Aula 15 – Strings, Entrada do Teclado, Busca e Ordenação

Norton Trevisan Roman

10 de maio de 2013

A Classe Projeto

```
class Projeto {
    Residencia[] condominio;
    int ultimo = -1; // último alocado

    boolean adicionaRes(Residencia r) {
        if (this.ultimo < this.condominio.length-1) {
            ultimo++;
            this.condominio[ultimo] = r;
            return(true);
        }
        return(false);
    }

    Projeto(int tam) {
        condominio = new Residencia[tam];
    }

    public static void main(String[] args) {
        Projeto proj = new Projeto(3);

        AreaCasa c = new AreaCasa(10,5);
        AreaPiscina p = new AreaPiscina(5);
        Residencia r = new Residencia(c, p);
        proj.adicionaRes(r);

        c = new AreaCasa(12,7);
        p = new AreaPiscina(6);
        r = new Residencia(c, p);
        proj.adicionaRes(r);

        c = new AreaCasa(10,6);
        p = new AreaPiscina(3.5);
        r = new Residencia(c, p);
        proj.adicionaRes(r);

        System.out.println(
              proj.condominio[1].casa.area());
    }
}
```

A Classe Projeto

```
class Projeto {
    Residencia[] condominio;
    int ultimo = -1; // último alocado

    boolean adicionaRes(Residencia r) {
        if (this.ultimo < this.condominio.length-1) {
            ultimo++;
            this.condominio[ultimo] = r;
            return(true);
        }
        return(false);
    }

    Projeto(int tam) {
        condominio = new Residencia[tam];
    }

    public static void main(String[] args) {
        Projeto proj = new Projeto(3);

        AreaCasa c = new AreaCasa(10,5);
        AreaPiscina p = new AreaPiscina(5);
        Residencia r = new Residencia(c, p);
        proj.adicionaRes(r);

        c = new AreaCasa(12,7);
        p = new AreaPiscina(6);
        r = new Residencia(c, p);
        proj.adicionaRes(r);

        c = new AreaCasa(10,6);
        p = new AreaPiscina(3.5);
        r = new Residencia(c, p);
        proj.adicionaRes(r);

        System.out.println(
              proj.condominio[1].casa.area());
    }
}
```

- Podemos acessar um único elemento do arranjo

A Classe Projeto

```
class Projeto {
    Residencia[] condominio;
    int ultimo = -1; // último alocado

    boolean adicionaRes(Residencia r) {
        if (this.ultimo < this.condominio.length-1) {
            ultimo++;
            this.condominio[ultimo] = r;
            return(true);
        }
        return(false);
    }

    Projeto(int tam) {
        condominio = new Residencia[tam];
    }

    public static void main(String[] args) {
        Projeto proj = new Projeto(3);

        AreaCasa c = new AreaCasa(10,5);
        AreaPiscina p = new AreaPiscina(5);
        Residencia r = new Residencia(c, p);
        proj.adicionaRes(r);

        c = new AreaCasa(12,7);
        p = new AreaPiscina(6);
        r = new Residencia(c, p);
        proj.adicionaRes(r);

        c = new AreaCasa(10,6);
        p = new AreaPiscina(3.5);
        r = new Residencia(c, p);
        proj.adicionaRes(r);

        System.out.println(
              proj.condominio[1].casa.area());
    }
}
```

- Podemos acessar um único elemento do arranjo
  - Método *area*

A Classe Projeto

```
class Projeto {
    Residencia[] condominio;
    int ultimo = -1; // último alocado

    boolean adicionaRes(Residencia r) {
        if (this.ultimo < this.condominio.length-1) {
            ultimo++;
            this.condominio[ultimo] = r;
            return(true);
        }
        return(false);
    }

    Projeto(int tam) {
        condominio = new Residencia[tam];
    }

    public static void main(String[] args) {
        Projeto proj = new Projeto(3);

        AreaCasa c = new AreaCasa(10,5);
        AreaPiscina p = new AreaPiscina(5);
        Residencia r = new Residencia(c, p);
        proj.adicionaRes(r);

        c = new AreaCasa(12,7);
        p = new AreaPiscina(6);
        r = new Residencia(c, p);
        proj.adicionaRes(r);

        c = new AreaCasa(10,6);
        p = new AreaPiscina(3.5);
        r = new Residencia(c, p);
        proj.adicionaRes(r);

        System.out.println(
              proj.condominio[1].casa.area());
    }
}
```

- Podemos acessar um único elemento do arranjo
  - Método *area* do objeto *casa*

A Classe Projeto

```
class Projeto {
    Residencia[] condominio;
    int ultimo = -1; // último alocado

    boolean adicionaRes(Residencia r) {
        if (this.ultimo < this.condominio.length-1) {
            ultimo++;
            this.condominio[ultimo] = r;
            return(true);
        }
        return(false);
    }

    Projeto(int tam) {
        condominio = new Residencia[tam];
    }

    public static void main(String[] args) {
        Projeto proj = new Projeto(3);

        AreaCasa c = new AreaCasa(10,5);
        AreaPiscina p = new AreaPiscina(5);
        Residencia r = new Residencia(c, p);
        proj.adicionaRes(r);

        c = new AreaCasa(12,7);
        p = new AreaPiscina(6);
        r = new Residencia(c, p);
        proj.adicionaRes(r);

        c = new AreaCasa(10,6);
        p = new AreaPiscina(3.5);
        r = new Residencia(c, p);
        proj.adicionaRes(r);

        System.out.println(
              proj.condominio[1].casa.area());
    }
}
```

- Podemos acessar um único elemento do arranjo
  - Método *area* do objeto *casa* do segundo objeto de *condominio* (classe *Residencia*)

A Classe Projeto

```
class Projeto {
    Residencia[] condominio;
    int ultimo = -1; // último alocado

    boolean adicionaRes(Residencia r) {
        if (this.ultimo < this.condominio.length-1) {
            ultimo++;
            this.condominio[ultimo] = r;
            return(true);
        }
        return(false);
    }

    Projeto(int tam) {
        condominio = new Residencia[tam];
    }

    public static void main(String[] args) {
        Projeto proj = new Projeto(3);

        AreaCasa c = new AreaCasa(10,5);
        AreaPiscina p = new AreaPiscina(5);
        Residencia r = new Residencia(c, p);
        proj.adicionaRes(r);

        c = new AreaCasa(12,7);
        p = new AreaPiscina(6);
        r = new Residencia(c, p);
        proj.adicionaRes(r);

        c = new AreaCasa(10,6);
        p = new AreaPiscina(3.5);
        r = new Residencia(c, p);
        proj.adicionaRes(r);

        System.out.println(
              proj.condominio[1].casa.area());
    }
}
```

- Podemos acessar um único elemento do arranjo
  - Método *area* do objeto *casa* do segundo objeto de *condominio* (classe *Residencia*) do objeto *proj*

Revendo Strings

- Até agora, como havíamos definido strings?

Revendo Strings

- Até agora, como havíamos definido strings?
  - Arranjos de caracteres

Revendo Strings

- Até agora, como havíamos definido strings?
  - Arranjos de caracteres

```
public static void main(String[] args) {
    char[] str = {'o','b','a'};
    char[] str2 = new char[2];

    str2[0] = 'e';
    str2[1] = 'i';
}
```

Revendo Strings

- Até agora, como havíamos definido strings?
  - Arranjos de caracteres
- Embora funcione, seu uso é bastante limitado em Java

```
public static void main(String[] args) {
    char[] str = {'o','b','a'};
    char[] str2 = new char[2];

    str2[0] = 'e';
    str2[1] = 'i';
}
```

Revendo Strings

- Até agora, como havíamos definido strings?
  - Arranjos de caracteres
- Embora funcione, seu uso é bastante limitado em Java
- Felizmente, há uma classe Java definida apenas para o uso de strings:

```
public static void main(String[] args) {
    char[] str = {'o','b','a'};
    char[] str2 = new char[2];

    str2[0] = 'e';
    str2[1] = 'i';
}
```

Revendo Strings

- Até agora, como havíamos definido strings?
  - Arranjos de caracteres
- Embora funcione, seu uso é bastante limitado em Java
- Felizmente, há uma classe Java definida apenas para o uso de strings:

```
public static void main(String[] args) {
    char[] str = {'o','b','a'};
    char[] str2 = new char[2];

    str2[0] = 'e';
    str2[1] = 'i';
}
```

```
public static void main(String[] args) {
    String s = "Meu string";

    System.out.println(s);
}
```

Revendo Strings

- Até agora, como havíamos definido strings?
  - Arranjos de caracteres
- Embora funcione, seu uso é bastante limitado em Java
- Felizmente, há uma classe Java definida apenas para o uso de strings:
- Podemos então reescrever os nomes dos materiais da piscina

```
public static void main(String[] args) {
    char[] str = {'o','b','a'};
    char[] str2 = new char[2];

    str2[0] = 'e';
    str2[1] = 'i';
}
```

```
public static void main(String[] args) {
    String s = "Meu string";

    System.out.println(s);
}
```

Revendo Strings

- Até agora, como havíamos definido strings?
  - Arranjos de caracteres
- Embora funcione, seu uso é bastante limitado em Java
- Felizmente, há uma classe Java definida apenas para o uso de strings:
- Podemos então reescrever os nomes dos materiais da piscina
  - De...

```
public static void main(String[] args) {
    char[] str = {'o','b','a'};
    char[] str2 = new char[2];

    str2[0] = 'e';
    str2[1] = 'i';
}
```

```
public static void main(String[] args) {
    String s = "Meu string";

    System.out.println(s);
}
```

```
static char[][] nomes =
          {{'A','l','v','e','n','a','r','i','a'},
          {'V','i','n','i','l'},
          {'F','i','b','r','a'},
          {'P','l','á','s','t','i','c','o'}};
```

Revendo Strings

- Até agora, como havíamos definido strings?
  - Arranjos de caracteres
- Embora funcione, seu uso é bastante limitado em Java
- Felizmente, há uma classe Java definida apenas para o uso de strings:
- Podemos então reescrever os nomes dos materiais da piscina
  - De...
  - Para...

```
public static void main(String[] args) {
    char[] str = {'o','b','a'};
    char[] str2 = new char[2];

    str2[0] = 'e';
    str2[1] = 'i';
}
```

```
public static void main(String[] args) {
    String s = "Meu string";

    System.out.println(s);
}
```

```
static String[] nomes = {"Alvenaria",
                         "Vinil",
                         "Fibra",
                        "Plástico"};
```

Strings

- Como fazemos para acessarmos o caracter em uma determinada posição no string?

Strings

- Como fazemos para acessarmos o caracter em uma determinada posição no string?

```
public static void main(String[] args) {
    String s = "Meu string";

    System.out.println(s[4]);
}
```

Strings

- Como fazemos para acessarmos o caracter em uma determinada posição no string?

```
public static void main(String[] args) {
    String s = "Meu string";

    System.out.println(s[4]);
}
```

```
AreaPiscina.java:102: array required, but java.lang.String found
        System.out.println(s[4]);
                           ^
1 error
```

Strings

- Como fazemos para acessarmos o caracter em uma determinada posição no string?

```
public static void main(String[] args) {
    String s = "Meu string";

    System.out.println(s[4]);
}
```

```
AreaPiscina.java:102: array required, but java.lang.String found
        System.out.println(s[4]);
                           ^
1 error
```

  - Strings não são arranjos!

Strings

- Como fazemos para acessarmos o caracter em uma determinada posição no string?

```
public static void main(String[] args) {
    String s = "Meu string";

    System.out.println(s.charAt(4));
}
```

```
AreaPiscina.java:102: array required, but java.lang.String found
        System.out.println(s[4]);
                           ^
1 error
```

  - Strings não são arranjos! Deve-se usar o método *charAt()*

Strings

- Como fazemos para acessarmos o caracter em uma determinada posição no string?

```
public static void main(String[] args) {
    String s = "Meu string";

    System.out.println(s.charAt(4));
}
```

```
AreaPiscina.java:102: array required, but java.lang.String found
        System.out.println(s[4]);
                           ^
1 error
```

  - Strings não são arranjos! Deve-se usar o método *charAt()*

- E como podemos modificar um caracter no String?

Strings

- Como fazemos para acessarmos o caracter em uma determinada posição no string?

```
public static void main(String[] args) {
    String s = "Meu string";

    System.out.println(s.charAt(4));
}
```

```
AreaPiscina.java:102: array required, but java.lang.String found
        System.out.println(s[4]);
                           ^
1 error
```

  - Strings não são arranjos! Deve-se usar o método *charAt()*

- E como podemos modificar um caracter no String?
  - Não podemos. Objetos *String* são constantes

Strings

- Como fazemos para acessarmos o caracter em uma determinada posição no string?

```
public static void main(String[] args) {
    String s = "Meu string";

    System.out.println(s.charAt(4));
}
```

```
AreaPiscina.java:102: array required, but java.lang.String found
        System.out.println(s[4]);
                           ^
1 error
```

  - Strings não são arranjos! Deve-se usar o método *charAt()*

- E como podemos modificar um caracter no String?
  - Não podemos. Objetos *String* são constantes
  - Não podem ser mudados após terem sido criados

Strings

- Como fazemos para acessarmos o caracter em uma determinada posição no string?

```
public static void main(String[] args) {
    String s = "Meu string";

    System.out.println(s.charAt(4));
}
```

```
AreaPiscina.java:102: array required, but java.lang.String found
        System.out.println(s[4]);
                           ^
1 error
```

  - Strings não são arranjos! Deve-se usar o método *charAt()*

- E como podemos <u>modificar</u> um caracter no String?
  - Não podemos. Objetos *String* são constantes
  - Não podem ser mudados após terem sido criados

```
public static void main(String[] args) {
    String s = "Meu string";
    char[] s2 = s.toCharArray();
    s2[4] = 'b';
    s = String.valueOf(s2);
    //ou s = new String(s2);

    System.out.println(s);
}
```

- Solução: transformar em arranjo...

Strings

- Como fazemos para acessarmos o caracter em uma determinada posição no string?

```
public static void main(String[] args) {
    String s = "Meu string";

    System.out.println(s.charAt(4));
}
```

```
AreaPiscina.java:102: array required, but java.lang.String found
        System.out.println(s[4]);
                           ^
1 error
```

  - Strings não são arranjos! Deve-se usar o método *charAt()*

- E como podemos modificar um caracter no String?
  - Não podemos. Objetos *String* são constantes
  - Não podem ser mudados após terem sido criados

```
public static void main(String[] args) {
    String s = "Meu string";
    char[] s2 = s.toCharArray();
    s2[4] = 'b';
    s = String.valueOf(s2);
    //ou s = new String(s2);

    System.out.println(s);
}
```

- Solução: transformar em arranjo...
  - Sempre criará um novo objeto

Olhando mais de perto...

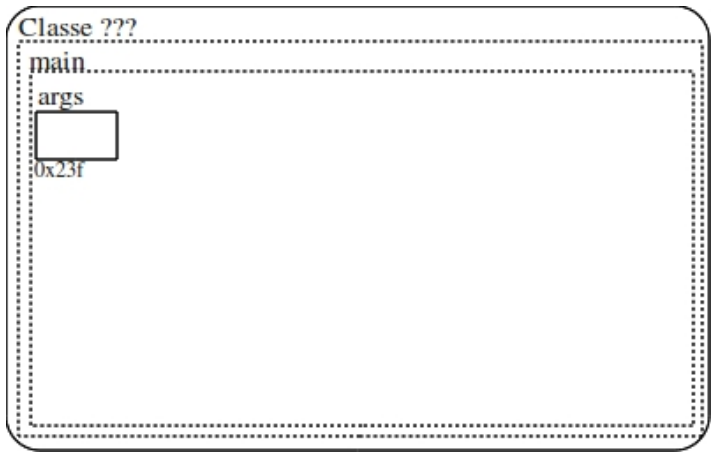
- O que houve com “Meu string”?

```
public static void main(String[] args) {
    String s = "Meu string";
    char[] s2 = s.toCharArray();
    s2[4] = 'b';
    s = String.valueOf(s2);
    //ou s = new String(s2);

    System.out.println(s);
}
```

Olhando mais de perto...

