Aula 16 – Strings, Entrada e Busca Sequencial

Norton Trevisan Roman

28 de maio de 2018

Revendo Strings

- Até agora, como havíamos definido strings?

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

- Podemos então reescrever os nomes dos materiais da piscina

Revendo Strings

- Podemos então reescrever os nomes dos materiais da piscina
- De...

```
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

- Podemos então reescrever os nomes dos materiais da piscina
- De...
- Para...

```
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'}};
```

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

Strings

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

Strings

- Como fazemos para acessarmos o caractere 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 caractere 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 caractere 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

- E como podemos modificar um caractere no String?

Strings

- E como podemos modificar um caractere no String?
- Não podemos. Objetos `String` são constantes

Strings

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

Strings

- E como podemos modificar um caractere 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

- E como podemos modificar um caractere 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...

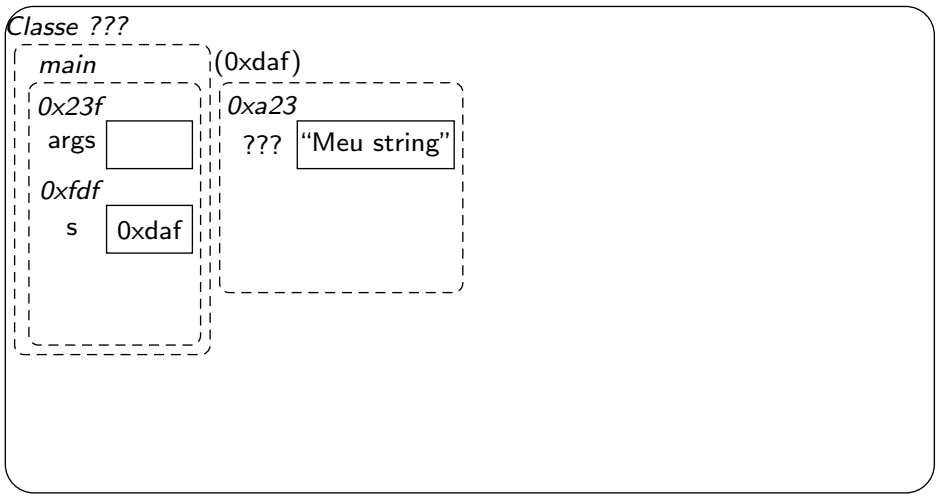
- O que houve com ‘‘Meu string’’?
- Vejamos o comportamento da memória:

```
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:

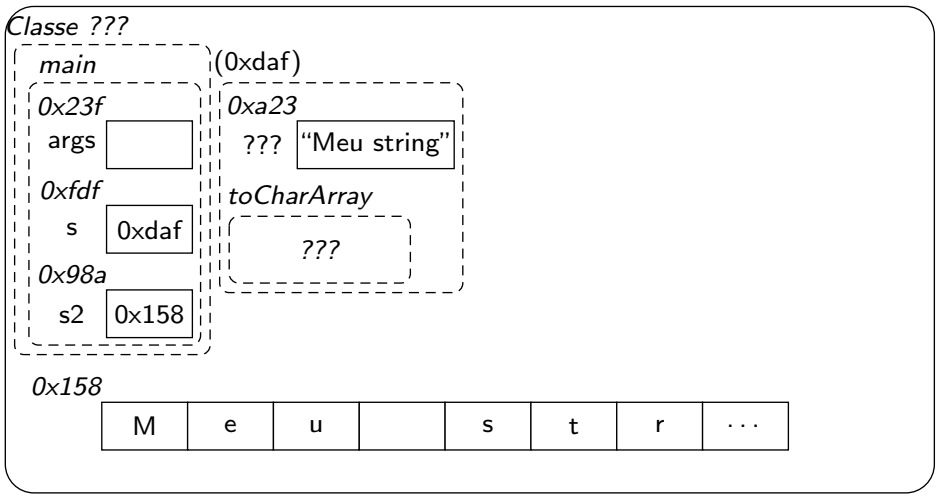
```
public static void main(String[]
                                args) {
  String s = "Meu string";
  char[] s2 = s.toCharArray();
  s2[4] = 'b';
  s = String.valueOf(s2);
  //ou s = new String(s2);
  System.out.println(s);
}
```



Olhando mais de perto...

- O que houve com ‘‘Meu string’’?
- Vejamos o comportamento da memória:

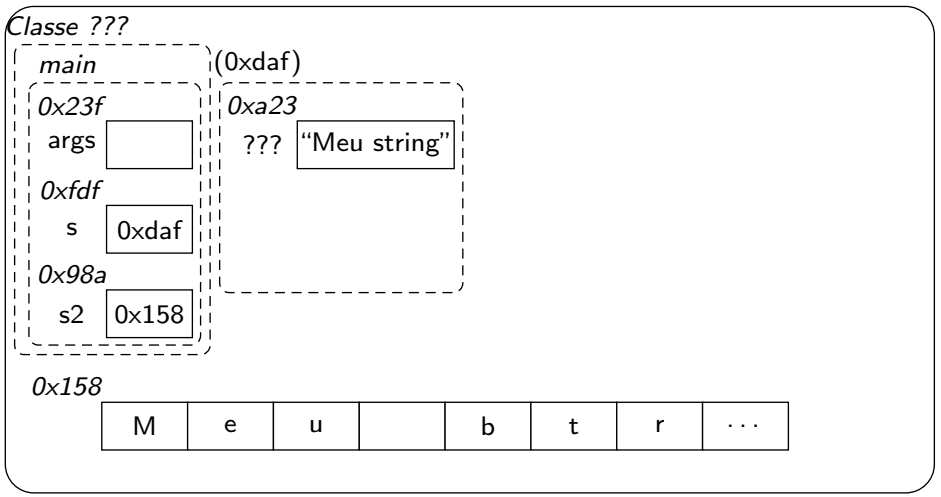
```
public static void main(String[]
                                  args) {
  String s = "Meu string";
  char[] s2 = s.toCharArray();
  s2[4] = 'b';
  s = String.valueOf(s2);
  //ou s = new String(s2);
  System.out.println(s);
}
```



Olhando mais de perto...

- O que houve com ‘‘Meu string’’?
- Vejamos o comportamento da memória:

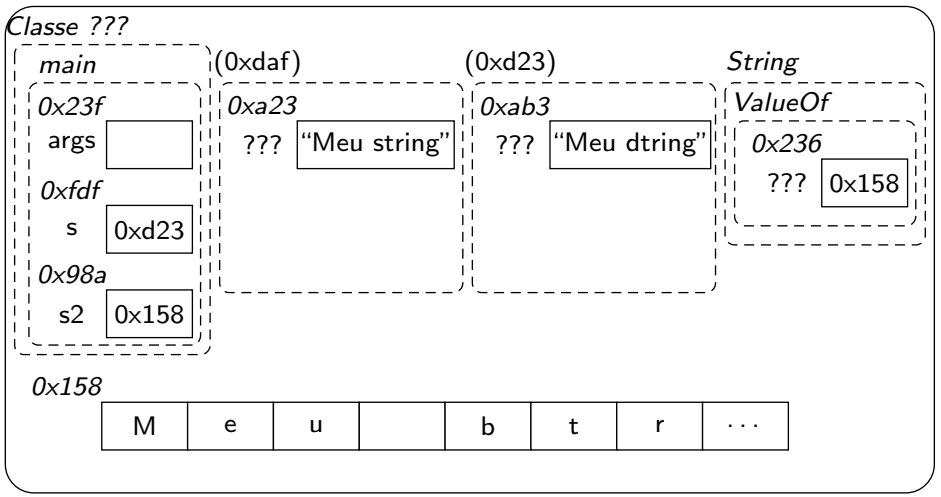
```
public static void main(String[]
                                  args) {
  String s = "Meu string";
  char[] s2 = s.toCharArray();
  s2[4] = 'b';
  s = String.valueOf(s2);
  //ou s = new String(s2);
  System.out.println(s);
}
```



Olhando mais de perto...

