Arrays

Guilherme Carvalho (gvc)

Instruções para a aula

- Faça silêncio.
- Se tiver dúvidas, pergunte a mim.
- Se tiver vergonha, levante a mão e me chame.
- Evite ter vergonha, sua dúvida pode ser a do seu colega.
- Participe.
- Se tiverem dúvidas, pergunte a mim.
- Faça silêncio.
- Participe.
- E se lembre de tirar dúvida comigo.
- COMIGO!
- Beleza?

A classe Object

- Primeiro, vou falar sobre a classe básica de Java: Object.
- Todas as classes de Java são (indiretamente) Objects.
- Ela tem dois métodos muito importantes:
  - public String toString();
  - public boolean equals(Object o);

A classe Object

- O método toString() cria uma representação em String do objeto.
- O método equals() compara dois objetos para ver se são iguais.
- É uma boa prática implementá-los em toda classe que fazemos.

Exemplo

- Imaginem uma situação em que temos uma classe Conta, que tem um número associado. Cada conta tem um número e se duas Contas tiverem o mesmo número, elas são a mesma conta. Ao imprimirmos uma conta queremos que o seguinte texto apareça: “Conta de número 1234”, se o número da conta for 1234.

- Como nós fazemos?

Exemplo

```
public class Conta {
   String numero;

   public Conta(String numero) {
        this.numero = numero;
   }

   public static void main(String[] args) {
        Conta c = new Conta("1234");
        Conta c2 = new Conta("1234);
        System.out.println(c);
   }
}
```

Conta@3e25a5 Não é o que nós queremos

Exemplo

- É só escrever o toString()!

```
...
...
public String toString() {
  String retorno = "Conta de número " + this.numero;
  return retorno;
}

public static void main(String[] args) {
  Conta c = new Conta("1234");
  Conta c2 = new Conta("1234");
  System.out.println(c); Conta de número 1234
}
...
```

Exemplo

- E para que duas contas com mesmo número sejam iguais?

```
public static void main(String[] args) {
  Conta c = new Conta("1234");
  Conta c2 = new Conta("1234");
  System.out.println(c.equals(c2));
}
```

False

Exemplo

- É só escrever o equals!

```
public boolean equals(Conta c) {
  boolean retorno = this.numero.equals(c.getNumero());
  return retorno;
}

public static void main(String[] args) {
  Conta c = new Conta(“1234”);
  Conta c2 = new Conta(“1234);
  System.out.println(c.equals(c2)); True
}
```

Arrays

- O que é um array?
  - É uma sequência numerada de itens do mesmo tipo, que tem tamanho fixo e pode ser acessada por um índice.

Arrays

- Quando devem ser usados?
  - Quando temos um grande número de variáveis com o mesmo significado para o programa.
- Quando não devem ser usados?
  - Quando os elementos do array não tem um significado comum.

Arrays

- Como declarar?

```
Tipo[] nome;
```

- Explicando:
  - `Tipo` é o tipo do que vai ser guardado no array.
  - `[]` serve para dizer para a JVM que será um array.
  - `nome` é o nome pelo qual você 'chamará' o array.

Arrays

- Como instanciar?

```
nome = new Tipo[tamanho];
```

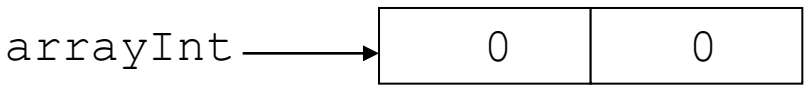
- Explicando:
  - `nome` é o nome, lembra?
  - `Tipo` é o tipo do array.
  - `tamanho` é o tamanho do array.
  - E o `new`?

Arrays

- Ao instanciarmos um array, cada posição sua vai conter o valor default do tipo do array.