- O que houve com “Meu string”?
- Vejamos o comportamento da memória:

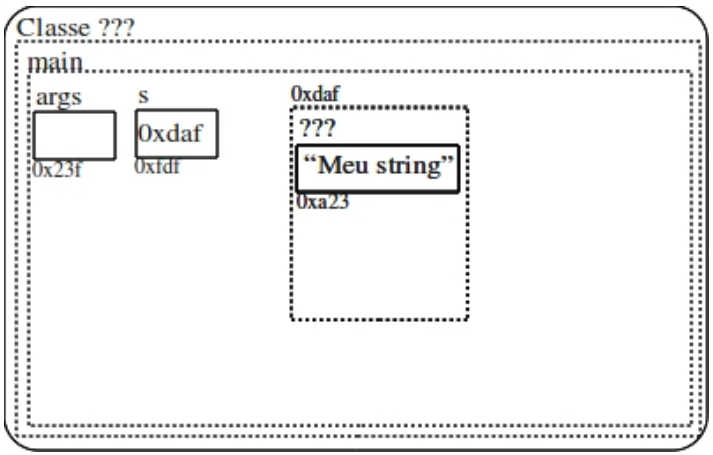
```
public static void main(String[] args) {
    String s = "Meu string";
    char[] s2 = s.toCharArray();
    s2[4] = 'b';
    s = String.valueOf(s2);
    //ou s = new String(s2);

    System.out.println(s);
}
```



Olhando mais de perto...

- O que houve com “Meu string”?
- Vejamos o comportamento da memória:

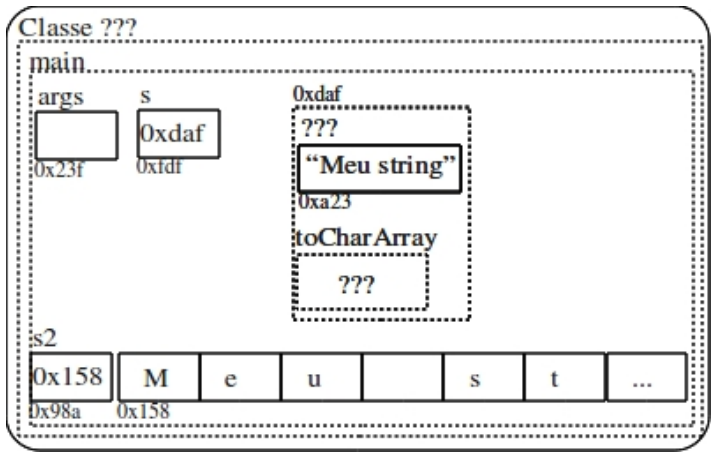
```
public static void main(String[] args) {
    String s = "Meu string";
    char[] s2 = s.toCharArray();
    s2[4] = 'b';
    s = String.valueOf(s2);
    //ou s = new String(s2);

    System.out.println(s);
}
```



Olhando mais de perto...

- O que houve com “Meu string”?
- Vejamos o comportamento da memória:

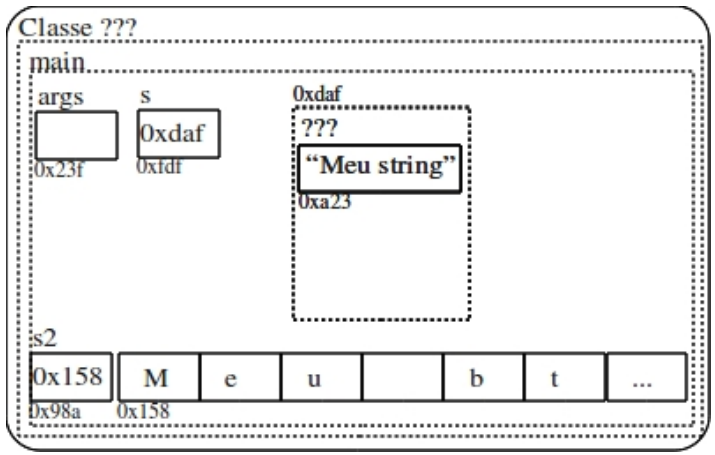
```
public static void main(String[] args) {
    String s = "Meu string";
    char[] s2 = s.toCharArray();
    s2[4] = 'b';
    s = String.valueOf(s2);
    //ou s = new String(s2);

    System.out.println(s);
}
```



Olhando mais de perto...

- O que houve com “Meu string”?
- Vejamos o comportamento da memória:

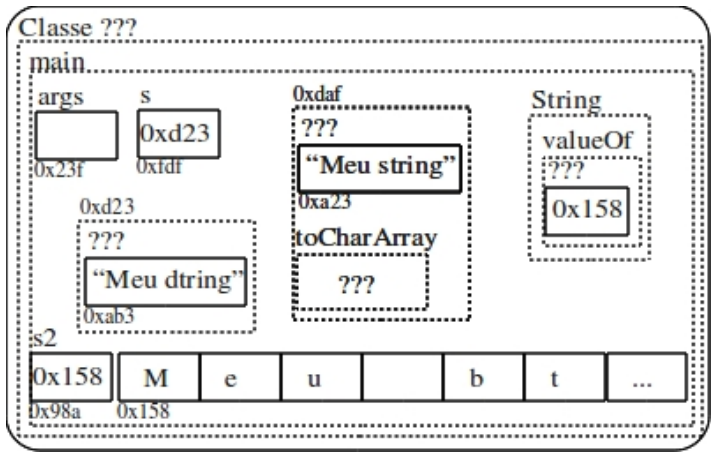
```
public static void main(String[] args) {
    String s = "Meu string";
    char[] s2 = s.toCharArray();
    s2[4] = 'b';
    s = String.valueOf(s2);
    //ou s = new String(s2);

    System.out.println(s);
}
```



Olhando mais de perto...

- O que houve com “Meu string”?
- Vejamos o comportamento da memória:

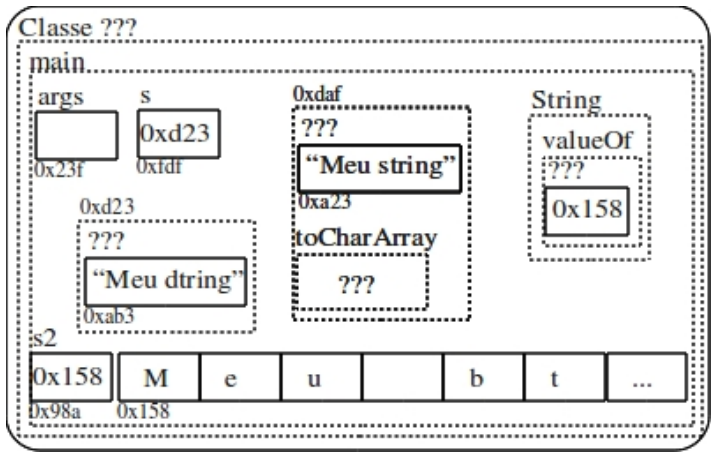
```
public static void main(String[] args) {
    String s = "Meu string";
    char[] s2 = s.toCharArray();
    s2[4] = 'b';
    s = String.valueOf(s2);
    //ou s = new String(s2);

    System.out.println(s);
}
```



Olhando mais de perto...

- O que houve com “Meu string”?
- Vejamos o comportamento da memória:
  - O primeiro objeto String teve sua referência perdida

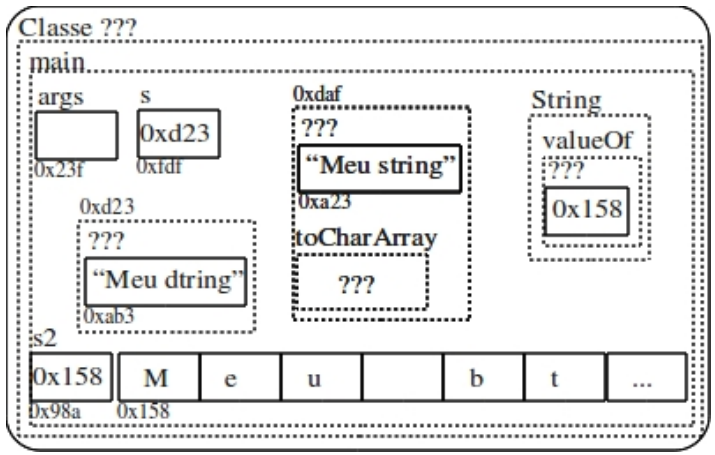
```
public static void main(String[] args) {
    String s = "Meu string";
    char[] s2 = s.toCharArray();
    s2[4] = 'b';
    s = String.valueOf(s2);
    //ou s = new String(s2);

    System.out.println(s);
}
```



Olhando mais de perto...

- O que houve com “Meu string”?
- Vejamos o comportamento da memória:
  - O primeiro objeto String teve sua referência perdida
  - Será desalocado pelo garbage collector

```
public static void main(String[] args) {
    String s = "Meu string";
    char[] s2 = s.toCharArray();
    s2[4] = 'b';
    s = String.valueOf(s2);
    //ou s = new String(s2);

    System.out.println(s);
}
```



Strings

- O que acontece se fizermos *str1 == str2*?

Strings

- O que acontece se fizermos *str1 == str2*?
  - Serão comparadas as referências (endereços)

Strings

- O que acontece se fizermos *str1 == str2*?
  - Serão comparadas as referências (endereços)
  - Será *true* somente se ambos *str1* e *str2* contiverem o mesmo endereço na memória

Strings

- O que acontece se fizermos *str1 == str2*?
  - Serão comparadas as referências (endereços)
  - Será *true* somente se ambos *str1* e *str2* contiverem o mesmo endereço na memória

- E se fizermos...

```
public static void main(String[] args)
{
    String s = "Meu string";
    s = s + "oba";
    s = s + 'c';
    s = s + 4;
    s = s + 23.5;
}
```

Strings

- O que acontece se fizermos *str1 == str2*?
  - Serão comparadas as referências (endereços)
  - Será *true* somente se ambos *str1* e *str2* contiverem o mesmo endereço na memória

- E se fizermos...
  - s = "Meu string"

```
public static void main(String[] args)
{
    String s = "Meu string";
    s = s + "oba";
    s = s + 'c';
    s = s + 4;
    s = s + 23.5;
}
```

Strings

- O que acontece se fizermos *str1 == str2*?
  - Serão comparadas as referências (endereços)
  - Será *true* somente se ambos *str1* e *str2* contiverem o mesmo endereço na memória

- E se fizermos...
  - s = "Meu string"
  - s = "Meu stringoba"

```
public static void main(String[] args)
{
    String s = "Meu string";
    s = s + "oba";
    s = s + 'c';
    s = s + 4;
    s = s + 23.5;
}
```

Strings

- O que acontece se fizermos *str1 == str2*?
  - Serão comparadas as referências (endereços)
  - Será *true* somente se ambos *str1* e *str2* contiverem o mesmo endereço na memória

- E se fizermos...
  - s = "Meu string"
  - s = "Meu stringoba"
  - s = "Meu stringobac"

```
public static void main(String[] args)
{
    String s = "Meu string";
    s = s + "oba";
    s = s + 'c';
    s = s + 4;
    s = s + 23.5;
}
```

Strings

- O que acontece se fizermos *str1 == str2*?
  - Serão comparadas as referências (endereços)
  - Será *true* somente se ambos *str1* e *str2* contiverem o mesmo endereço na memória

- E se fizermos...
  - s = "Meu string"
  - s = "Meu stringoba"
  - s = "Meu stringobac"
  - s = "Meu stringobac4"

```
public static void main(String[] args)
{
    String s = "Meu string";
    s = s + "oba";
    s = s + 'c';
    s = s + 4;
    s = s + 23.5;
}
```

Strings

- O que acontece se fizermos *str1 == str2*?
  - Serão comparadas as referências (endereços)
  - Será *true* somente se ambos *str1* e *str2* contiverem o mesmo endereço na memória

- E se fizermos...
  - s = "Meu string"
  - s = "Meu stringoba"
  - s = "Meu stringobac"
  - s = "Meu stringobac4"
  - s = "Meu stringobac423.5"

```
public static void main(String[] args)
{
    String s = "Meu string";
    s = s + "oba";
    s = s + 'c';
    s = s + 4;
    s = s + 23.5;
}
```

Strings

- O que acontece se fizermos *str1 == str2*?
  - Serão comparadas as referências (endereços)
  - Será *true* somente se ambos *str1* e *str2* contiverem o mesmo endereço na memória

- E se fizermos...
  - s = "Meu string"
  - s = "Meu stringoba"
  - s = "Meu stringobac"
  - s = "Meu stringobac4"
  - s = "Meu stringobac423.5"

```
public static void main(String[] args)
{
    String s = "Meu string";
    s = s + "oba";
    s = s + 'c';
    s = s + 4;
    s = s + 23.5;
}
```

- Como strings são imutáveis, a cada mudança criamos nova cópia na memória

Strings

- O que acontece se fizermos *str1 == str2*?
  - Serão comparadas as referências (endereços)
  - Será *true* somente se ambos *str1* e *str2* contiverem o mesmo endereço na memória

- E se fizermos...
  - s = "Meu string"
  - s = "Meu stringoba"
  - s = "Meu stringobac"
  - s = "Meu stringobac4"
  - s = "Meu stringobac423.5"

```
public static void main(String[] args)
{
    String s = "Meu string";
    s = s + "oba";
    s = s + 'c';
    s = s + 4;
    s = s + 23.5;
}
```

- Como strings são imutáveis, a cada mudança criamos nova cópia na memória
  - Sobrescrevendo o endereço que havia em *s*

Strings

- O que acontece se fizermos *str1 == str2*?
  - Serão comparadas as referências (endereços)
  - Será *true* somente se ambos *str1* e *str2* contiverem o mesmo endereço na memória

- E se fizermos...
  - s = "Meu string"
  - s = "Meu stringoba"
  - s = "Meu stringobac"
  - s = "Meu stringobac4"
  - s = "Meu stringobac423.5"

```
public static void main(String[] args)
{
    String s = "Meu string";
    s = s + "oba";
    s = s + 'c';
    s = s + 4;
    s = s + 23.5;
}
```

- Como strings são imutáveis, a cada mudança criamos nova cópia na memória
  - Sobrescrevendo o endereço que havia em *s*
  - E perdendo assim a referência à cópia antiga

Strings

- Como fazer para comparar dois strings?

Strings

- Como fazer para comparar dois strings?
  - método *equals*

```
public static void main(String[] args) {
    String s = "Meu string";
    String s2 = "Meu string";

    System.out.println(s.equals(s2));

}
```

Strings

- Como fazer para comparar dois strings?
  - método *equals*
- E para saber o tamanho de um string?

```
public static void main(String[] args) {
    String s = "Meu string";
    String s2 = "Meu string";

    System.out.println(s.equals(s2));

}
```

Strings

- Como fazer para comparar dois strings?
  - método *equals*
- E para saber o tamanho de um string?
  - método *length*

```
public static void main(String[] args) {
    String s = "Meu string";
    String s2 = "Meu string";

    System.out.println(s.equals(s2));
    System.out.println(s.length());
}
```

Strings

- Como fazer para comparar dois strings?
  - método *equals*
- E para saber o tamanho de um string?
  - método *length*

```
public static void main(String[] args) {
    String s = "Meu string";
    String s2 = "Meu string";

    System.out.println(s.equals(s2));
    System.out.println(s.length());
}
```

- http://java.sun.com/javase/6/docs/api/java/lang/String.html

Strings

- Como fazer para comparar dois strings?
  - método *equals*
- E para saber o tamanho de um string?
  - método *length*

```
public static void main(String[] args) {
    String s = "Meu string";
    String s2 = "Meu string";

    System.out.println(s.equals(s2));
    System.out.println(s.length());
}
```

- http://java.sun.com/javase/6/docs/api/java/lang/String.html
- O que será impresso?

```
public static void main(String[] args) {
    String s1 = "Bom dia";
    String s2 = s1;

    s1 = "Boa noite";
    System.out.println(s2);
}
```

Strings

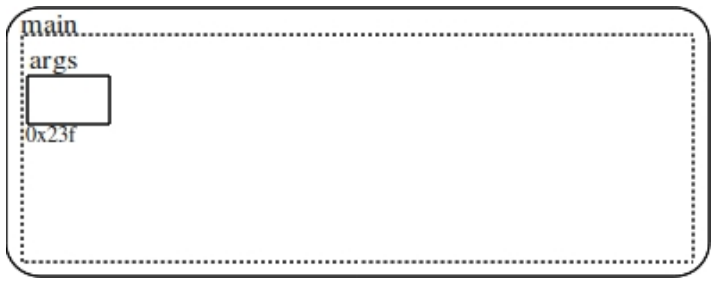
- Como fazer para comparar dois strings?
  - método *equals*
- E para saber o tamanho de um string?
  - método *length*

```
public static void main(String[] args) {
    String s = "Meu string";
    String s2 = "Meu string";

    System.out.println(s.equals(s2));
    System.out.println(s.length());
}
```

- http://java.sun.com/javase/6/docs/api/java/lang/String.html

```
public static void main(String[] args) {
    String s1 = "Bom dia";
    String s2 = s1;

    s1 = "Boa noite";
    System.out.println(s2);
}
```

- O que será impresso?