- O que houve com ‘‘Meu string’’?
- Vejamos o comportamento da memória:

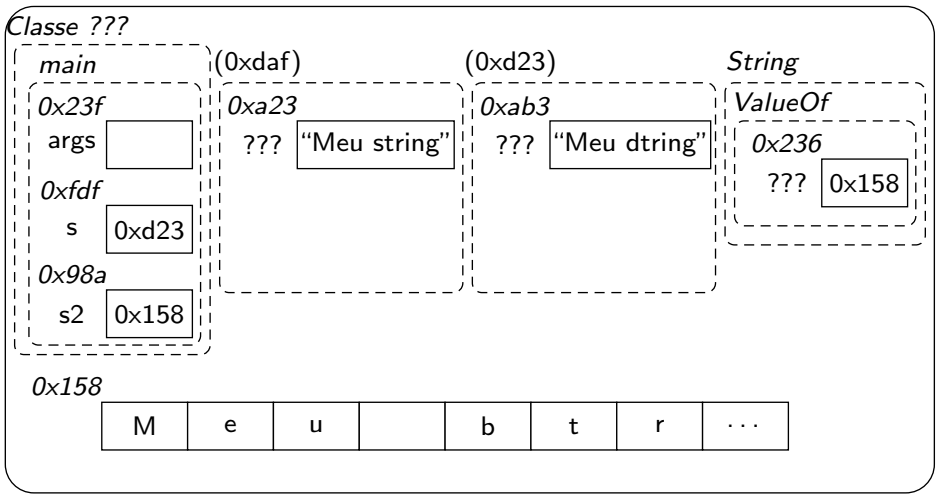
```
public static void main(String[]
                                args) {
  String s = "Meu string";
  char[] s2 = s.toCharArray();
  s2[4] = 'b';
  s = String.valueOf(s2);
  //ou s = new String(s2);
  System.out.println(s);
}
```



Olhando mais de perto...

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

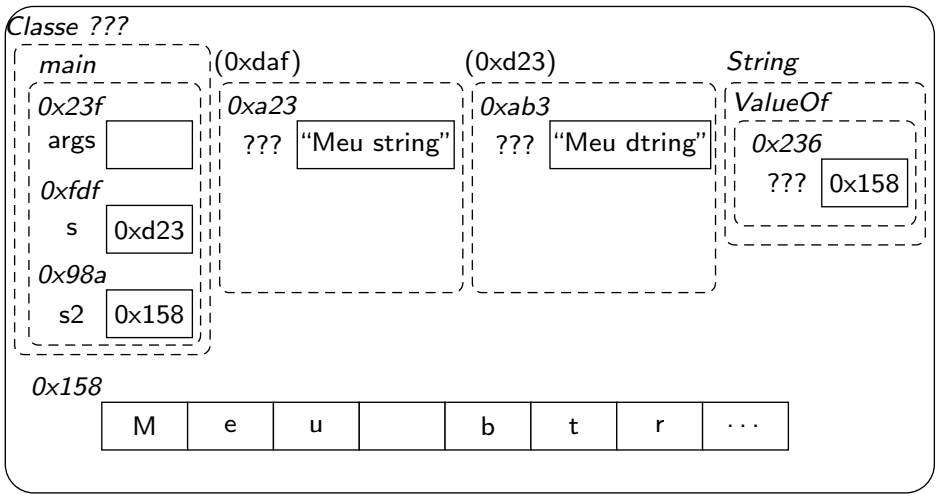
```
public static void main(String[]
                                args) {
  String s = "Meu string";
  char[] s2 = s.toCharArray();
  s2[4] = 'b';
  s = String.valueOf(s2);
  //ou s = new String(s2);
  System.out.println(s);
}
```



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)
  - 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

```
String str1 = "Exemplo";
String str2 = new String("Exemplo");
String str3 = "Outro exemplo";

if (str1 == str2) ...
if (str1 == str3) ...
str3 = str1;
if (str1 == str3) ...
```

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

```
String str1 = "Exemplo";
String str2 = new String("Exemplo");
String str3 = "Outro exemplo";

if (str1 == str2) ... // false (objetos diferentes)
if (str1 == str3) ...
str3 = str1;
if (str1 == str3) ...
```

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

```
String str1 = "Exemplo";
String str2 = new String("Exemplo");
String str3 = "Outro exemplo";

if (str1 == str2) ... // false (objetos diferentes)
if (str1 == str3) ... // false (objetos diferentes)
str3 = str1;
if (str1 == str3) ...
```

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

```
String str1 = "Exemplo";
String str2 = new String("Exemplo");
String str3 = "Outro exemplo";

if (str1 == str2) ... // false (objetos diferentes)
if (str1 == str3) ... // false (objetos diferentes)
str3 = str1;
if (str1 == str3) ... // true (mesmo objeto)
```

Strings

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- 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

- 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://docs.oracle.com/javase/7/docs/api/java/lang/String.html

Strings

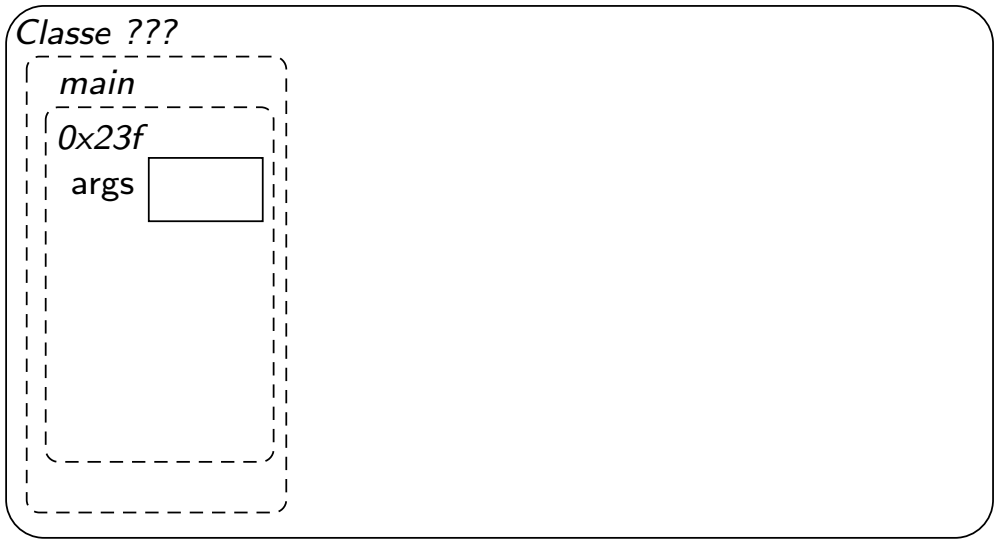
- O que será impresso?

