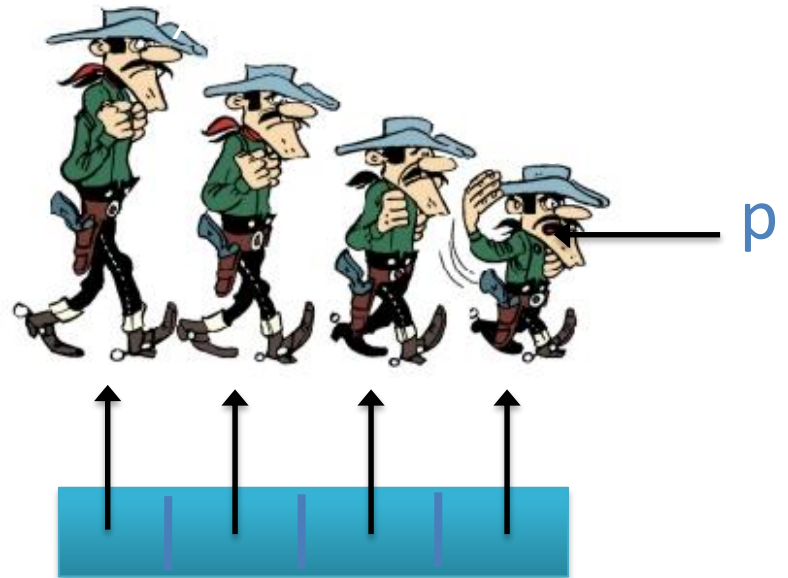
Objectos com vectores

```
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;  
    }  
}
```

Objectos com vectores

```
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();
```

brothers



Referências

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

Sumário

- Referências
- Vectores de Objectos