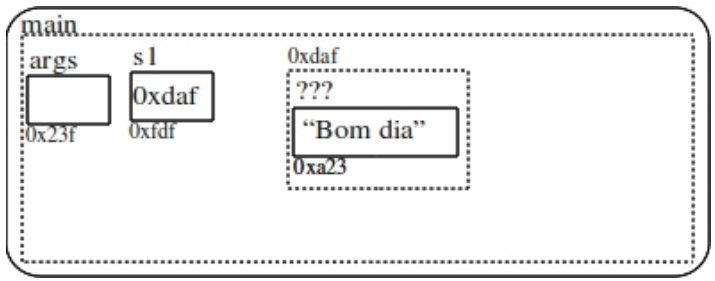
Strings

- Como fazer para comparar dois strings?
  - método *equals*
- E para saber o tamanho de um string?
  - método *length*

```
public static void main(String[] args) {
    String s = "Meu string";
    String s2 = "Meu string";

    System.out.println(s.equals(s2));
    System.out.println(s.length());
}
```

- http://java.sun.com/javase/6/docs/api/java/lang/String.html
- O que será impresso?

```
public static void main(String[] args) {
    String s1 = "Bom dia";
    String s2 = s1;

    s1 = "Boa noite";
    System.out.println(s2);
}
```



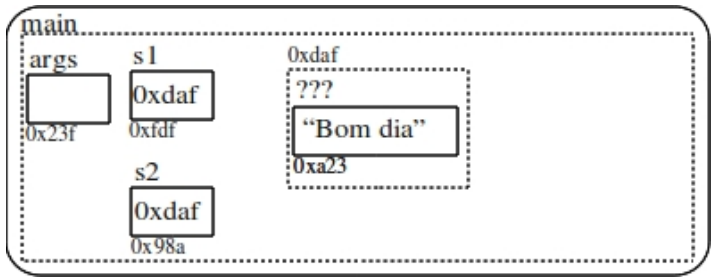
Strings

- Como fazer para comparar dois strings?
  - método *equals*
- E para saber o tamanho de um string?
  - método *length*

```
public static void main(String[] args) {
    String s = "Meu string";
    String s2 = "Meu string";

    System.out.println(s.equals(s2));
    System.out.println(s.length());
}
```

- http://java.sun.com/javase/6/docs/api/java/lang/String.html
- O que será impresso?

```
public static void main(String[] args) {
    String s1 = "Bom dia";
    String s2 = s1;

    s1 = "Boa noite";
    System.out.println(s2);
}
```



Strings

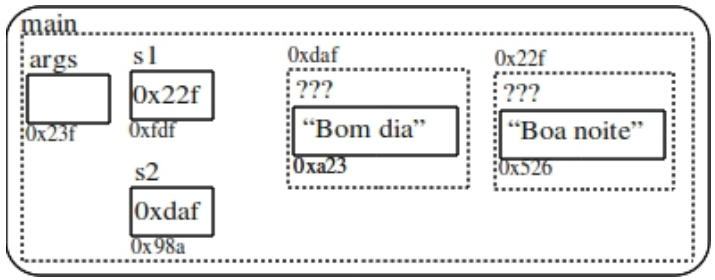
- Como fazer para comparar dois strings?
  - método *equals*
- E para saber o tamanho de um string?
  - método *length*

```
public static void main(String[] args) {
    String s = "Meu string";
    String s2 = "Meu string";

    System.out.println(s.equals(s2));
    System.out.println(s.length());
}
```

- http://java.sun.com/javase/6/docs/api/java/lang/String.html

- O que será impresso?

```
public static void main(String[] args) {
    String s1 = "Bom dia";
    String s2 = s1;

    s1 = "Boa noite";
    System.out.println(s2);
}
```



Strings

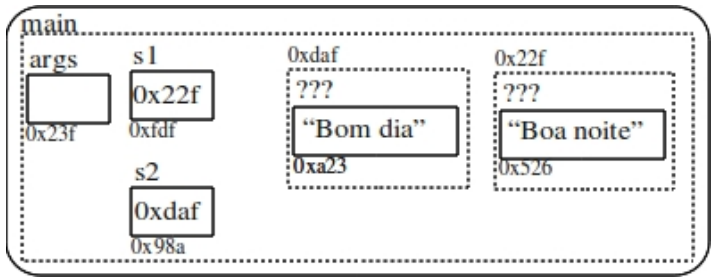
- Como fazer para comparar dois strings?
  - método *equals*
- E para saber o tamanho de um string?
  - método *length*

```
public static void main(String[] args) {
    String s = "Meu string";
    String s2 = "Meu string";

    System.out.println(s.equals(s2));
    System.out.println(s.length());
}
```

- http://java.sun.com/javase/6/docs/api/java/lang/String.html

- O que será impresso?
  - "Bom dia"

```
public static void main(String[] args) {
    String s1 = "Bom dia";
    String s2 = s1;

    s1 = "Boa noite";
    System.out.println(s2);
}
```



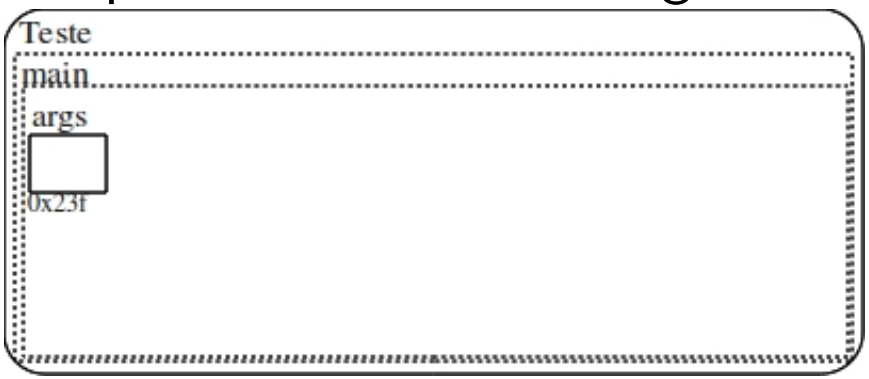
Strings

- E qual a saída desse código?

```
class Teste {
    void misterio(String y) {
        y = "Novo string";
    }

    public static void main(String[] args) {
        String x = "String antigo";
        Teste a = new Teste();

        a.misterio(x);

        System.out.println(x);
    }
}
```

Strings

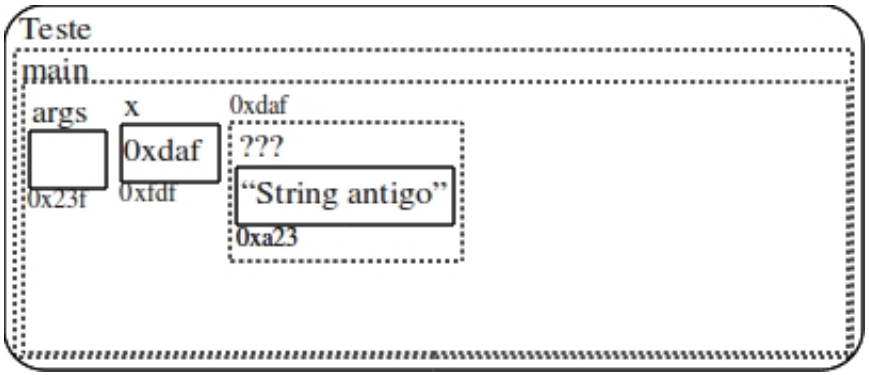
- E qual a saída desse código?




```
class Teste {
    void misterio(String y) {
        y = "Novo string";
    }

    public static void main(String[] args) {
        String x = "String antigo";
        Teste a = new Teste();

        a.misterio(x);

        System.out.println(x);
    }
}
```

Strings

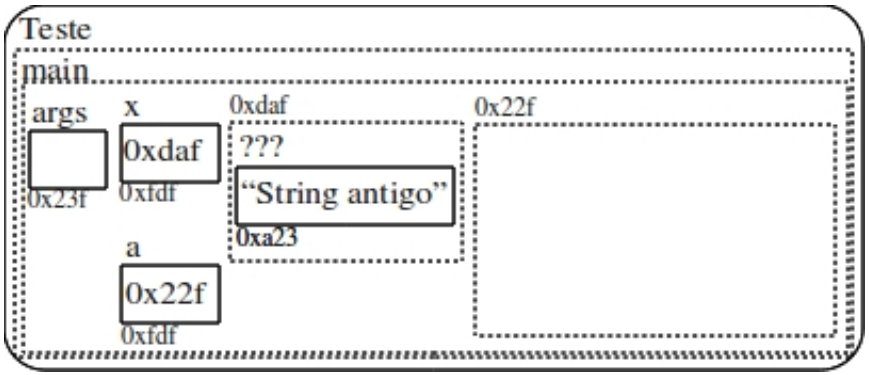
- E qual a saída desse código?




```
class Teste {
    void misterio(String y) {
        y = "Novo string";
    }

    public static void main(String[] args) {
        String x = "String antigo";
        Teste a = new Teste();

        a.misterio(x);

        System.out.println(x);
    }
}
```

Strings

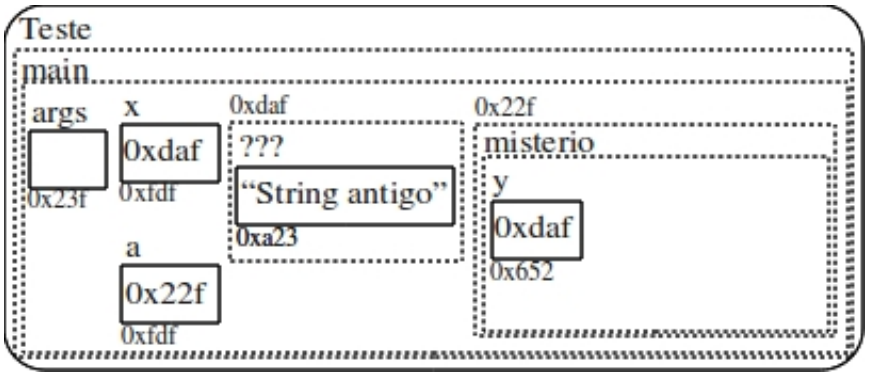
- E qual a saída desse código?




```
class Teste {
    void misterio(String y) {
        y = "Novo string";
    }

    public static void main(String[] args) {
        String x = "String antigo";
        Teste a = new Teste();

        a.misterio(x);

        System.out.println(x);
    }
}
```

Strings

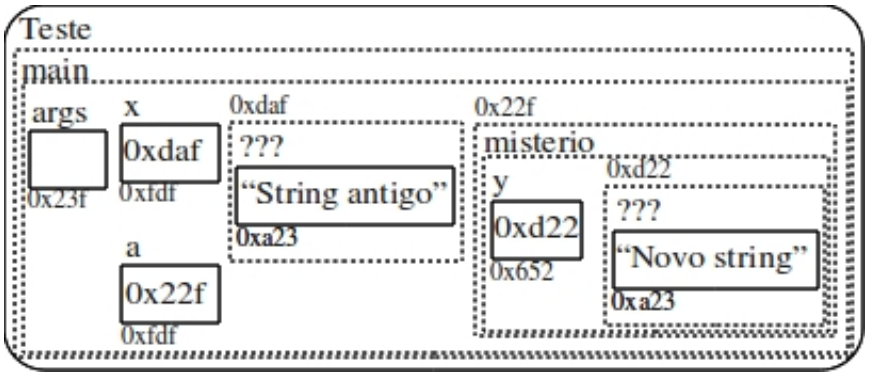
- E qual a saída desse código?




```
class Teste {
    void misterio(String y) {
        y = "Novo string";
    }

    public static void main(String[] args) {
        String x = "String antigo";
        Teste a = new Teste();

        a.misterio(x);

        System.out.println(x);
    }
}
```

Strings

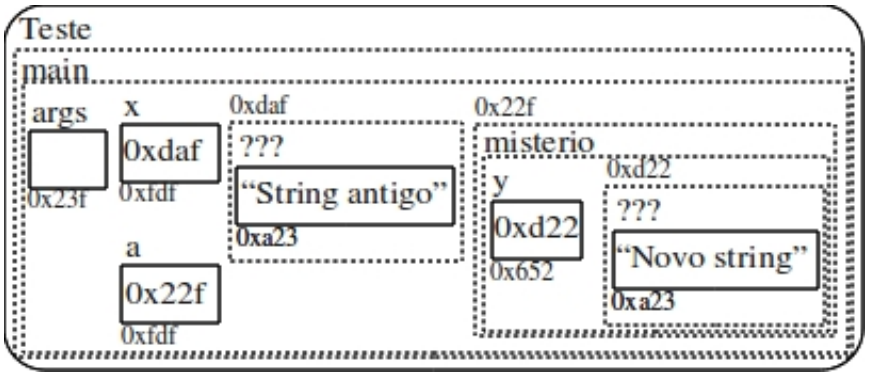
- E qual a saída desse código?




```
class Teste {
    void misterio(String y) {
        y = "Novo string";
    }

    public static void main(String[] args) {
        String x = "String antigo";
        Teste a = new Teste();

        a.misterio(x);

        System.out.println(x);
    }
}
```

Strings

- E qual a saída desse código?

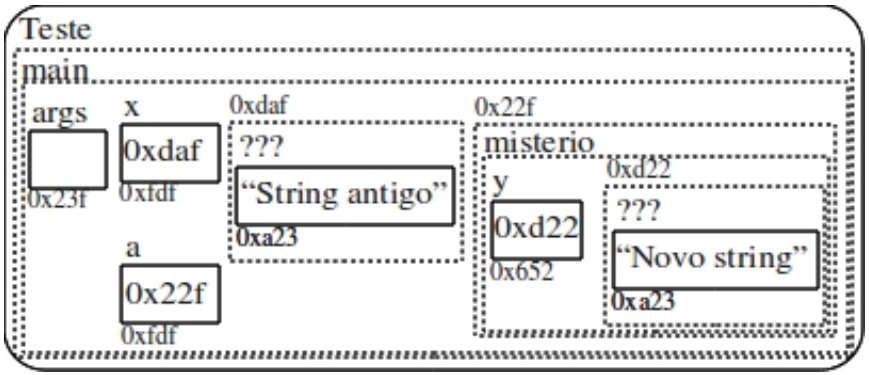



```
class Teste {
    void misterio(String y) {
        y = "Novo string";
    }

    public static void main(String[] args) {
        String x = "String antigo";
        Teste a = new Teste();

        a.misterio(x);

        System.out.println(x);
    }
}
```

- "String antigo"

Strings

- E qual a saída desse código?




```
class Teste {
    void misterio(String y) {
        y = "Novo string";
    }

    public static void main(String[] args) {
        String x = "String antigo";
        Teste a = new Teste();

        a.misterio(x);

        System.out.println(x);
    }
}
```

- "String antigo"
- Modificamos o parâmetro, e não a variável do objeto

Parâmetros da Linha de Comando

- Considere o método main. O que são esses args?

```
public static void main(String[] args) {


}
```

Parâmetros da Linha de Comando

- Considere o método main. O que são esses args?
  - São um arranjo de Strings

```
public static void main(String[] args) {

}
```

Parâmetros da Linha de Comando

- Considere o método main. O que são esses args?
  - São um arranjo de Strings
- E o que o código no *for* faz?

```
public static void main(String[] args) {
    for (String arg : args)
                        System.out.println(arg);
}
```

Parâmetros da Linha de Comando

- Considere o método main. O que são esses args?
  - São um arranjo de Strings
- E o que o código no *for* faz?
  - Vamos ver...

```
public static void main(String[] args) {
    for (String arg : args)
                        System.out.println(arg);
}
```

Parâmetros da Linha de Comando

- Considere o método main. O que são esses args?
  - São um arranjo de Strings
- E o que o código no *for* faz?
  - Vamos ver...

```
public static void main(String[] args) {
    for (String arg : args)
                        System.out.println(arg);
}
```

```
$java AreaPiscina
```

Parâmetros da Linha de Comando

- Considere o método main. O que são esses args?
  - São um arranjo de Strings
- E o que o código no *for* faz?
  - Vamos ver...

```
public static void main(String[] args) {
    for (String arg : args)
                        System.out.println(arg);
}
```

```
$java AreaPiscina

$ java AreaPiscina oba
oba
```

Parâmetros da Linha de Comando

- Considere o método main. O que são esses args?
  - São um arranjo de Strings
- E o que o código no *for* faz?
  - Vamos ver...

```
public static void main(String[] args) {
    for (String arg : args)
                        System.out.println(arg);
}
```

```
$java AreaPiscina

$ java AreaPiscina oba
oba

$ java AreaPiscina oba 23
oba
23
```