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

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

Strings

- O que será impresso?

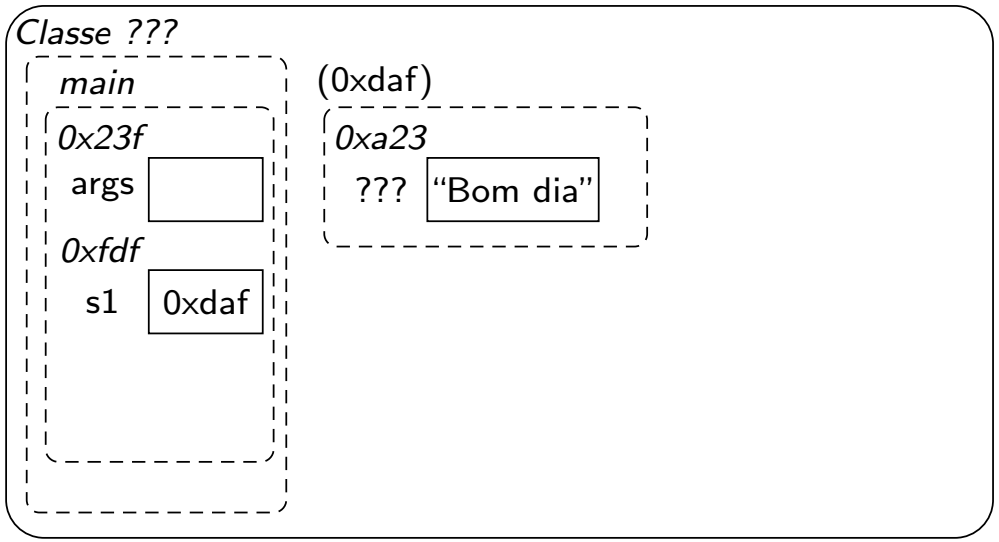
```
public static void main(String[] args)
{
  String s1 = "Bom dia";
  String s2 = s1;

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



Strings

- O que será impresso?

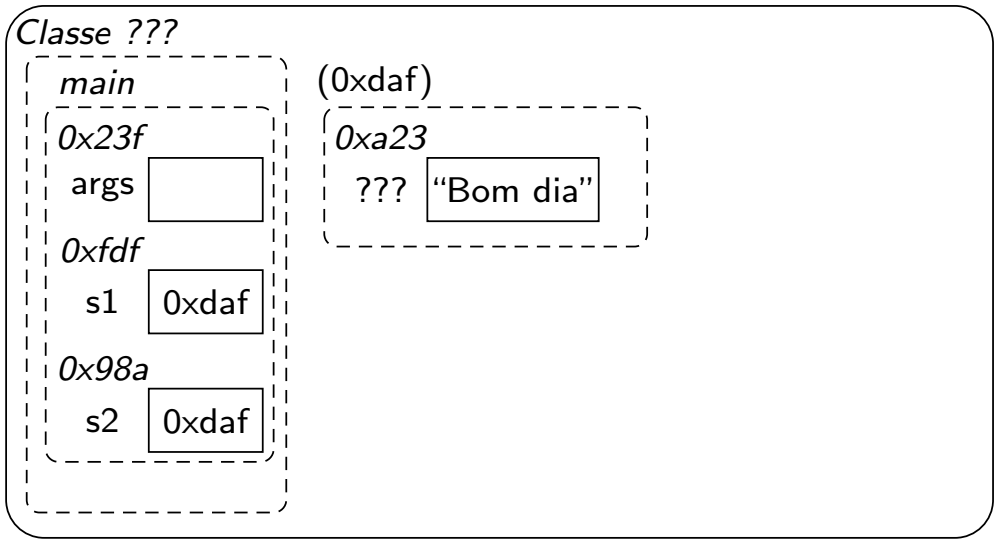
```
public static void main(String[] args)
{
  String s1 = "Bom dia";
  String s2 = s1;

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



Strings

- O que será impresso?

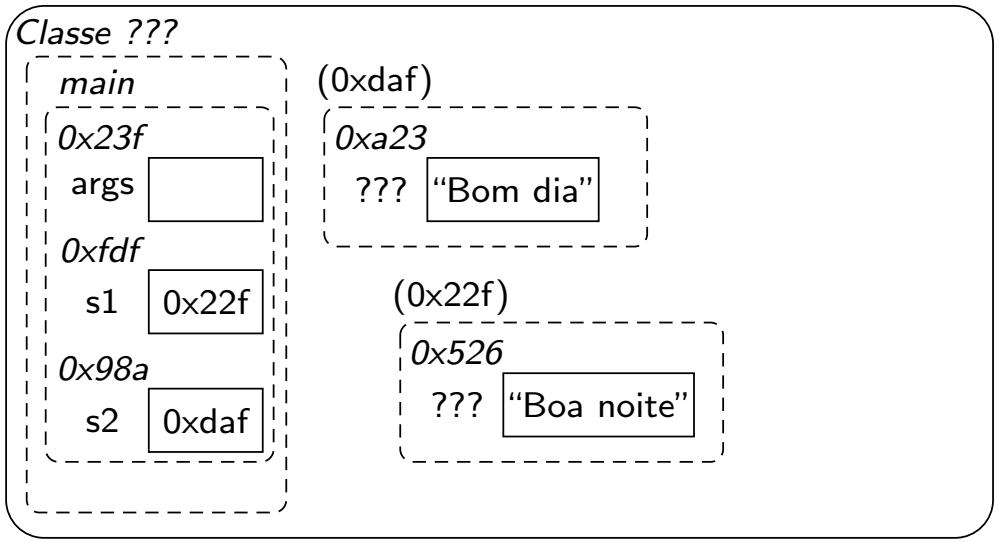
```
public static void main(String[] args)
{
  String s1 = "Bom dia";
  String s2 = s1;

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



Strings

- O que será impresso?

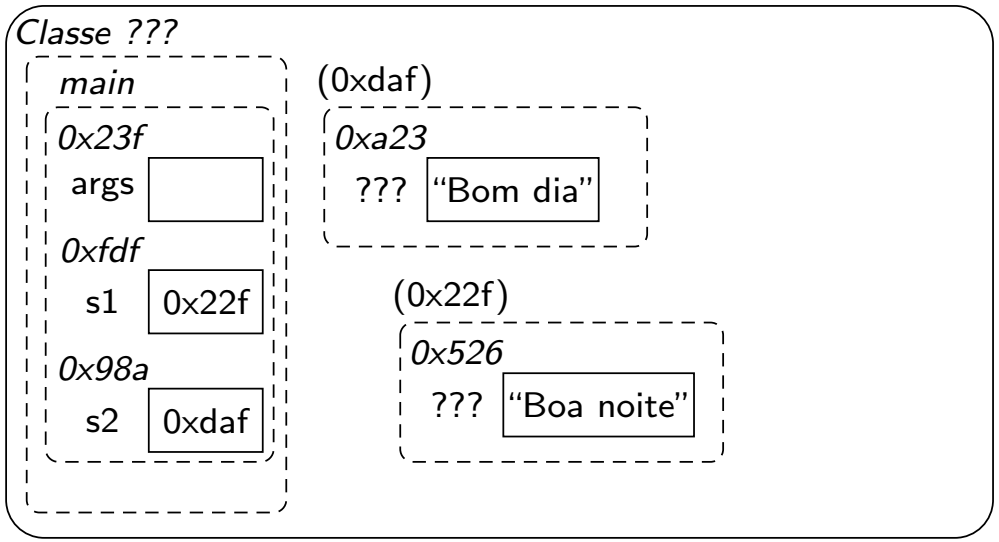
```
public static void main(String[] args)
{
  String s1 = "Bom dia";
  String s2 = s1;