Parâmetros da Linha de Comando

- Considere o método main. O que são esses args?
  - São um arranjo de Strings
- E o que o código no *for* faz?
  - Vamos ver...

```
public static void main(String[] args) {
    for (String arg : args)
                        System.out.println(arg);
}
```

```
$java AreaPiscina

$ java AreaPiscina oba
oba

$ java AreaPiscina oba 23
oba
23

$ java AreaPiscina oba 23 eba
oba
23
eba
```

Parâmetros da Linha de Comando

- Considere o método main. O que são esses args?
  - São um arranjo de Strings
- E o que o código no *for* faz?
  - Vamos ver...
  - Escreve os argumentos passados na linha de comando

```
public static void main(String[] args) {
    for (String arg : args)
                        System.out.println(arg);
}
```

```
$java AreaPiscina

$ java AreaPiscina oba
oba

$ java AreaPiscina oba 23
oba
23

$ java AreaPiscina oba 23 eba
oba
23
eba
```

Parâmetros da Linha de Comando

- Considere o método main. O que são esses args?
  - São um arranjo de Strings
- E o que o código no *for* faz?
  - Vamos ver...
  - Escreve os argumentos passados na linha de comando
  - Informações adicionais separadas por espaço

```
public static void main(String[] args) {
    for (String arg : args)
                        System.out.println(arg);
}
```

```
$java AreaPiscina

$ java AreaPiscina oba
oba

$ java AreaPiscina oba 23
oba
23

$ java AreaPiscina oba 23 eba
oba
23
eba
```

Parâmetros da Linha de Comando

- Considere o método main. O que são esses args?
  - São um arranjo de Strings
- E o que o código no *for* faz?
  - Vamos ver...
  - Escreve os argumentos passados na linha de comando
  - Informações adicionais separadas por espaço
  - Usadas para passar alguma informação inicial ao programa

```
public static void main(String[] args) {
    for (String arg : args)
                        System.out.println(arg);
}
```

```
$java AreaPiscina

$ java AreaPiscina oba
oba

$ java AreaPiscina oba 23
oba
23

$ java AreaPiscina oba 23 eba
oba
23
eba
```

Parâmetros da Linha de Comando

- Considere o método main. O que são esses args?
  - São um arranjo de Strings
- E o que o código no *for* faz?
  - Vamos ver...
  - Escreve os argumentos passados na linha de comando
  - Informações adicionais separadas por espaço
  - Usadas para passar alguma informação inicial ao programa

```
public static void main(String[] args) {
    for (String arg : args)
                        System.out.println(arg);
}
```

```
$java AreaPiscina

$ java AreaPiscina oba
oba

$ java AreaPiscina oba 23
oba
23

$ java AreaPiscina oba 23 eba
oba
23
eba
```

- Então *args* nos dá um arranjo com cada argumento passado ao programa, na ordem em que é passado

Parâmetros da Linha de Comando

- Poderíamos, por exemplo, usar a linha de comando para definir o material de uma piscina

Parâmetros da Linha de Comando

- Poderíamos, por exemplo, usar a linha de comando para definir o material de uma piscina

```
class AreaPiscina {
    ...
    AreaPiscina() {
        double[] aux = {1500, 1100, 750, 500};
        this.precos = aux;
        this.raio = 10;
    }
    ...
    static int tipoMat(String nome) {
        for (int i=0; i<nomes.length; i++) {
            if (nomes[i].equals(nome)) return(i);
        }
        return(-1);
    }

    public static void main(String[] args) {
        if (args.length != 1) System.out.println(
                    "Número de parâmetros inválido");
        else {
            AreaPiscina p = new AreaPiscina();
            int material =
                        AreaPiscina.tipoMat(args[0]);

            if (material != -1)
                System.out.println(p.valor(p.area(),
                                           material));
            else
                System.out.println("Material
                                            inválido");
        }
    }
}
```

Parâmetros da Linha de Comando

- Poderíamos, por exemplo, usar a linha de comando para definir o material de uma piscina

Linha de Comando

```
$ java AreaPiscina
Número de parâmetros inválido

$ java AreaPiscina Plástic
Material inválido

$ java AreaPiscina Plástico
157079.63267948967

$ java AreaPiscina Alvenaria
471238.89803846896
```

```
class AreaPiscina {
    ...
    AreaPiscina() {
        double[] aux = {1500, 1100, 750, 500};
        this.precos = aux;
        this.raio = 10;
    }
    ...
    static int tipoMat(String nome) {
        for (int i=0; i<nomes.length; i++) {
            if (nomes[i].equals(nome)) return(i);
        }
        return(-1);
    }

    public static void main(String[] args) {
        if (args.length != 1) System.out.println(
                    "Número de parâmetros inválido");
        else {
            AreaPiscina p = new AreaPiscina();
            int material =
                        AreaPiscina.tipoMat(args[0]);

            if (material != -1)
                System.out.println(p.valor(p.area(),
                                        material));
            else
                System.out.println("Material
                                        inválido");
        }
    }
}
```

Parâmetros da Linha de Comando

- Ou poderíamos usá-la para definir o material e o raio de uma piscina

Parâmetros da Linha de Comando

- Ou poderíamos usá-la para definir o material e o raio de uma piscina

```
class AreaPiscina {
    ...
    AreaPiscina(double raio) {
        this();
        this.raio = raio;
    }
    ...
    public static void main(String[] args) {
        if (args.length != 2) System.out.println(
                    "Número de parâmetros inválido");
        else {
            int material =
                        AreaPiscina.tipoMat(args[0]);
            AreaPiscina p = new AreaPiscina(
                        Double.parseDouble(args[1]));

            if (material != -1)
                System.out.println(p.valor(p.area(),
                                           material));
            else
                System.out.println("Material
                                           inválido");
        }
    }
}
```

Parâmetros da Linha de Comando

- Ou poderíamos usá-la para definir o material e o raio de uma piscina
  - Como a linha de comando tem só strings, temos que converter

```
class AreaPiscina {
    ...
    AreaPiscina(double raio) {
        this();
        this.raio = raio;
    }
    ...
    public static void main(String[] args) {
        if (args.length != 2) System.out.println(
                    "Número de parâmetros inválido");
        else {
            int material =
                        AreaPiscina.tipoMat(args[0]);
            AreaPiscina p = new AreaPiscina(
                        Double.parseDouble(args[1]));

            if (material != -1)
                System.out.println(p.valor(p.area(),
                                           material));
            else
                System.out.println("Material
                                           inválido");
        }
    }
}
```

Parâmetros da Linha de Comando

- Ou poderíamos usá-la para definir o material e o raio de uma piscina
  - Como a linha de comando tem só strings, temos que converter
  - Para isso usamos métodos da classe *Double*

```
class AreaPiscina {
    ...
    AreaPiscina(double raio) {
        this();
        this.raio = raio;
    }
    ...
    public static void main(String[] args) {
        if (args.length != 2) System.out.println(
                    "Número de parâmetros inválido");
        else {
            int material =
                        AreaPiscina.tipoMat(args[0]);
            AreaPiscina p = new AreaPiscina(
                        Double.parseDouble(args[1]));

            if (material != -1)
                System.out.println(p.valor(p.area(),
                                           material));
            else
                System.out.println("Material
                                           inválido");
        }
    }
}
```

Parâmetros da Linha de Comando

- Ou poderíamos usá-la para definir o material e o raio de uma piscina
  - Como a linha de comando tem só strings, temos que converter
  - Para isso usamos métodos da classe *Double*

Linha de Comando

```
$ java AreaPiscina Alvenaria 10
471238.89803846896

$ java AreaPiscina Alvenaria 20
1884955.5921538759
```

```
class AreaPiscina {
    ...
    AreaPiscina(double raio) {
        this();
        this.raio = raio;
    }
    ...
    public static void main(String[] args) {
        if (args.length != 2) System.out.println(
                    "Número de parâmetros inválido");
        else {
            int material =
                        AreaPiscina.tipoMat(args[0]);
            AreaPiscina p = new AreaPiscina(
                        Double.parseDouble(args[1]));

            if (material != -1)
                System.out.println(p.valor(p.area(),
                                           material));
            else
                System.out.println("Material
                                           inválido");
        }
    }
}
```

Parâmetros da Linha de Comando

- Todo tipo primitivo tem sua classe equivalente

Parâmetros da Linha de Comando

- Todo tipo primitivo tem sua classe equivalente
  - Todas, à exceção de *Character*, com o método "parseTipo(String s)"

Parâmetros da Linha de Comando

- Todo tipo primitivo tem sua classe equivalente
  - Todas, à exceção de *Character*, com o método "parseTipo(String s)"

Primitivo	*Classe*	*parse*
int	Integer	parseInt(String s)
long	Long	parseLong(String s)
float	Float	parseFloat(String s)
double	Double	parseDouble(String s)
boolean	Boolean	parseBoolean(String s)
char	Character	–

Parâmetros da Linha de Comando

- Todo tipo primitivo tem sua classe equivalente
  - Todas, à exceção de *Character*, com o método "parseTipo(String s)"

Primitivo	*Classe*	*parse*
int	Integer	parseInt(String s)
long	Long	parseLong(String s)
float	Float	parseFloat(String s)
double	Double	parseDouble(String s)
boolean	Boolean	parseBoolean(String s)
char	Character	–

- Documentação: http://download.oracle.com/javase/6/docs/api/overview-summary.html

Entrada

- Entrada via linha de comando, embora interessante, é limitada

Entrada

- Entrada via linha de comando, embora interessante, é limitada
  - Reduz as possibilidades de interação com o usuário

Entrada

- Entrada via linha de comando, embora interessante, é limitada
  - Reduz as possibilidades de interação com o usuário
- O ideal seria trocar mensagens

Entrada

- Entrada via linha de comando, embora interessante, é limitada
  - Reduz as possibilidades de interação com o usuário
- O ideal seria trocar mensagens
  - Escrevendo coisas na tela (*println()*)

Entrada

- Entrada via linha de comando, embora interessante, é limitada
  - Reduz as possibilidades de interação com o usuário
- O ideal seria trocar mensagens
  - Escrevendo coisas na tela (*println()*)
  - Lendo coisas do teclado (como?)

Entrada

- Entrada via linha de comando, embora interessante, é limitada
  - Reduz as possibilidades de interação com o usuário
- O ideal seria trocar mensagens
  - Escrevendo coisas na tela (*println()*)
  - Lendo coisas do teclado (como?)
- Via *Scanner* (Java ≥ 5)

Entrada

- Entrada via linha de comando, embora interessante, é limitada
  - Reduz as possibilidades de interação com o usuário
- O ideal seria trocar mensagens
  - Escrevendo coisas na tela (*println()*)
  - Lendo coisas do teclado (como?)
- Via *Scanner* (Java ≥ 5)

```
import java.util.Scanner;

class Projeto {
    /* Representação do condomínio */
    Residencia[] condominio;
    ...
    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        System.out.print("Quantas residências
                                há no condomínio? ");
        String sTotal = entrada.nextLine();

        int nRes = Integer.parseInt(sTotal);

        Projeto proj = new Projeto(nRes);

        for (int i=0; i<nRes; i++) {
            Residencia res = new Residencia(
                                    new AreaCasa(),
                                new AreaPiscina());
            proj.adicionaRes(res);
        }

        System.out.println(proj.condominio.length);
    }
}
```

Entrada – Olhando mais de perto

- Primeiro temos que dizer ao compilador onde *Scanner* está

```
import java.util.Scanner;

class Projeto {
    /* Representação do condomínio */
    Residencia[] condominio;
    ...
    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        System.out.print("Quantas residências
                                há no condomínio? ");
        String sTotal = entrada.nextLine();

        int nRes = Integer.parseInt(sTotal);

        Projeto proj = new Projeto(nRes);

        for (int i=0; i<nRes; i++) {
            Residencia res = new Residencia(
                                new AreaCasa(),
                                new AreaPiscina());
            proj.adicionaRes(res);
        }

        System.out.println(proj.condominio.length);
    }
}
```

Entrada – Olhando mais de perto

- Primeiro temos que dizer ao compilador onde *Scanner* está
- Criamos então o objeto *Scanner*, que deve ler da entrada padrão de nosso programa (teclado)

```
import java.util.Scanner;

class Projeto {
    /* Representação do condomínio */
    Residencia[] condominio;
    ...
    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        System.out.print("Quantas residências
                                há no condomínio? ");
        String sTotal = entrada.nextLine();

        int nRes = Integer.parseInt(sTotal);

        Projeto proj = new Projeto(nRes);

        for (int i=0; i<nRes; i++) {
            Residencia res = new Residencia(
                                new AreaCasa(),
                                new AreaPiscina());
            proj.adicionaRes(res);
        }

        System.out.println(proj.condominio.length);
    }
}
```

Entrada – Olhando mais de perto

- Primeiro temos que dizer ao compilador onde *Scanner* está
- Criamos então o objeto *Scanner*, que deve ler da entrada padrão de nosso programa (teclado)
- Fazemos a pergunta...

```
import java.util.Scanner;

class Projeto {
    /* Representação do condomínio */
    Residencia[] condominio;
    ...
    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        System.out.print("Quantas residências
                                há no condomínio? ");
        String sTotal = entrada.nextLine();

        int nRes = Integer.parseInt(sTotal);

        Projeto proj = new Projeto(nRes);

        for (int i=0; i<nRes; i++) {
            Residencia res = new Residencia(
                                new AreaCasa(),
                                new AreaPiscina());
            proj.adicionaRes(res);
        }

        System.out.println(proj.condominio.length);
    }
}
```

Entrada – Olhando mais de perto

- Primeiro temos que dizer ao compilador onde *Scanner* está
- Criamos então o objeto *Scanner*, que deve ler da entrada padrão de nosso programa (teclado)
- Fazemos a pergunta...
- *Scanner* lê o string digitado (até um *enter* ser pressionado)

```
import java.util.Scanner;

class Projeto {
    /* Representação do condomínio */
    Residencia[] condominio;
    ...
    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        System.out.print("Quantas residências
                                há no condomínio? ");
        String sTotal = entrada.nextLine();

        int nRes = Integer.parseInt(sTotal);

        Projeto proj = new Projeto(nRes);

        for (int i=0; i<nRes; i++) {
            Residencia res = new Residencia(
                                    new AreaCasa(),
                                    new AreaPiscina());
            proj.adicionaRes(res);
        }

        System.out.println(proj.condominio.length);
    }
}
```

Entrada – Olhando mais de perto

- Primeiro temos que dizer ao compilador onde *Scanner* está
- Criamos então o objeto *Scanner*, que deve ler da entrada padrão de nosso programa (teclado)
- Fazemos a pergunta...
- *Scanner* lê o string digitado (até um *enter* ser pressionado)
- Recuperamos o inteiro nesse string

```
import java.util.Scanner;

class Projeto {
    /* Representação do condomínio */
    Residencia[] condominio;
    ...
    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        System.out.print("Quantas residências
                                há no condomínio? ");
        String sTotal = entrada.nextLine();

        int nRes = Integer.parseInt(sTotal);

        Projeto proj = new Projeto(nRes);

        for (int i=0; i<nRes; i++) {
            Residencia res = new Residencia(
                                new AreaCasa(),
                                new AreaPiscina());
            proj.adicionaRes(res);
        }

        System.out.println(proj.condominio.length);
    }
}
```

Entrada – Olhando mais de perto

- Primeiro temos que dizer ao compilador onde *Scanner* está
- Criamos então o objeto *Scanner*, que deve ler da entrada padrão de nosso programa (teclado)
- Fazemos a pergunta...
- *Scanner* lê o string digitado (até um *enter* ser pressionado)
- Recuperamos o inteiro nesse string
- Usamos para inicializar o objeto da classe *Projeto*

```
import java.util.Scanner;

class Projeto {
    /* Representação do condomínio */
    Residencia[] condominio;
    ...
    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        System.out.print("Quantas residências
                                há no condomínio? ");
        String sTotal = entrada.nextLine();

        int nRes = Integer.parseInt(sTotal);

        Projeto proj = new Projeto(nRes);

        for (int i=0; i<nRes; i++) {
            Residencia res = new Residencia(
                                new AreaCasa(),
                                new AreaPiscina());
            proj.adicionaRes(res);
        }

        System.out.println(proj.condominio.length);
    }
}
```

Entrada – Olhando mais de perto

- Primeiro temos que dizer ao compilador onde *Scanner* está
- Criamos então o objeto *Scanner*, que deve ler da entrada padrão de nosso programa (teclado)
- Fazemos a pergunta...
- *Scanner* lê o string digitado (até um *enter* ser pressionado)
- Recuperamos o inteiro nesse string
- Usamos para inicializar o objeto da classe *Projeto*

```
import java.util.Scanner;

class Projeto {
    /* Representação do condomínio */
    Residencia[] condominio;
    ...
    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        System.out.print("Quantas residências
                                há no condomínio? ");
        String sTotal = entrada.nextLine();

        int nRes = Integer.parseInt(sTotal);

        Projeto proj = new Projeto(nRes);

        for (int i=0; i<nRes; i++) {
            Residencia res = new Residencia(
                                new AreaCasa(),
                                new AreaPiscina());
            proj.adicionaRes(res);
        }

        System.out.println(proj.condominio.length);
    }
}
```

- Criamos residências padrão no condomínio

Scanner

- E se o que for lido não for o esperado?