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



Strings

- 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

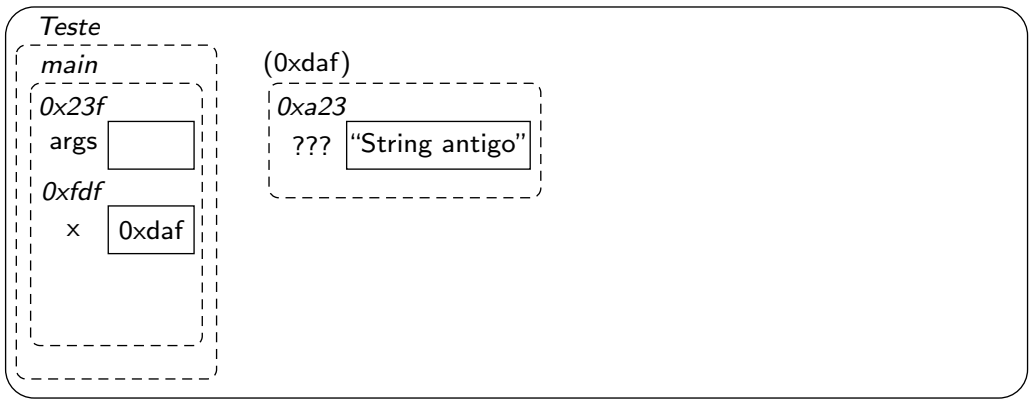
- E qual a saída desse código?

```
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?

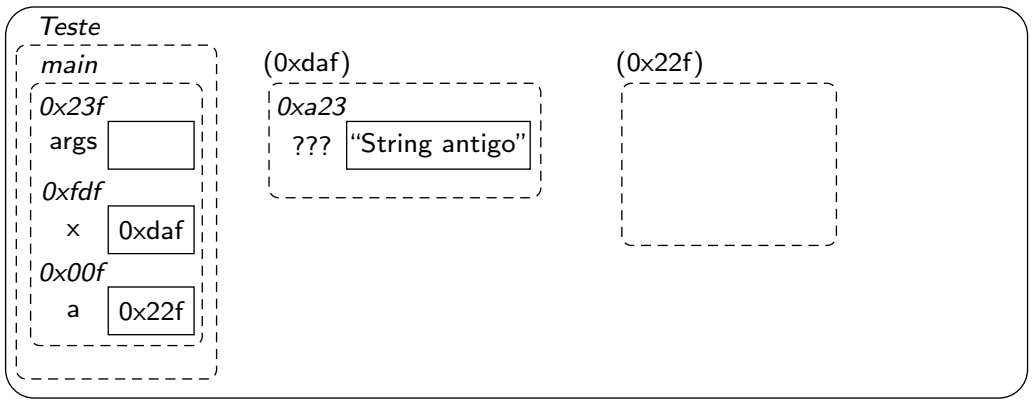
```
class Teste {
  void misterio(String y) {
    y = "Novo string";
  }

  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?

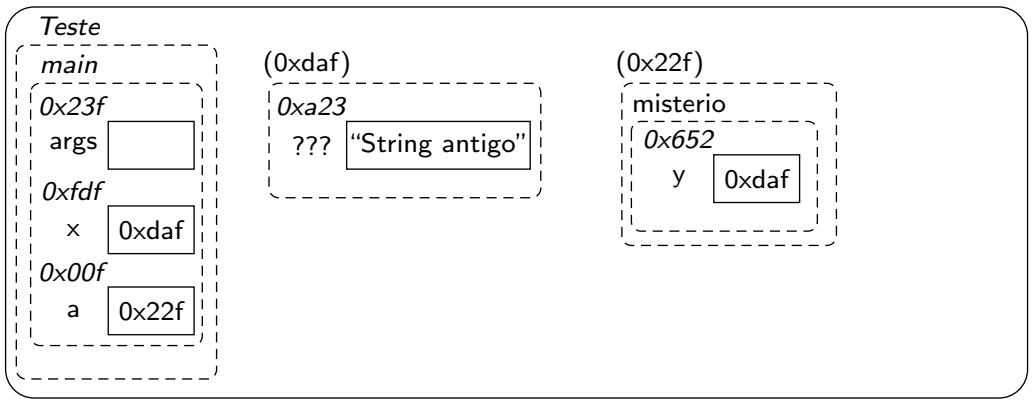
```
class Teste {
  void misterio(String y) {
    y = "Novo string";
  }

  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?

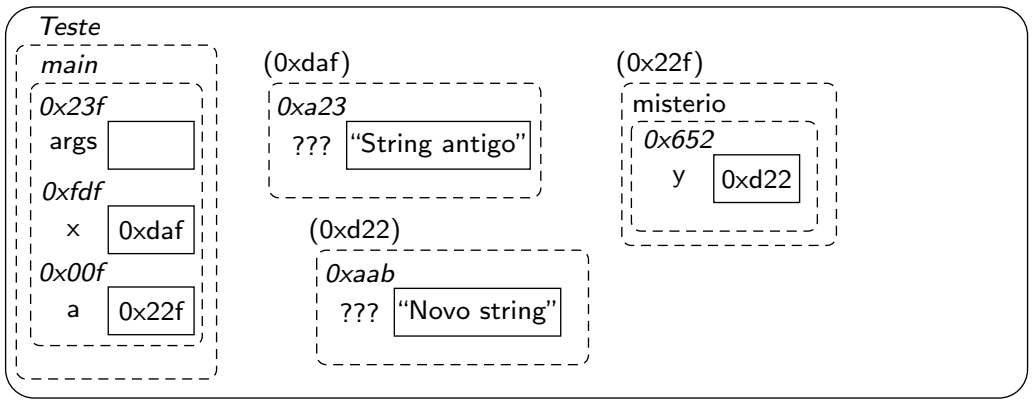
```
class Teste {
  void misterio(String y) {
    y = "Novo string";
  }

  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?

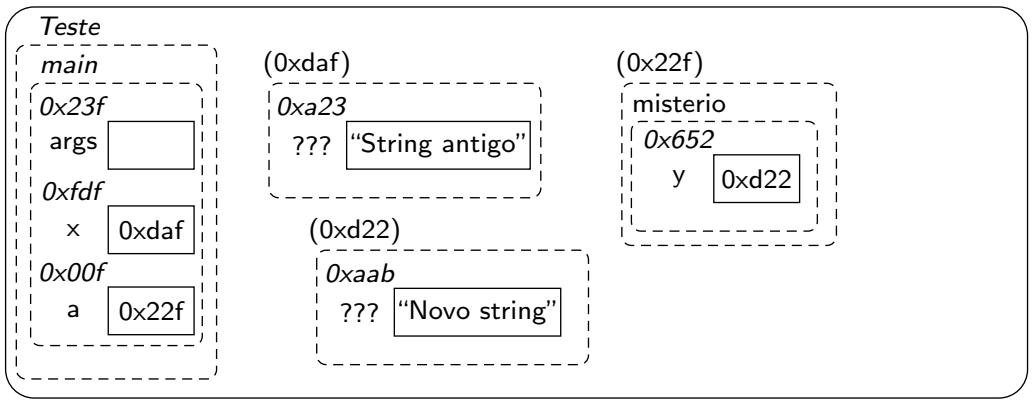
```
class Teste {
  void misterio(String y) {
    y = "Novo string";
  }

  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?
- “String antigo”

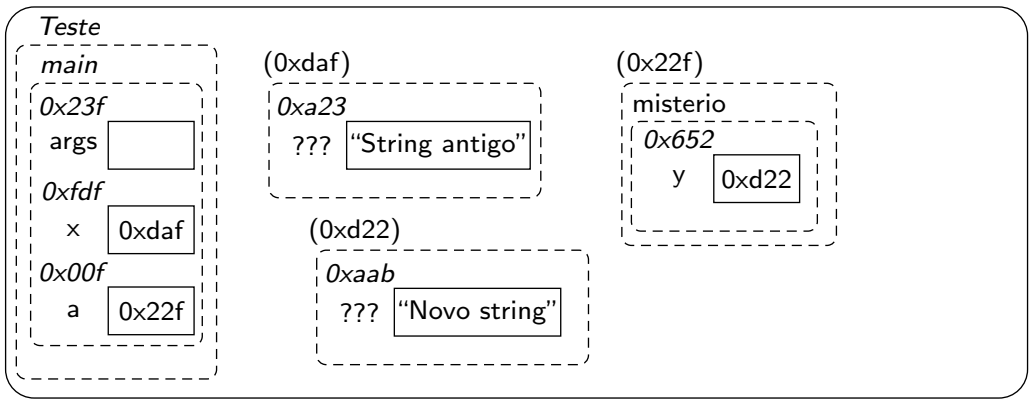
```
class Teste {
  void misterio(String y) {
    y = "Novo string";
  }

  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?
- “String antigo”
- Modificamos o parâmetro, e não a variável do objeto

```
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);
  }
}
```



Parâmetros da Linha de Comando

- Considere o método `main`

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


}
```

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

- Vamos ver...

Parâmetros da Linha de Comando

- Vamos ver...

```
$java AreaPiscina
```

Parâmetros da Linha de Comando

- Vamos ver...

```
$java AreaPiscina

$ java AreaPiscina oba
oba
```

Parâmetros da Linha de Comando

- Vamos ver...

```
$java AreaPiscina

$ java AreaPiscina oba
oba

$ java AreaPiscina oba 23
oba
23
```

Parâmetros da Linha de Comando

- Vamos ver...

```
$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

- Vamos ver...
  - Escreve os argumentos passados na linha de comando

```
$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

- Vamos ver...
  - Escreve os argumentos passados na linha de comando
  - Informações adicionais separadas por espaço

```
$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

- Vamos ver...
  - Escreve os argumentos passados na linha de comando
  - Informações adicionais separadas por espaço
  - Usadas para passar alguma informação inicial ao programa

```
$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

- Vamos ver...
  - Escreve os argumentos passados na linha de comando
  - Informações adicionais separadas por espaço
  - Usadas para passar alguma informação inicial ao programa

```
$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 {
  ...
  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

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 {
  ...
  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 {
  ...
  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 {
  ...
  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 {
  ...
  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 {
  ...
  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	–

`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

- O ideal seria trocar mensagens

Entrada

- O ideal seria trocar mensagens
  - Escrevendo coisas na tela (`println()`)

Entrada

- O ideal seria trocar mensagens
  - Escrevendo coisas na tela (`println()`)
  - Lendo coisas do teclado

Entrada

- O ideal seria trocar mensagens
  - Escrevendo coisas na tela (`println()`)
  - Lendo coisas do teclado
- Como?

Entrada

- O ideal seria trocar mensagens
  - Escrevendo coisas na tela (println())
  - Lendo coisas do teclado
- Como? Via Scanner (Java $\geq$ 5)

```
import java.util.Scanner;

class Projeto {
  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 {
  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 {
  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

- Fazemos a pergunta...

```
import java.util.Scanner;

class Projeto {
  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

- Fazemos a pergunta...
- Scanner lê o string digitado (até um *enter* ser pressionado)

```
import java.util.Scanner;

class Projeto {
  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

- Fazemos a pergunta...
- Scanner lê o string digitado (até um *enter* ser pressionado)
- Recuperamos o inteiro nesse string

```
import java.util.Scanner;

class Projeto {
  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

- Usamos para inicializar o objeto da classe `Projeto`

```
import java.util.Scanner;

class Projeto {
  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

- Usamos para inicializar o objeto da classe `Projeto`
- Criamos residências padrão no condomínio

```
import java.util.Scanner;

class Projeto {
  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

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

```
import java.util.Scanner;

class Projeto {
  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

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

```
import java.util.Scanner;

class Projeto {
  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

- 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

```
import java.util.Scanner;

class Projeto {
  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);
  }
}
```

Scanner

- Scanner pode também ser usado para varrer as informações em um string

Scanner

- 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

- 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

- 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

- 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 Sequencial

Busca em Arranjo

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

Busca em Arranjo

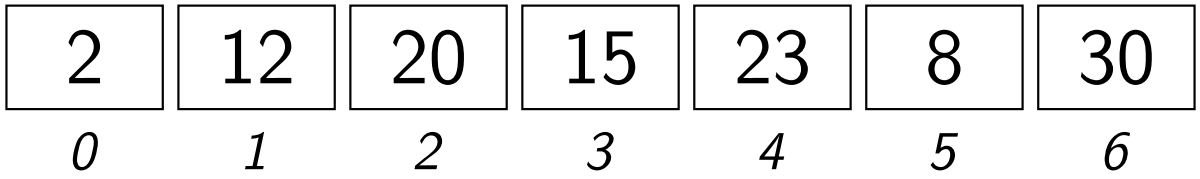
- Suponha que temos um arranjo de inteiros
- Como fazemos para verificar se um determinado número está lá?
  - Varremos o arranjo, da esquerda para a direita
  - Se acharmos o número, então ele está no arranjo
  - Se chegarmos ao final do arranjo e não acharmos, ele não está

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 Sequencial!