Scanner

- E se o que for lido não for o esperado?
  - Um não inteiro quando se queria inteiro, por exemplo

Scanner

- E se o que for lido não for o esperado?
  - Um não inteiro quando se queria inteiro, por exemplo
- Um erro em tempo de execução ocorre

Scanner

- E se o que for lido não for o esperado?
  - Um não inteiro quando se queria inteiro, por exemplo
- Um erro em tempo de execução ocorre
- Scanner pode também ser usado para varrer as informações em um string

Scanner

- E se o que for lido não for o esperado?
  - Um não inteiro quando se queria inteiro, por exemplo
- Um erro em tempo de execução ocorre
- Scanner pode também ser usado para varrer as informações em um string

```
import java.util.Scanner;

class Teste {
    public static void main(String[] args) {
        String meuString = "3 tokens 5,3 true";

        Scanner sc = new Scanner (meuString);

        int i = sc.nextInt();
        String str = sc.next();
        double d = sc.nextDouble();
        boolean b = sc.nextBoolean();

        System.out.println(i);
        System.out.println(str);
        System.out.println(d);
        System.out.println(b);
    }
}
```

Scanner

- E se o que for lido não for o esperado?
  - Um não inteiro quando se queria inteiro, por exemplo
- Um erro em tempo de execução ocorre
- Scanner pode também ser usado para varrer as informações em um string

Linha de Comando

```
$ java Teste
3
tokens
5.3
true
```

```
import java.util.Scanner;

class Teste {
    public static void main(String[] args) {
        String meuString = "3 tokens 5,3 true";

        Scanner sc = new Scanner (meuString);

        int i = sc.nextInt();
        String str = sc.next();
        double d = sc.nextDouble();
        boolean b = sc.nextBoolean();

        System.out.println(i);
        System.out.println(str);
        System.out.println(d);
        System.out.println(b);
    }
}
```

Scanner

- E se o que for lido não for o esperado?
  - Um não inteiro quando se queria inteiro, por exemplo
- Um erro em tempo de execução ocorre
- Scanner pode também ser usado para varrer as informações em um string

Linha de Comando

```
$ java Teste
3
tokens
5.3
true
```

- Cuidado com floats e doubles!

```
import java.util.Scanner;

class Teste {
    public static void main(String[] args) {
        String meuString = "3 tokens 5,3 true";

        Scanner sc = new Scanner (meuString);

        int i = sc.nextInt();
        String str = sc.next();
        double d = sc.nextDouble();
        boolean b = sc.nextBoolean();

        System.out.println(i);
        System.out.println(str);
        System.out.println(d);
        System.out.println(b);
    }
}
```

Scanner

- E se o que for lido não for o esperado?
  - Um não inteiro quando se queria inteiro, por exemplo
- Um erro em tempo de execução ocorre
- Scanner pode também ser usado para varrer as informações em um string

Linha de Comando

```
$ java Teste
3
tokens
5.3
true
```

- Cuidado com floats e doubles!
  - Dependerão do locale instalado no computador

```
import java.util.Scanner;

class Teste {
    public static void main(String[] args) {
        String meuString = "3 tokens 5,3 true";

        Scanner sc = new Scanner (meuString);

        int i = sc.nextInt();
        String str = sc.next();
        double d = sc.nextDouble();
        boolean b = sc.nextBoolean();

        System.out.println(i);
        System.out.println(str);
        System.out.println(d);
        System.out.println(b);
    }
}
```

Busca em Arranjo

- Suponha que temos um arranjo de inteiros

Busca em Arranjo

- Suponha que temos um arranjo de inteiros
- Como fazemos para verificar se um determinado número está lá?

Busca em Arranjo

- Suponha que temos um arranjo de inteiros
- Como fazemos para verificar se um determinado número está lá?
  - Varremos o arranjo, da esquerda para a direita

Busca em Arranjo

- Suponha que temos um arranjo de inteiros
- Como fazemos para verificar se um determinado número está lá?
  - Varremos o arranjo, da esquerda para a direita
  - Se acharmos o número, então ele está no arranjo

Busca em Arranjo

- Suponha que temos um arranjo de inteiros
- Como fazemos para verificar se um determinado número está lá?
  - Varremos o arranjo, da esquerda para a direita
  - Se acharmos o número, então ele está no arranjo
  - Se chegarmos ao final do arranjo e não acharmos, ele não está

Busca em Arranjo

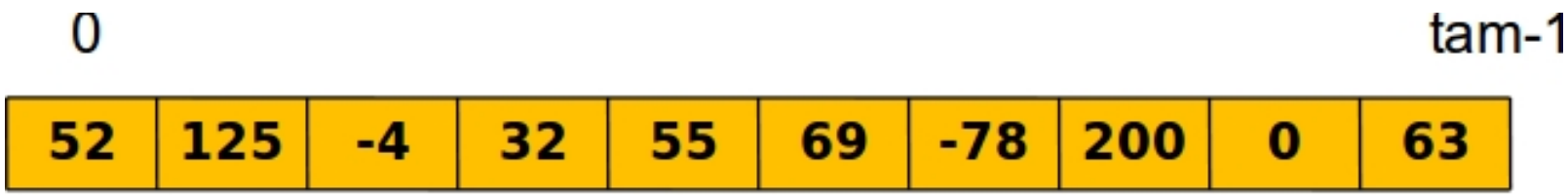
- Suponha que temos um arranjo de inteiros
- Como fazemos para verificar se um determinado número está lá?
  - Varremos o arranjo, da esquerda para a direita
  - Se acharmos o número, então ele está no arranjo
  - Se chegarmos ao final do arranjo e não acharmos, ele não está
- Busca Seqüencial!

Busca em Arranjo

- Suponha que temos um arranjo de inteiros
- Como fazemos para verificar se um determinado número está lá?
  - Varremos o arranjo, da esquerda para a direita
  - Se acharmos o número, então ele está no arranjo
  - Se chegarmos ao final do arranjo e não acharmos, ele não está
- Busca Seqüencial!
  - Ex: Buscando 32

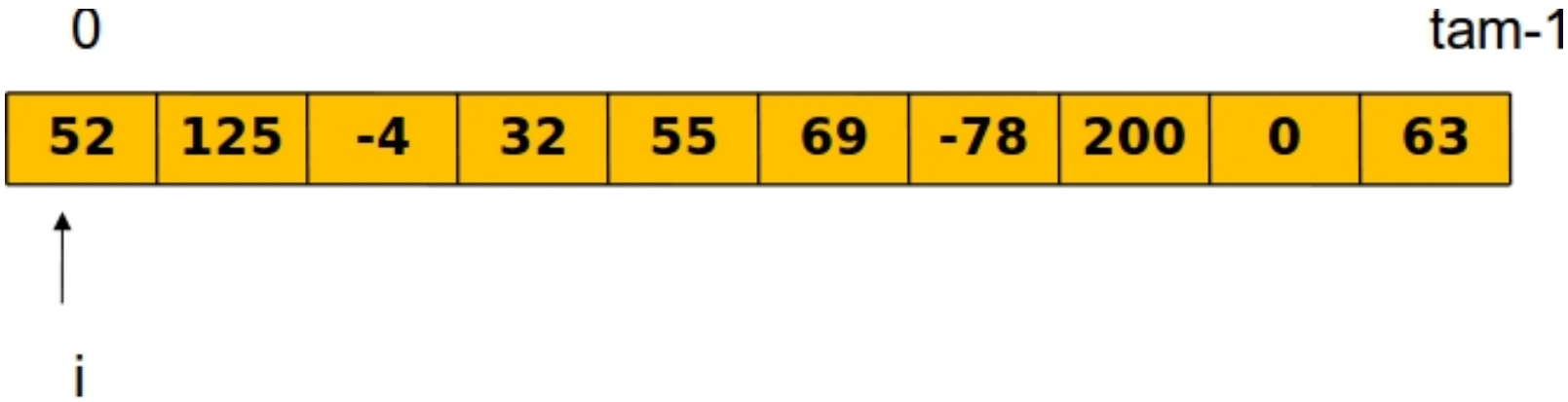
Busca em Arranjo

- Suponha que temos um arranjo de inteiros
- Como fazemos para verificar se um determinado número está lá?
  - Varremos o arranjo, da esquerda para a direita
  - Se acharmos o número, então ele está no arranjo
  - Se chegarmos ao final do arranjo e não acharmos, ele não está
- Busca Seqüencial!
  - Ex: Buscando 32



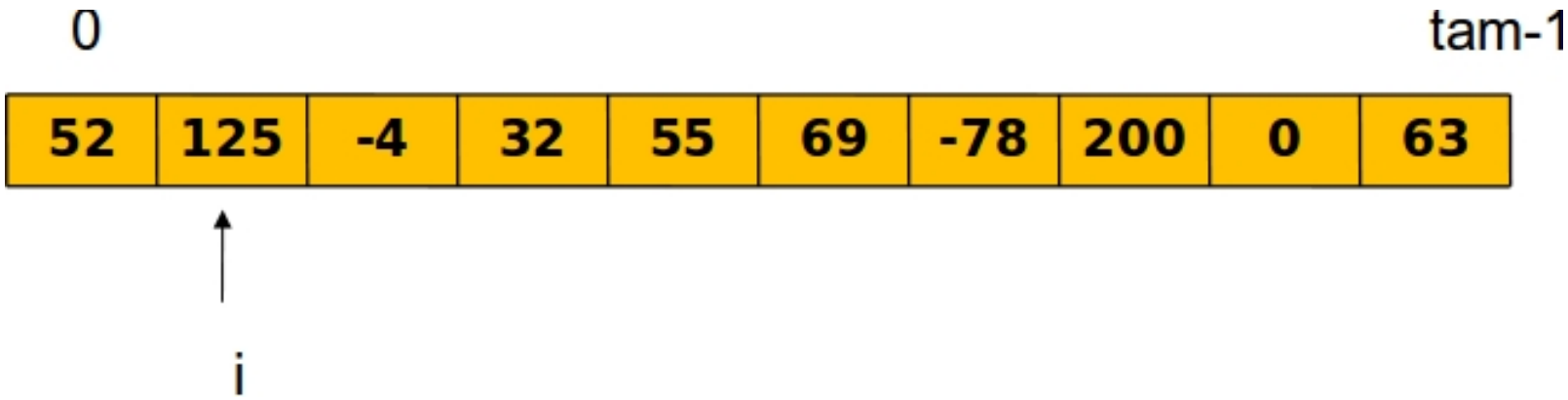
Busca em Arranjo

- Suponha que temos um arranjo de inteiros
- Como fazemos para verificar se um determinado número está lá?
  - Varremos o arranjo, da esquerda para a direita
  - Se acharmos o número, então ele está no arranjo
  - Se chegarmos ao final do arranjo e não acharmos, ele não está
- Busca Seqüencial!
  - Ex: Buscando 32



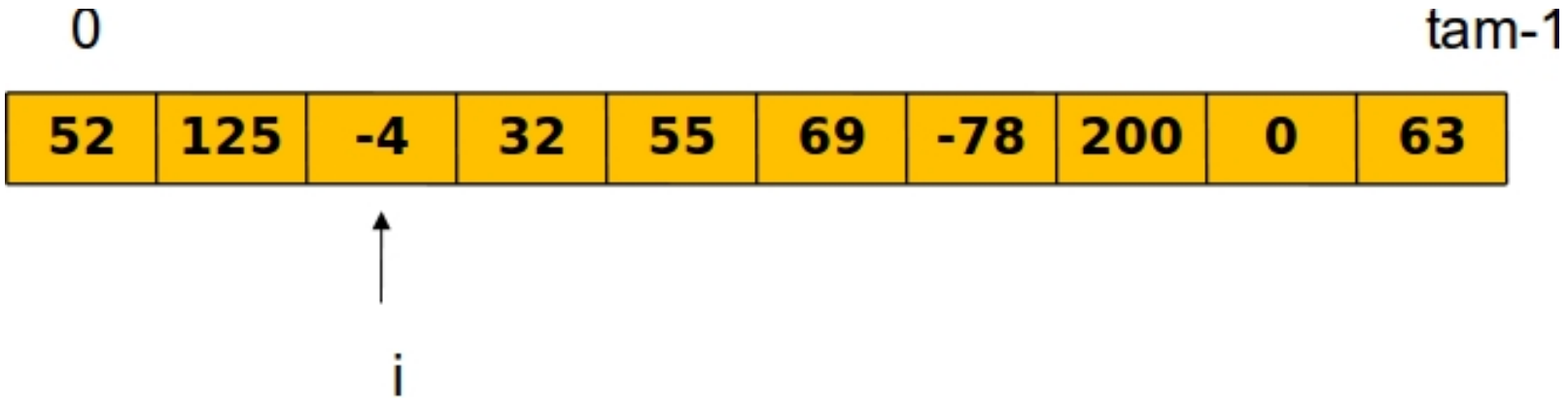
Busca em Arranjo

- Suponha que temos um arranjo de inteiros
- Como fazemos para verificar se um determinado número está lá?
  - Varremos o arranjo, da esquerda para a direita
  - Se acharmos o número, então ele está no arranjo
  - Se chegarmos ao final do arranjo e não acharmos, ele não está
- Busca Seqüencial!
  - Ex: Buscando 32



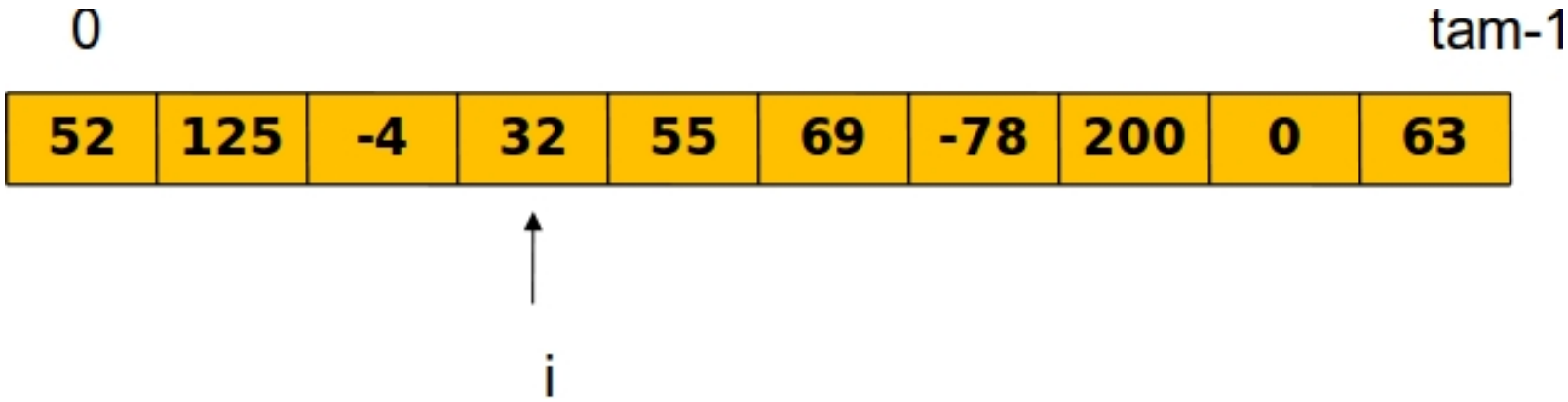
Busca em Arranjo

- Suponha que temos um arranjo de inteiros
- Como fazemos para verificar se um determinado número está lá?
  - Varremos o arranjo, da esquerda para a direita
  - Se acharmos o número, então ele está no arranjo
  - Se chegarmos ao final do arranjo e não acharmos, ele não está
- Busca Seqüencial!
  - Ex: Buscando 32