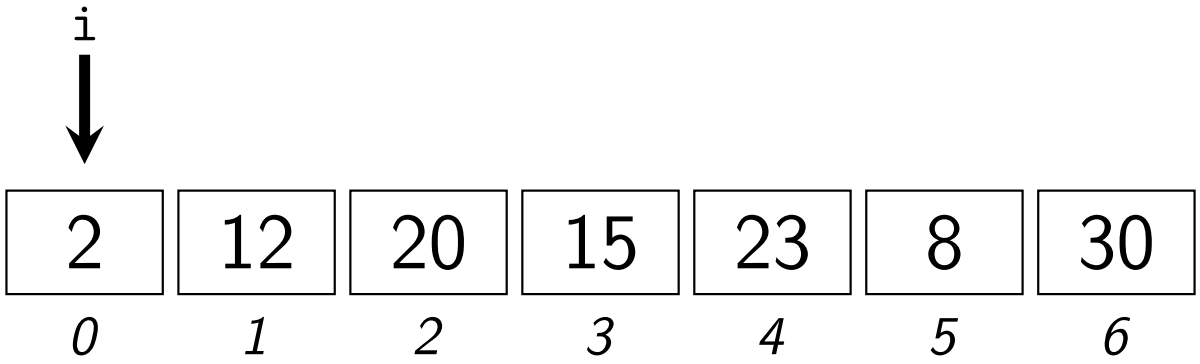
Busca em Arranjo

- Ex: Buscando 15



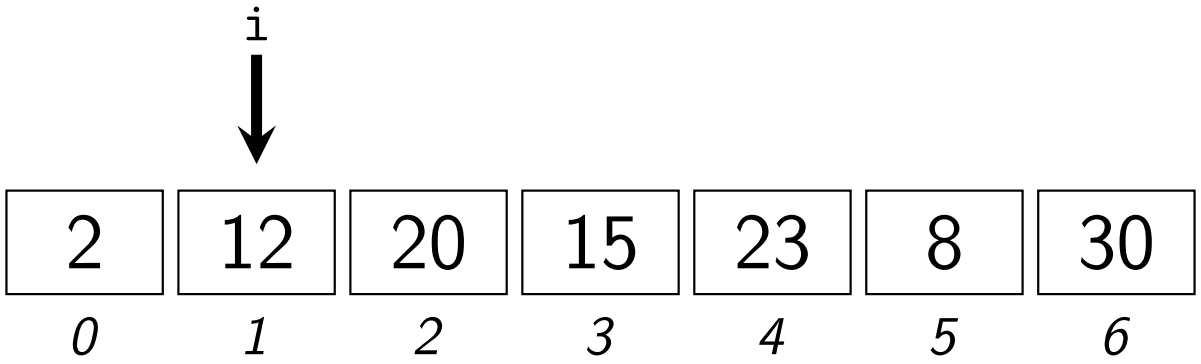
Busca em Arranjo

- Ex: Buscando 15



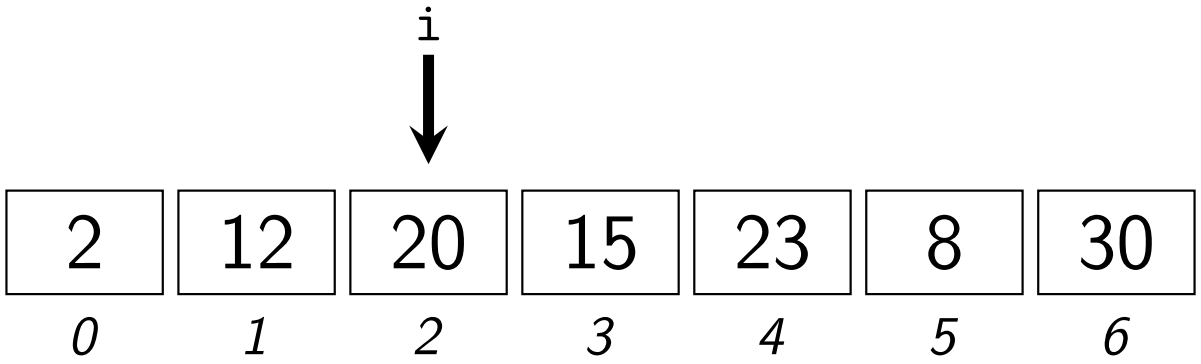
Busca em Arranjo

- Ex: Buscando 15



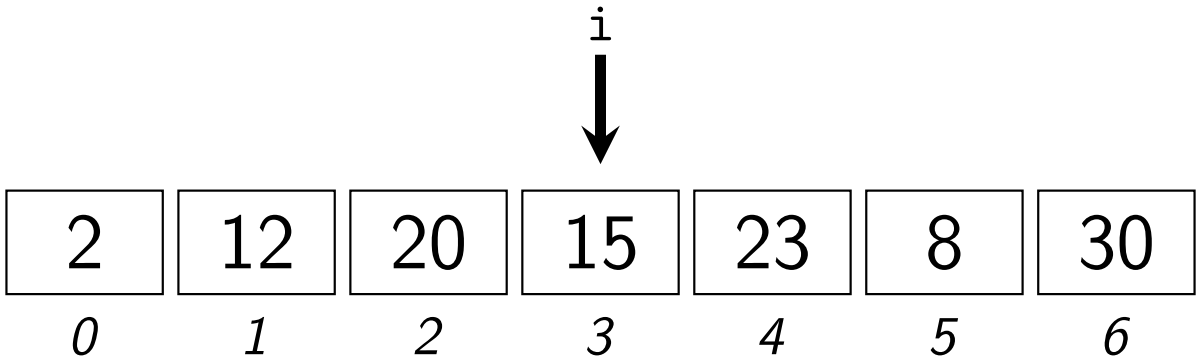
Busca em Arranjo

- Ex: Buscando 15



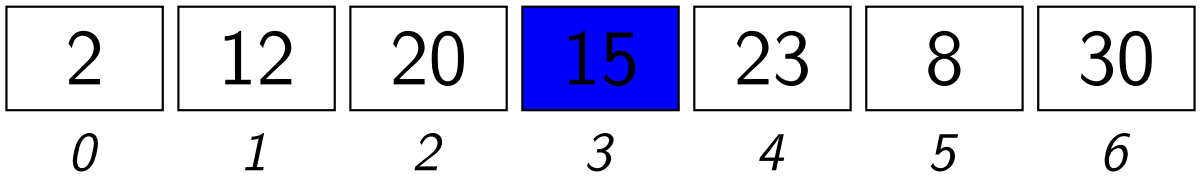
Busca em Arranjo

- Ex: Buscando 15



Busca em Arranjo

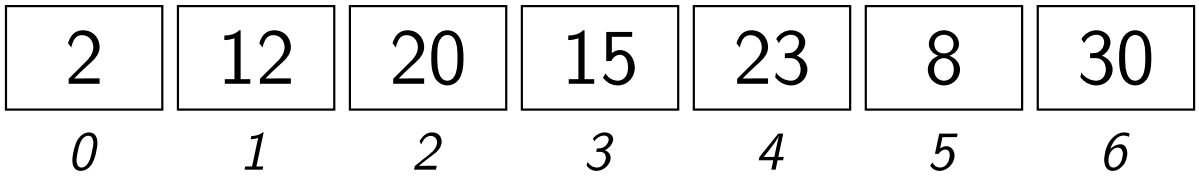
- Ex: Buscando 15



Encontrou! Índice 3 (quarta posição).

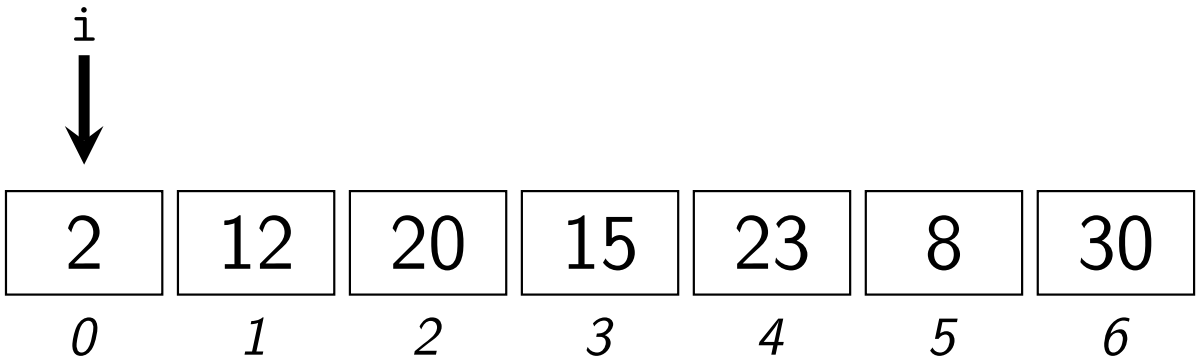
Busca em Arranjo

- E o 16?



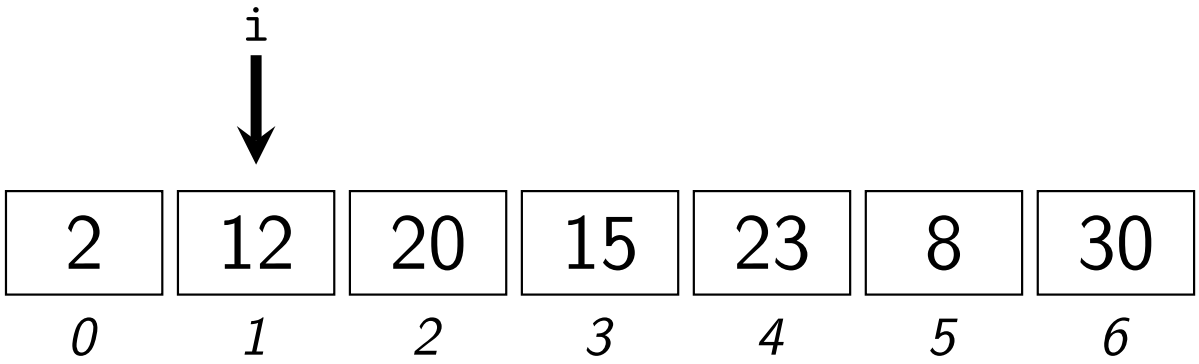
Busca em Arranjo

- E o 16?



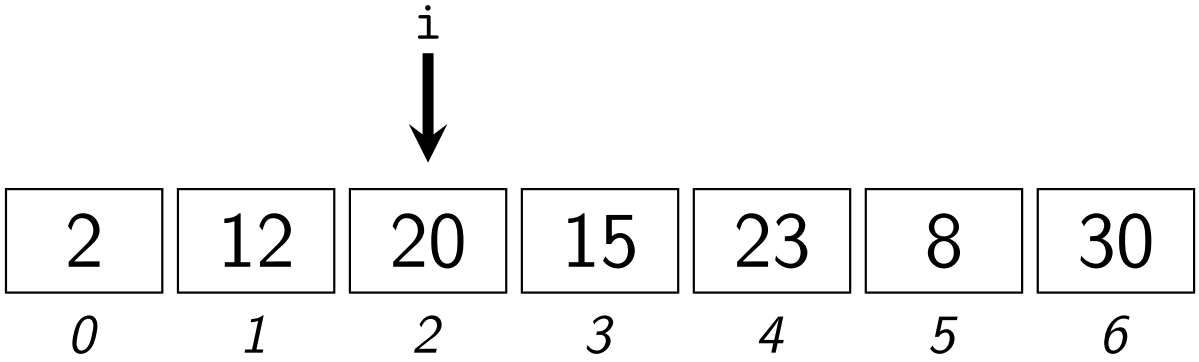
Busca em Arranjo

- E o 16?



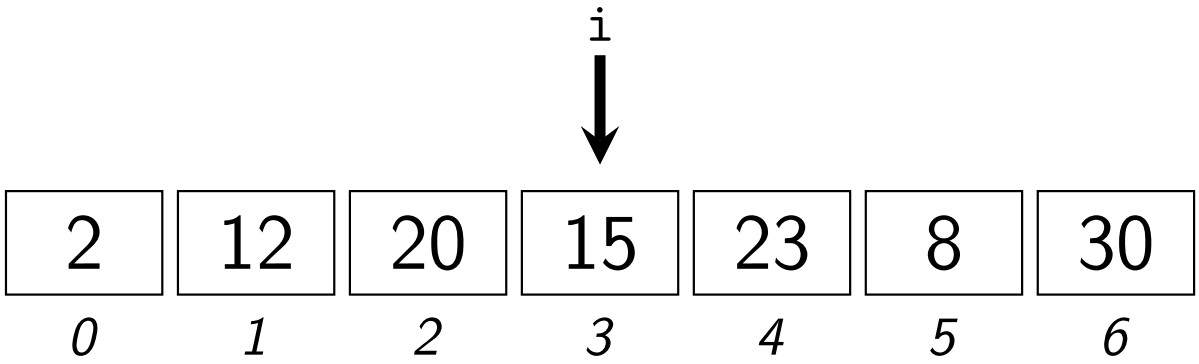
Busca em Arranjo

- E o 16?



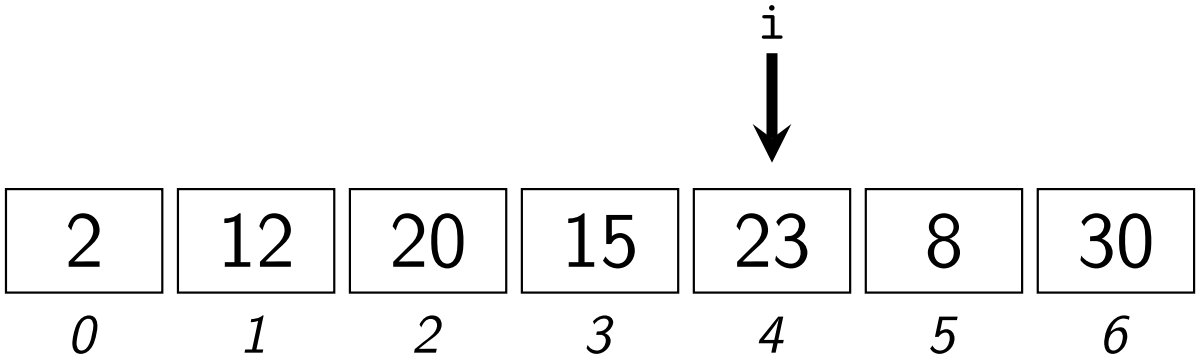
Busca em Arranjo

- E o 16?