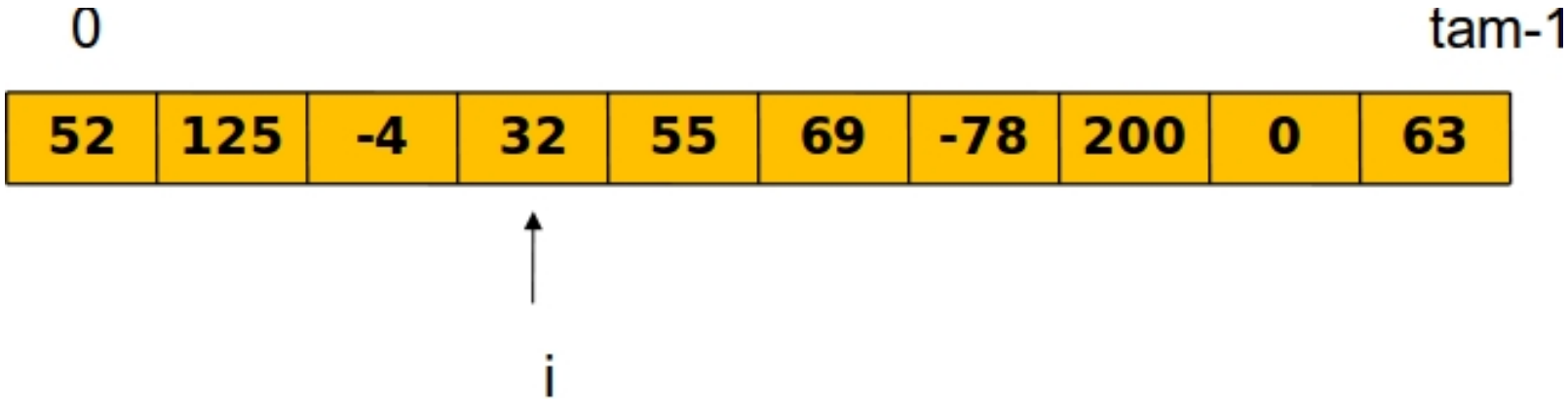
Busca em Arranjo

- Suponha que temos um arranjo de inteiros
- Como fazemos para verificar se um determinado número está lá?
  - Varremos o arranjo, da esquerda para a direita
  - Se acharmos o número, então ele está no arranjo
  - Se chegarmos ao final do arranjo e não acharmos, ele não está
- Busca Seqüencial!
  - Ex: Buscando 32



Busca em Arranjo

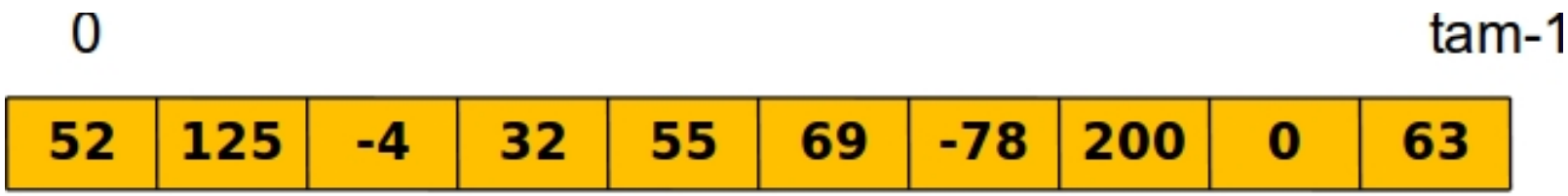
- Suponha que temos um arranjo de inteiros
- Como fazemos para verificar se um determinado número está lá?
  - Varremos o arranjo, da esquerda para a direita
  - Se acharmos o número, então ele está no arranjo
  - Se chegarmos ao final do arranjo e não acharmos, ele não está
- Busca Seqüencial!
  - Ex: Buscando 32



Encontrou! Posição 3.

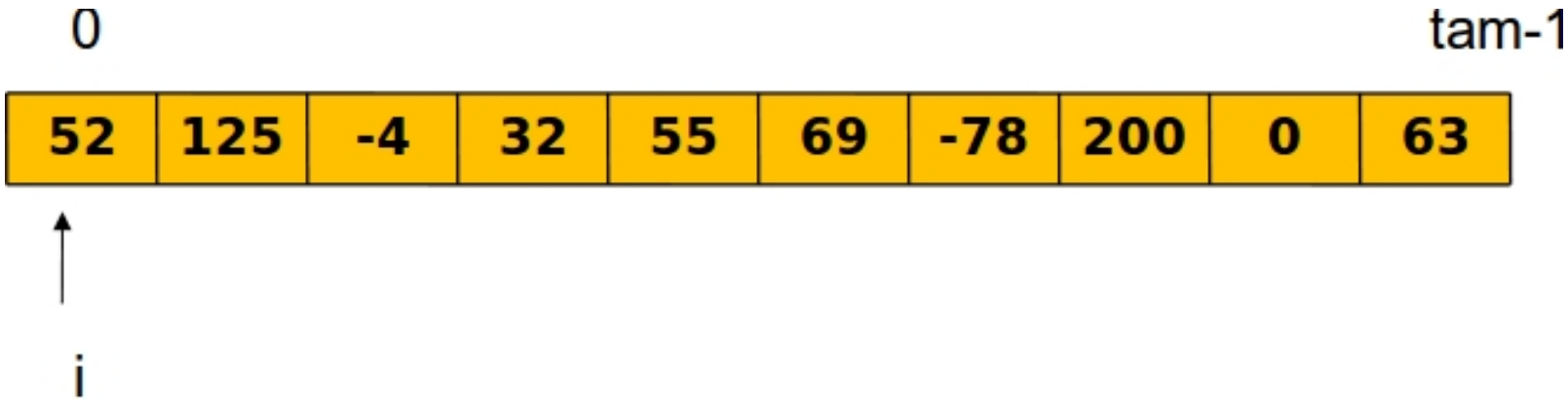
Busca em Arranjo

- Suponha que temos um arranjo de inteiros
- Como fazemos para verificar se um determinado número está lá?
  - Varremos o arranjo, da esquerda para a direita
  - Se acharmos o número, então ele está no arranjo
  - Se chegarmos ao final do arranjo e não acharmos, ele não está
- Busca Seqüencial!
  - Ex: Buscando -53



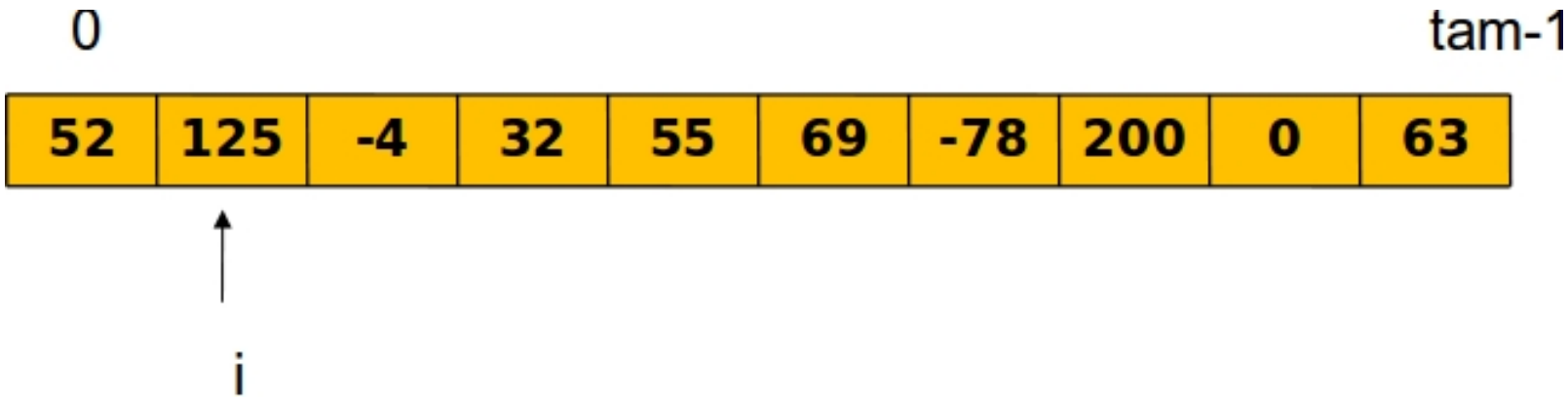
Busca em Arranjo

- Suponha que temos um arranjo de inteiros
- Como fazemos para verificar se um determinado número está lá?
  - Varremos o arranjo, da esquerda para a direita
  - Se acharmos o número, então ele está no arranjo
  - Se chegarmos ao final do arranjo e não acharmos, ele não está
- Busca Seqüencial!
  - Ex: Buscando -53



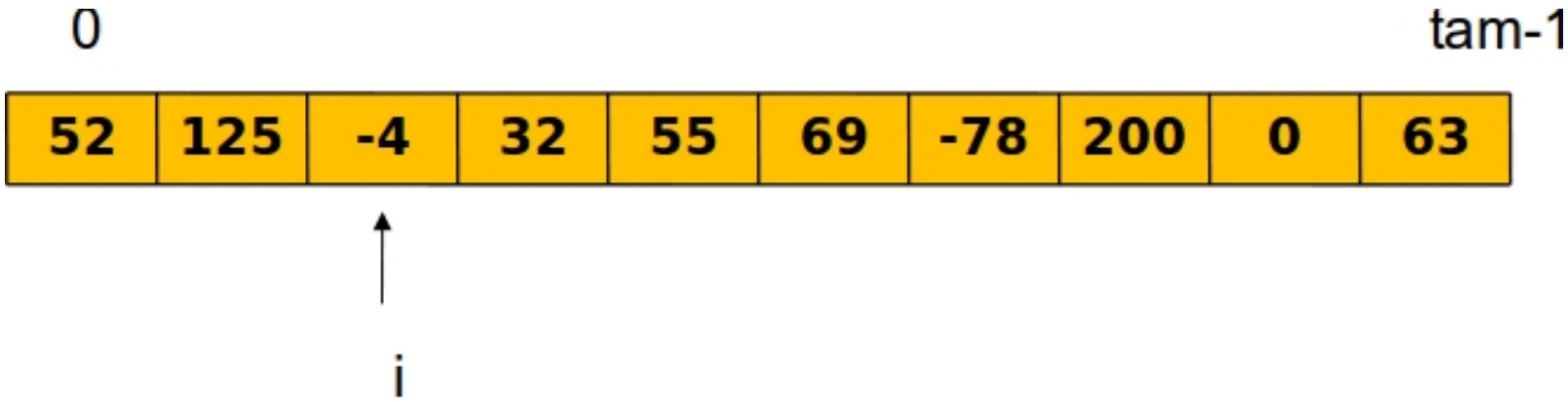
Busca em Arranjo

- Suponha que temos um arranjo de inteiros
- Como fazemos para verificar se um determinado número está lá?
  - Varremos o arranjo, da esquerda para a direita
  - Se acharmos o número, então ele está no arranjo
  - Se chegarmos ao final do arranjo e não acharmos, ele não está
- Busca Seqüencial!
  - Ex: Buscando -53



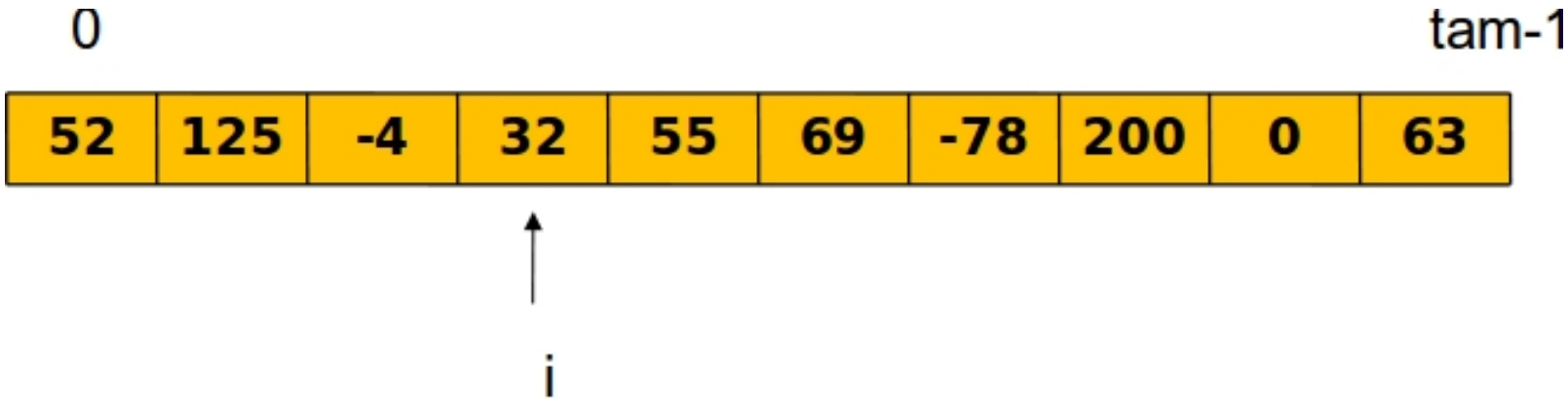
Busca em Arranjo

- Suponha que temos um arranjo de inteiros
- Como fazemos para verificar se um determinado número está lá?
  - Varremos o arranjo, da esquerda para a direita
  - Se acharmos o número, então ele está no arranjo
  - Se chegarmos ao final do arranjo e não acharmos, ele não está
- Busca Seqüencial!
  - Ex: Buscando -53



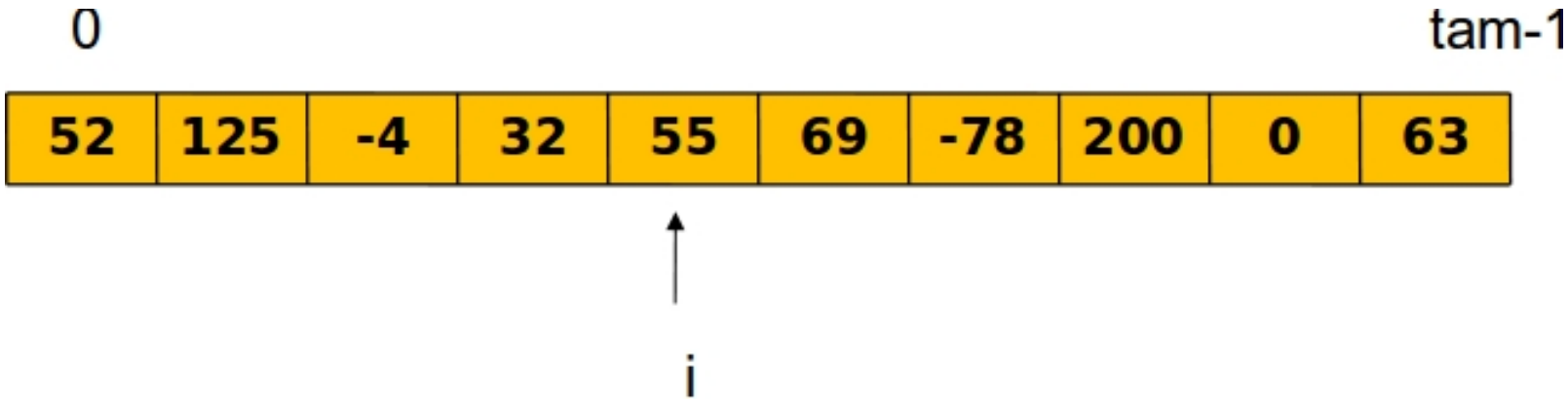
Busca em Arranjo

- Suponha que temos um arranjo de inteiros
- Como fazemos para verificar se um determinado número está lá?
  - Varremos o arranjo, da esquerda para a direita
  - Se acharmos o número, então ele está no arranjo
  - Se chegarmos ao final do arranjo e não acharmos, ele não está
- Busca Seqüencial!
  - Ex: Buscando -53



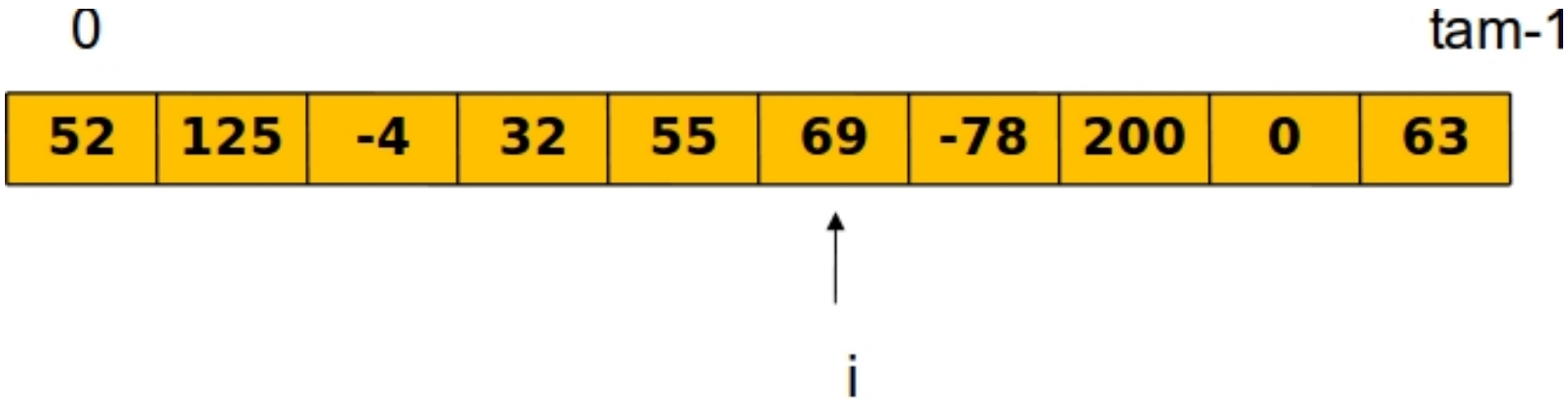
Busca em Arranjo

- Suponha que temos um arranjo de inteiros
- Como fazemos para verificar se um determinado número está lá?
  - Varremos o arranjo, da esquerda para a direita
  - Se acharmos o número, então ele está no arranjo
  - Se chegarmos ao final do arranjo e não acharmos, ele não está
- Busca Seqüencial!
  - Ex: Buscando -53



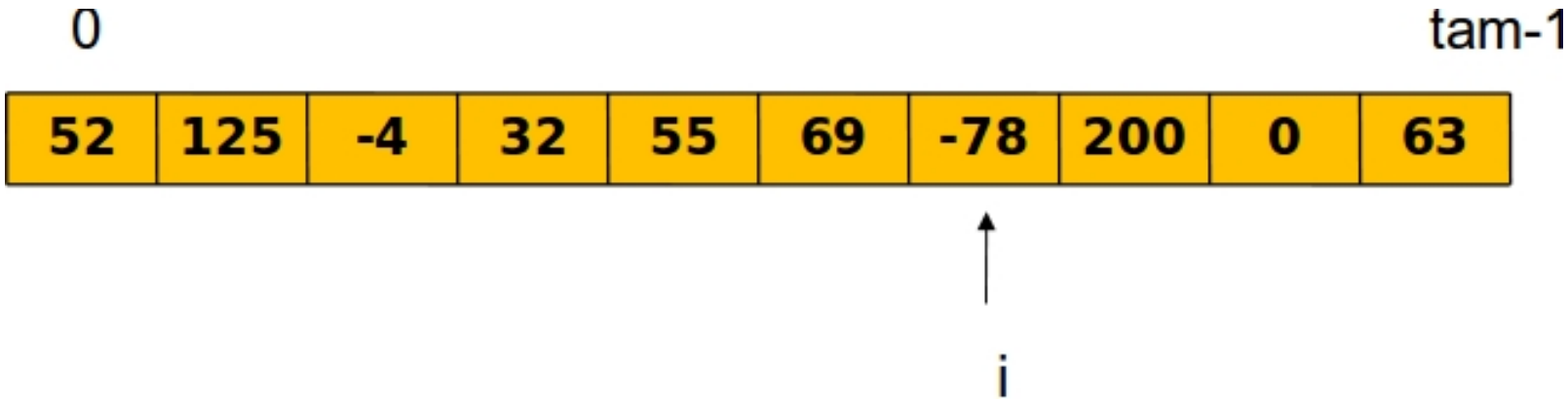
Busca em Arranjo

- Suponha que temos um arranjo de inteiros
- Como fazemos para verificar se um determinado número está lá?
  - Varremos o arranjo, da esquerda para a direita
  - Se acharmos o número, então ele está no arranjo
  - Se chegarmos ao final do arranjo e não acharmos, ele não está
- Busca Seqüencial!
  - Ex: Buscando -53



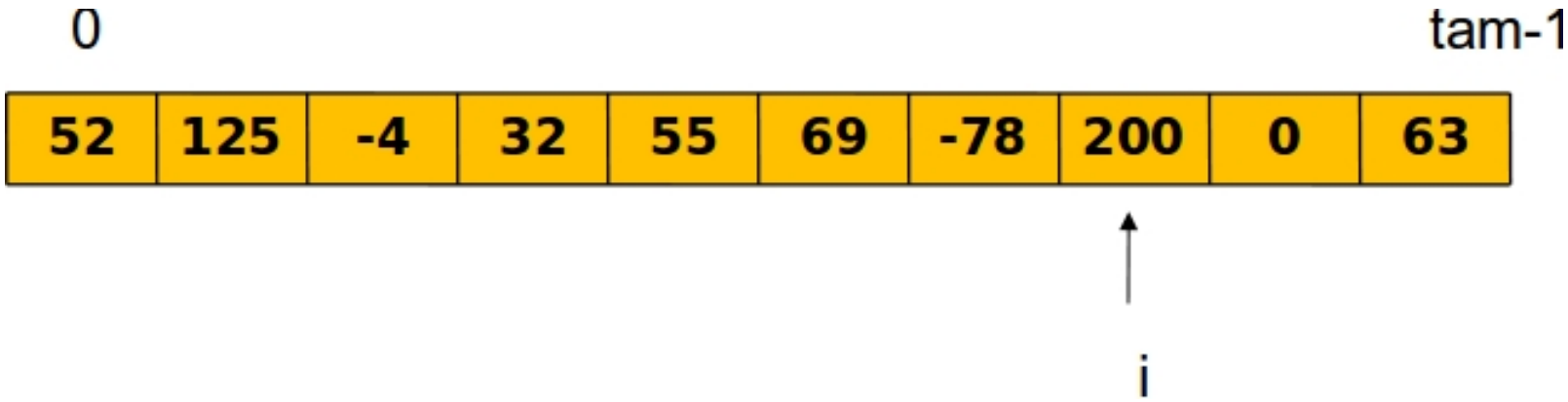
Busca em Arranjo

- Suponha que temos um arranjo de inteiros
- Como fazemos para verificar se um determinado número está lá?
  - Varremos o arranjo, da esquerda para a direita
  - Se acharmos o número, então ele está no arranjo
  - Se chegarmos ao final do arranjo e não acharmos, ele não está
- Busca Seqüencial!
  - Ex: Buscando -53



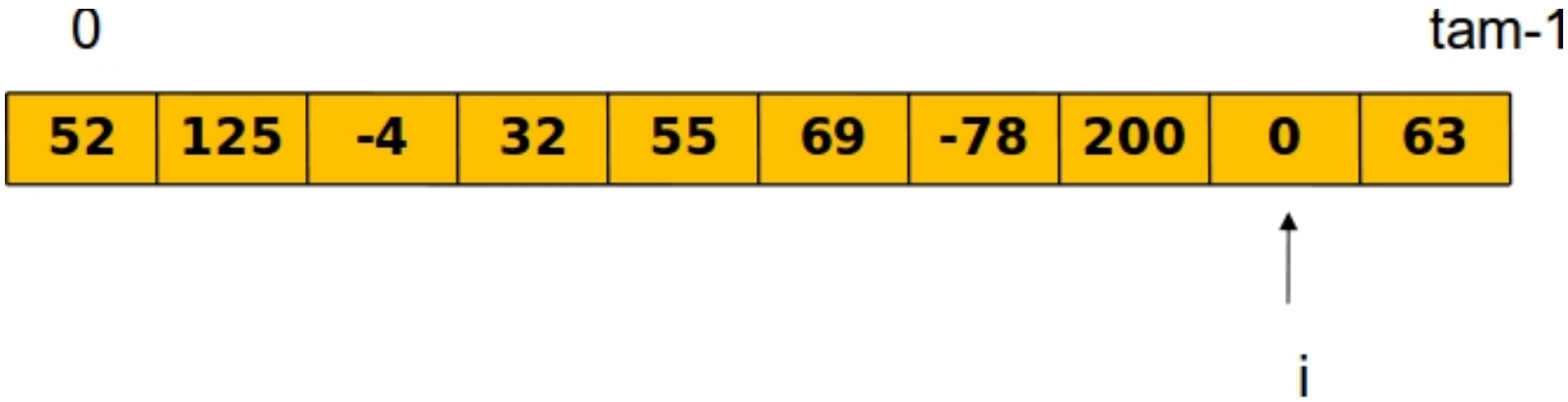
Busca em Arranjo

- Suponha que temos um arranjo de inteiros
- Como fazemos para verificar se um determinado número está lá?
  - Varremos o arranjo, da esquerda para a direita
  - Se acharmos o número, então ele está no arranjo
  - Se chegarmos ao final do arranjo e não acharmos, ele não está
- Busca Seqüencial!
  - Ex: Buscando -53



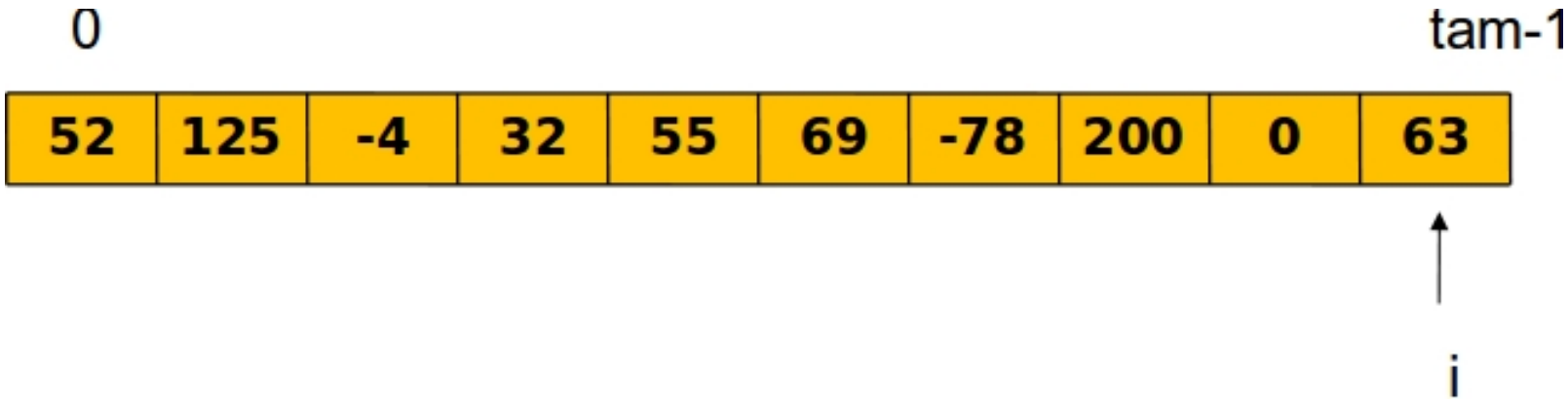
Busca em Arranjo

- Suponha que temos um arranjo de inteiros
- Como fazemos para verificar se um determinado número está lá?
  - Varremos o arranjo, da esquerda para a direita
  - Se acharmos o número, então ele está no arranjo
  - Se chegarmos ao final do arranjo e não acharmos, ele não está
- Busca Seqüencial!
  - Ex: Buscando -53



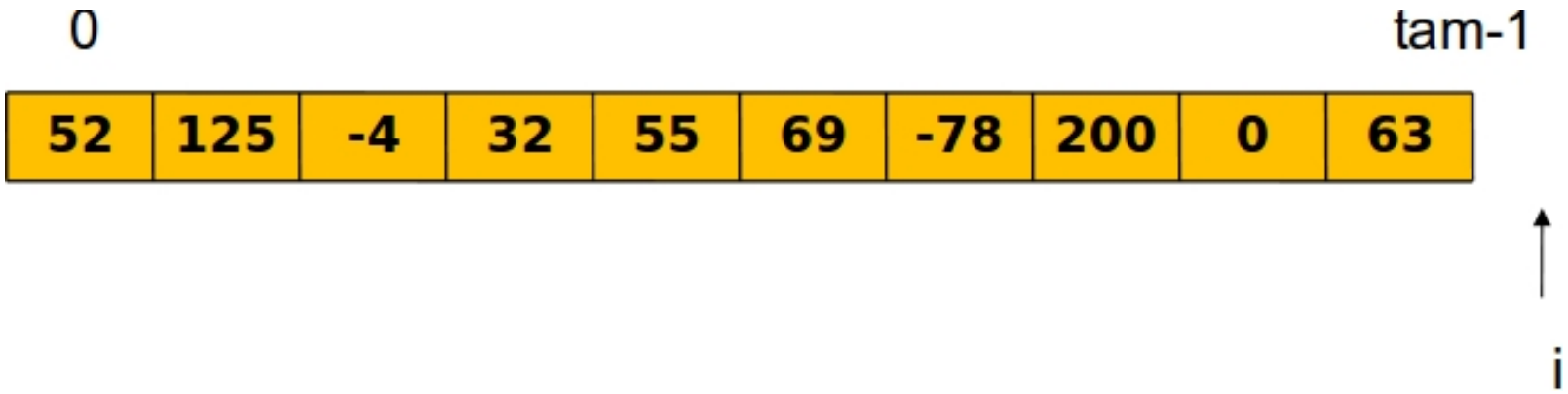
Busca em Arranjo

- Suponha que temos um arranjo de inteiros
- Como fazemos para verificar se um determinado número está lá?
  - Varremos o arranjo, da esquerda para a direita
  - Se acharmos o número, então ele está no arranjo
  - Se chegarmos ao final do arranjo e não acharmos, ele não está
- Busca Seqüencial!
  - Ex: Buscando -53



Busca em Arranjo

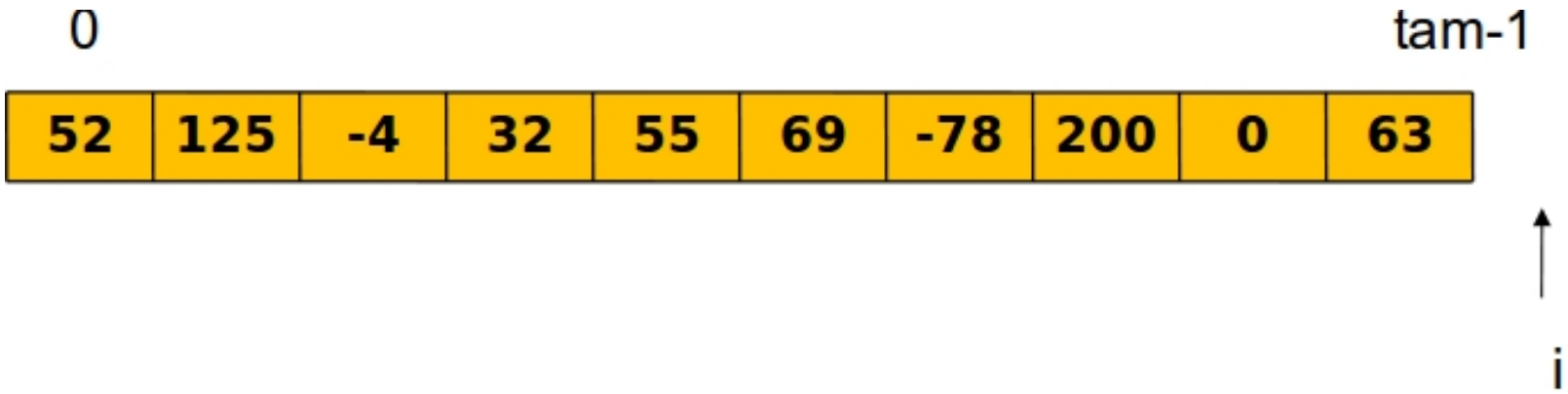
- Suponha que temos um arranjo de inteiros
- Como fazemos para verificar se um determinado número está lá?
  - Varremos o arranjo, da esquerda para a direita
  - Se acharmos o número, então ele está no arranjo
  - Se chegarmos ao final do arranjo e não acharmos, ele não está
- Busca Seqüencial!
  - Ex: Buscando -53



Não encontrou! Como sabemos?

Busca em Arranjo

- Suponha que temos um arranjo de inteiros
- Como fazemos para verificar se um determinado número está lá?
  - Varremos o arranjo, da esquerda para a direita
  - Se acharmos o número, então ele está no arranjo
  - Se chegarmos ao final do arranjo e não acharmos, ele não está
- Busca Seqüencial!
  - Ex: Buscando -53



Não encontrou! Como sabemos? $i == tam$.