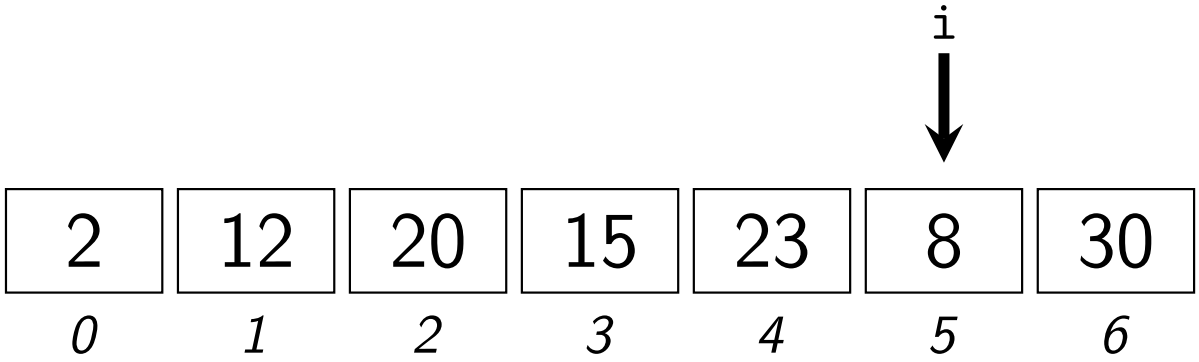
Busca em Arranjo

- E o 16?



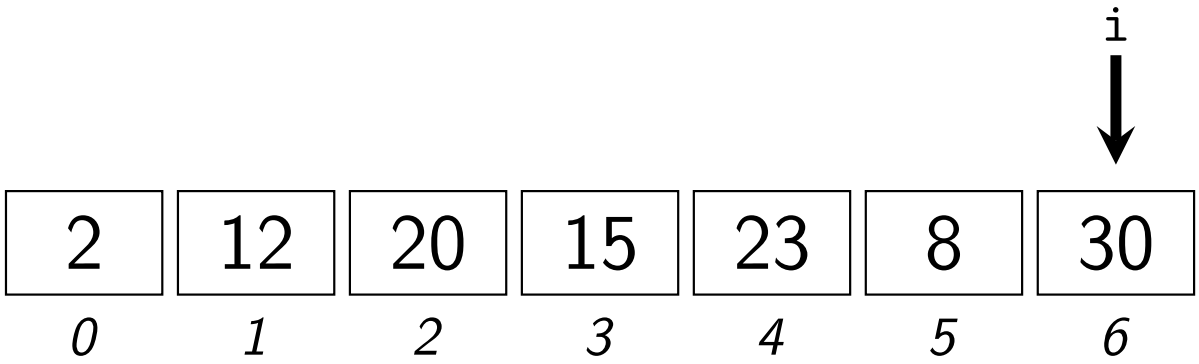
Busca em Arranjo

- E o 16?



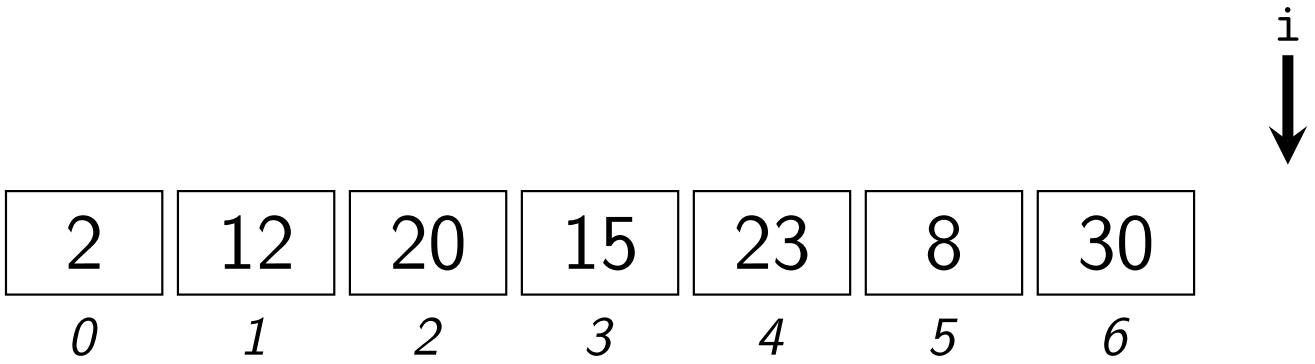
Busca em Arranjo

- E o 16?



Busca em Arranjo

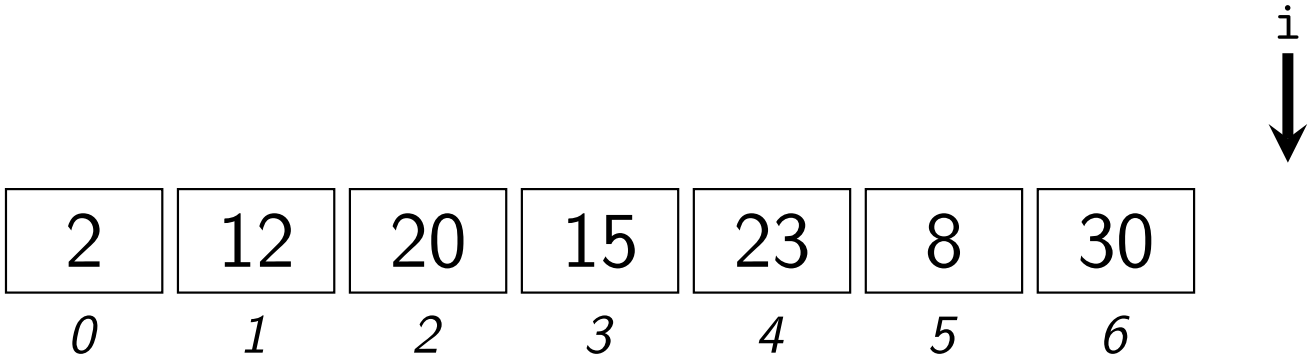
- E o 16?



Não encontrou! Como sabemos?

Busca em Arranjo

- E o 16?



Não encontrou! Como sabemos?

`i == 7`, ou seja, `i == tamanho`.

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...
- 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));
}
```

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));
}
```

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));
}
```

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));
}
```

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?
  - 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);
  }
  ...
}
```

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);
  }
  ...
}
```

Busca em Arranjo

```
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

```
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
```

Videoaula

https://www.youtube.com/watch?v=NUKflTnntcw

https://www.youtube.com/watch?v=dHou1G8iYo4
(para o Scanner)

https://www.youtube.com/watch?v=8weGr_G3Pqo
e
https://www.youtube.com/watch?v=4WYyd7MNDqQ