Busca em Arranjo

- Então...

```
static int buscaSeq(int[] arr, int el) {
    for (int i=0; i<arr.length; i++)
        if (arr[i] == el) return(i);
    return(-1);
}

public static void main(String[] args) {
    int[] v = {9, 8, 4, 6, 3, 4};

    System.out.println(buscaSeq(v, 4));
    System.out.println(buscaSeq(v, 12));
}
```

Busca em Arranjo

- Então...

```
static int buscaSeq(int[] arr, int el) {
    for (int i=0; i<arr.length; i++)
        if (arr[i] == el) return(i);
    return(-1);
}

public static void main(String[] args) {
    int[] v = {9, 8, 4, 6, 3, 4};

    System.out.println(buscaSeq(v, 4));
    System.out.println(buscaSeq(v, 12));
}
```

Saída

```
$ java Projeto
2
-1
```

Busca em Arranjo

- Então...
- Note que fizemos o método *static*

```
static int buscaSeq(int[] arr, int el) {
    for (int i=0; i<arr.length; i++)
        if (arr[i] == el) return(i);
    return(-1);
}

public static void main(String[] args) {
    int[] v = {9, 8, 4, 6, 3, 4};

    System.out.println(buscaSeq(v, 4));
    System.out.println(buscaSeq(v, 12));
}
```

Saída

```
$ java Projeto
2
-1
```

Busca em Arranjo

- Então...
- Note que fizemos o método *static*
  - Ele já dispõe, em seus parâmetros, de toda a informação de que precisa para executar

```
static int buscaSeq(int[] arr, int el) {
    for (int i=0; i<arr.length; i++)
        if (arr[i] == el) return(i);
    return(-1);
}

public static void main(String[] args) {
    int[] v = {9, 8, 4, 6, 3, 4};

    System.out.println(buscaSeq(v, 4));
    System.out.println(buscaSeq(v, 12));
}
```

Saída

```
$ java Projeto
2
-1
```

Busca em Arranjo

- Então...
- Note que fizemos o método *static*
  - Ele já dispõe, em seus parâmetros, de toda a informação de que precisa para executar
    - ★ Não precisa de nada mais específico de cada objeto particular

```
static int buscaSeq(int[] arr, int el) {
    for (int i=0; i<arr.length; i++)
        if (arr[i] == el) return(i);
    return(-1);
}

public static void main(String[] args) {
    int[] v = {9, 8, 4, 6, 3, 4};

    System.out.println(buscaSeq(v, 4));
    System.out.println(buscaSeq(v, 12));
}
```

Saída

```
$ java Projeto
2
-1
```

Busca em Arranjo

- Então...
- Note que fizemos o método *static*
  - Ele já dispõe, em seus parâmetros, de toda a informação de que precisa para executar
    - ★ Não precisa de nada mais específico de cada objeto particular
  - Sendo static, não há uma cópia por objeto – poupa memória

```
static int buscaSeq(int[] arr, int el) {
    for (int i=0; i<arr.length; i++)
        if (arr[i] == el) return(i);
    return(-1);
}

public static void main(String[] args) {
    int[] v = {9, 8, 4, 6, 3, 4};

    System.out.println(buscaSeq(v, 4));
    System.out.println(buscaSeq(v, 12));
}
```

Saída

```
$ java Projeto
2
-1
```

Busca em Arranjo

- Como ficaria a busca com objetos?

Busca em Arranjo

- Como ficaria a busca com objetos?
  - Ex: busque, no condomínio, a primeira casa com piscina de raio 3

Busca em Arranjo

- Como ficaria a busca com objetos?
  - Ex: busque, no condomínio, a primeira casa com piscina de raio 3

```
class Projeto {
    Residencia[] condominio;
    ...

    Projeto(int tam) {
        condominio = new Residencia[tam];
    }

    int buscaPiscSeq(double raio) {
        for (int i=0; i<this.condominio.length; i++)
            if (this.condominio[i].piscina.raio ==
                                        raio) return(i);
        return(-1);
    }

    public static void main(String[] args) {
        Projeto pr = new Projeto(5);

        for (int i=0; i<pr.condominio.length; i++) {
            AreaCasa c = new AreaCasa();
            AreaPiscina p = new AreaPiscina(i+2);
            Residencia r = new Residencia(c,p);
            pr.adicionaRes(r);
        }

        System.out.println(pr.buscaPiscSeq(3));
        System.out.println(pr.buscaPiscSeq(15));
    }
}
```

Busca em Arranjo

- Como ficaria a busca com objetos?
  - Ex: busque, no condomínio, a primeira casa com piscina de raio 3

```
class Projeto {
    Residencia[] condominio;
    ...

    Projeto(int tam) {
        condominio = new Residencia[tam];
    }

    int buscaPiscSeq(double raio) {
        for (int i=0; i<this.condominio.length; i++)
            if (this.condominio[i].piscina.raio ==
                                        raio) return(i);
        return(-1);
    }

    public static void main(String[] args) {
        Projeto pr = new Projeto(5);

        for (int i=0; i<pr.condominio.length; i++) {
            AreaCasa c = new AreaCasa();
            AreaPiscina p = new AreaPiscina(i+2);
            Residencia r = new Residencia(c,p);
            pr.adicionaRes(r);
        }

        System.out.println(pr.buscaPiscSeq(3));
        System.out.println(pr.buscaPiscSeq(15));
    }
}
```

Saída

```
$ java Projeto
1
-1
```

Busca em Arranjo

- Como ficaria a busca com objetos?
  - Ex: busque, no condomínio, a primeira casa com piscina de raio 3
  - Note que o método agora não é *static*, pois efetua a busca no arranjo de condomínios do objeto

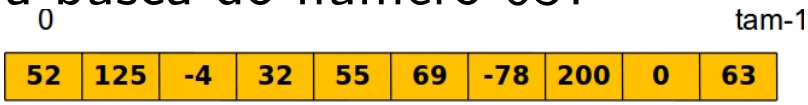
```
class Projeto {
    Residencia[] condominio;
    ...

    Projeto(int tam) {
        condominio = new Residencia[tam];
    }

    int buscaPiscSeq(double raio) {
        for (int i=0; i<this.condominio.length; i++)
            if (this.condominio[i].piscina.raio ==
                                        raio) return(i);
        return(-1);
    }

    public static void main(String[] args) {
        Projeto pr = new Projeto(5);

        for (int i=0; i<pr.condominio.length; i++) {
            AreaCasa c = new AreaCasa();
            AreaPiscina p = new AreaPiscina(i+2);
            Residencia r = new Residencia(c,p);
            pr.adicionaRes(r);
        }

        System.out.println(pr.buscaPiscSeq(3));
        System.out.println(pr.buscaPiscSeq(15));
    }
}
```

Saída

```
$ java Projeto
1
-1
```

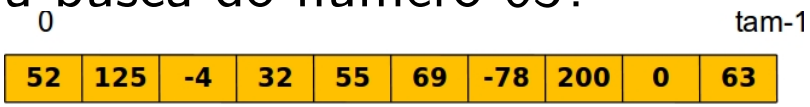
Busca Seqüencial

- O que posso fazer para melhorar a busca do número 63?

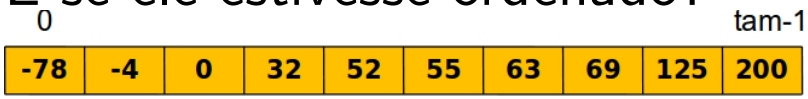


Busca Seqüencial

- O que posso fazer para melhorar a busca do número 63?

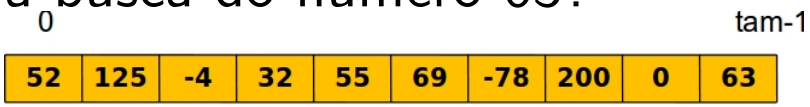


- E se ele estivesse ordenado?

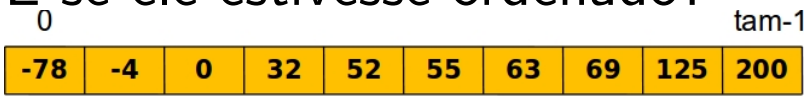


Busca Seqüencial

- O que posso fazer para melhorar a busca do número 63?



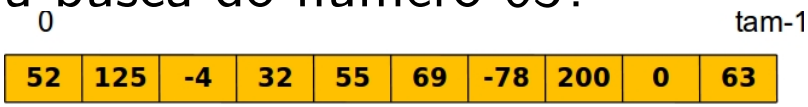
- E se ele estivesse ordenado?



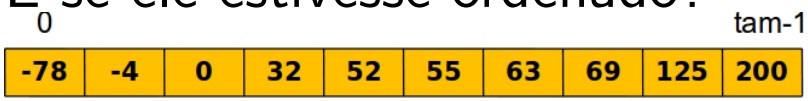
  - Poderíamos parar a busca assim que encontrássemos um número maior que ele

Busca Seqüencial

- O que posso fazer para melhorar a busca do número 63?



- E se ele estivesse ordenado?

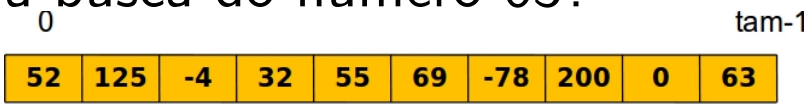


  - Poderíamos parar a busca assim que encontrássemos um número maior que ele

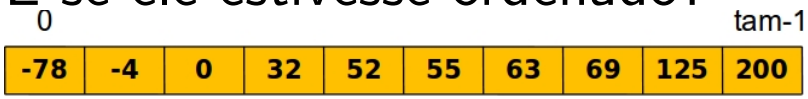
```
static int buscaSeq(int[] arr, int el) {
    for (int i=0; i<arr.length; i++) {
        if (arr[i] == el) return(i);
        if (arr[i] > el) break;
    }
    return(-1);
}

public static void main(String[] args) {
    int[] v = {-78,-4,0,32,52,55,63,69,125,200};

    System.out.println(buscaSeq(v, 63));
    System.out.println(buscaSeq(v, 68));
}
```

Busca Seqüencial

- O que posso fazer para melhorar a busca do número 63?



- E se ele estivesse ordenado?



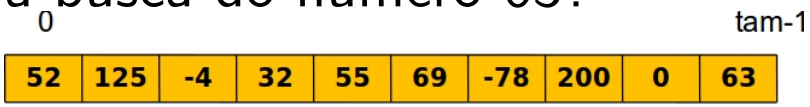
  - Poderíamos parar a busca assim que encontrássemos um número maior que ele

```
static int buscaSeq(int[] arr, int el) {
    for (int i=0; i<arr.length; i++) {
        if (arr[i] == el) return(i);
        if (arr[i] > el) break;
    }
    return(-1);
}

public static void main(String[] args) {
    int[] v = {-78,-4,0,32,52,55,63,69,125,200};

    System.out.println(buscaSeq(v, 63));
    System.out.println(buscaSeq(v, 68));
}
```

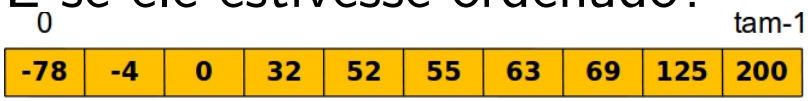
- Executamos menos comparações no caso do elemento não estar no arranjo

Busca Seqüencial

- O que posso fazer para melhorar a busca do número 63?



- E se ele estivesse ordenado?



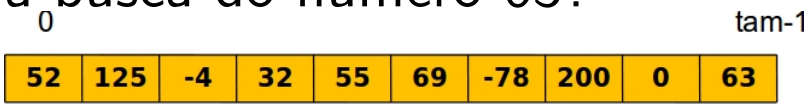
  - Poderíamos parar a busca assim que encontrássemos um número maior que ele

```
static int buscaSeq(int[] arr, int el) {
    for (int i=0; i<arr.length; i++) {
        if (arr[i] == el) return(i);
        if (arr[i] > el) break;
    }
    return(-1);
}

public static void main(String[] args) {
    int[] v = {-78,-4,0,32,52,55,63,69,125,200};

    System.out.println(buscaSeq(v, 63));
    System.out.println(buscaSeq(v, 68));
}
```

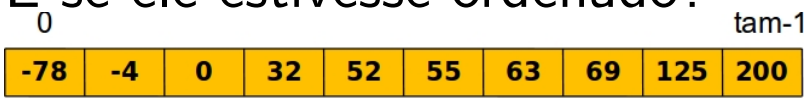
- Executamos menos comparações no caso do elemento não estar no arranjo
- Estar ordenado também torna fácil algumas tarefas:

Busca Seqüencial

- O que posso fazer para melhorar a busca do número 63?



- E se ele estivesse ordenado?



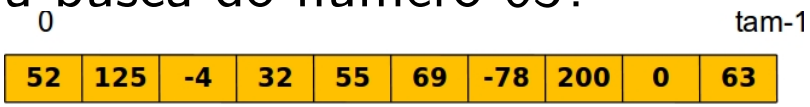
  - Poderíamos parar a busca assim que encontrássemos um número maior que ele

```
static int buscaSeq(int[] arr, int el) {
    for (int i=0; i<arr.length; i++) {
        if (arr[i] == el) return(i);
        if (arr[i] > el) break;
    }
    return(-1);
}

public static void main(String[] args) {
    int[] v = {-78,-4,0,32,52,55,63,69,125,200};

    System.out.println(buscaSeq(v, 63));
    System.out.println(buscaSeq(v, 68));
}
```

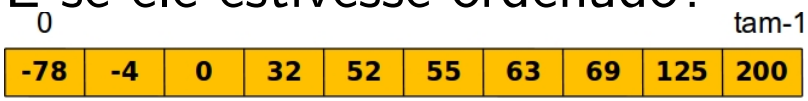
- Executamos menos comparações no caso do elemento não estar no arranjo
- Estar ordenado também torna fácil algumas tarefas:
  - Busca pelo menor elemento:

Busca Seqüencial

- O que posso fazer para melhorar a busca do número 63?



- E se ele estivesse ordenado?



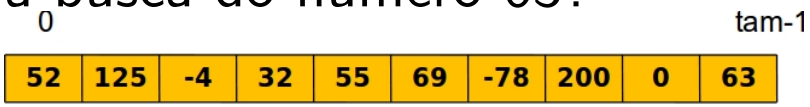
  - Poderíamos parar a busca assim que encontrássemos um número maior que ele

```
static int buscaSeq(int[] arr, int el) {
    for (int i=0; i<arr.length; i++) {
        if (arr[i] == el) return(i);
        if (arr[i] > el) break;
    }
    return(-1);
}

public static void main(String[] args) {
    int[] v = {-78,-4,0,32,52,55,63,69,125,200};

    System.out.println(buscaSeq(v, 63));
    System.out.println(buscaSeq(v, 68));
}
```

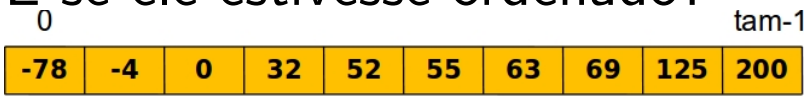
- Executamos menos comparações no caso do elemento não estar no arranjo
- Estar ordenado também torna fácil algumas tarefas:
  - Busca pelo menor elemento: *v[0]*

Busca Seqüencial

- O que posso fazer para melhorar a busca do número 63?



- E se ele estivesse ordenado?



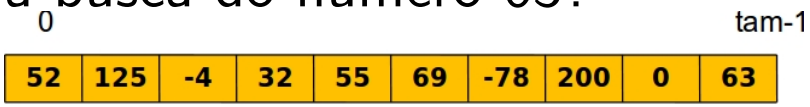
  - Poderíamos parar a busca assim que encontrássemos um número maior que ele

```
static int buscaSeq(int[] arr, int el) {
    for (int i=0; i<arr.length; i++) {
        if (arr[i] == el) return(i);
        if (arr[i] > el) break;
    }
    return(-1);
}

public static void main(String[] args) {
    int[] v = {-78,-4,0,32,52,55,63,69,125,200};

    System.out.println(buscaSeq(v, 63));
    System.out.println(buscaSeq(v, 68));
}
```

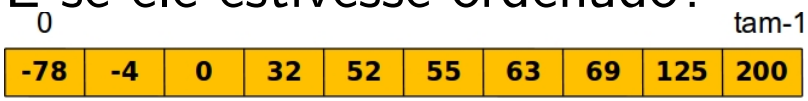
- Executamos menos comparações no caso do elemento não estar no arranjo
- Estar ordenado também torna fácil algumas tarefas:
  - Busca pelo menor elemento: *v[0]*
  - Busca pelo maior elemento:

Busca Seqüencial

- O que posso fazer para melhorar a busca do número 63?



- E se ele estivesse ordenado?



  - Poderíamos parar a busca assim que encontrássemos um número maior que ele

```
static int buscaSeq(int[] arr, int el) {
    for (int i=0; i<arr.length; i++) {
        if (arr[i] == el) return(i);
        if (arr[i] > el) break;
    }
    return(-1);
}

public static void main(String[] args) {
    int[] v = {-78,-4,0,32,52,55,63,69,125,200};

    System.out.println(buscaSeq(v, 63));
    System.out.println(buscaSeq(v, 68));
}
```

- Executamos menos comparações no caso do elemento não estar no arranjo
- Estar ordenado também torna fácil algumas tarefas:
  - Busca pelo menor elemento: *v[0]*
  - Busca pelo maior elemento: *v[v.length-1]*