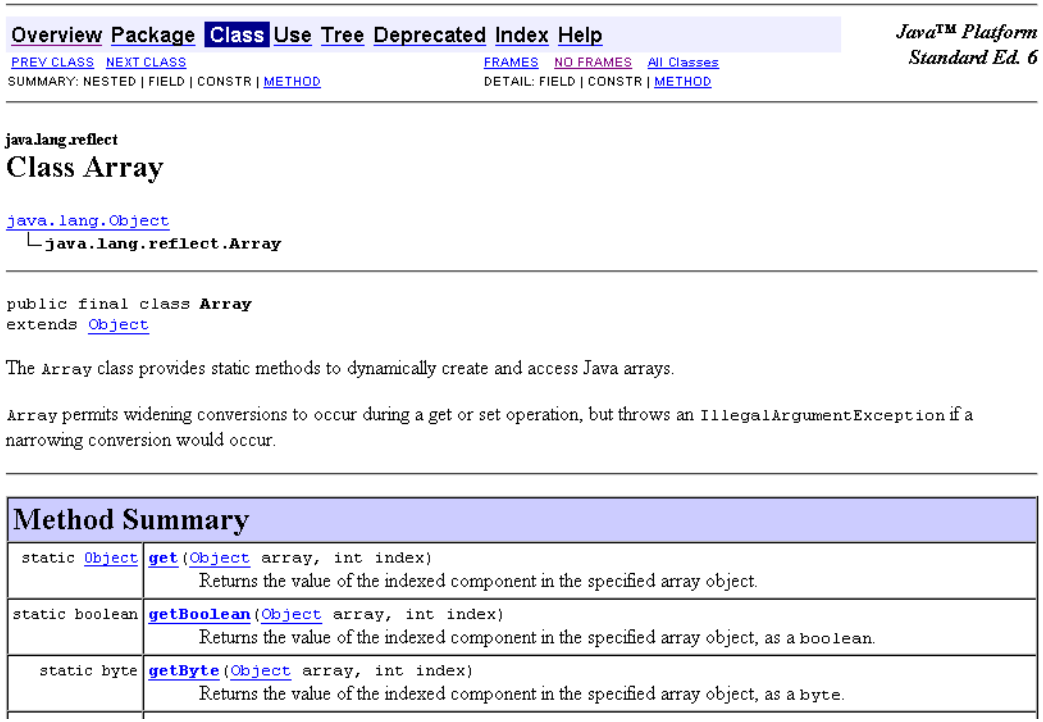
```
String[] array = new String[2];
```


```
int[] arrayInt = new int[2];
```



Arrays

- Em Java, arrays são objetos.
- Mas eles têm operações especiais, como a de acesso a um determinado elemento.



Overview Package Class Use Tree Deprecated Index Help

Java™ Platform Standard Ed. 6

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

java.lang.reflect

Class Array

java.lang.Object
  └ **java.lang.reflect.Array**

public final class **Array**
extends Object

The Array class provides static methods to dynamically create and access Java arrays.

Array permits widening conversions to occur during a get or set operation, but throws an IllegalArgumentException if a narrowing conversion would occur.

Method Summary	
static Object	**get**(Object array, int index) Returns the value of the indexed component in the specified array object.
static boolean	**getBoolean**(Object array, int index) Returns the value of the indexed component in the specified array object, as a boolean.
static byte	**getByte**(Object array, int index) Returns the value of the indexed component in the specified array object, as a byte.

Arrays

- Então, como nós acessamos elementos de um array?
  - Usamos o operador [ ], passando um índice.

```
int[] array = new int[5];
int x = array[1];
```

  - Vale lembrar que os índices no array começam em `0` e vão até `length - 1`

Loops

- Existe um loop que só funciona para objetos que sejam coleções, é chamado de for each.

```
int[] array = new int[5];
for(int x : array) {
  System.out.println("x = " + x);
}
```

- Não usem hoje!

Coleções

- Existem momentos em que não queremos nos preocupar com a implementação de uma coleção. Simplesmente queremos que um objeto possa ser guardado e buscado posteriormente.

Coleções

- Para termos esse poder temos que ter um trabalho inicial. Devemos implementar nossa própria coleção.
- O que esperamos de uma boa coleção?
  - Um método de inserção
  - Um método de remoção
  - Um método de busca

Coleções

- O que precisaremos para a inserção?

```
Conta[] contas;
int tamanho;
int proximaPosicaoVaga;


void inserir(Conta c) {
  if(proximaPosicaoVaga == tamanho) {
       duplicarArray();
  }
  contas[proximaPosicaoVaga] = c;
  proximaPosicaoVaga = proximaPosicaoVaga + 1;
}
```

Coleções

- E para duplicar o array?
  - Criamos um outro array com o dobro do tamanho.
  - Copiamos o conteúdo do array cheio para o array novo.
  - Multiplicamos o atributo `tamanho` por 2.
  - Atualizamos a referência do array cheio.

Coleções

- Para a remoção, devemos primeiro buscar um índice.

```
int buscarIndice(Conta c) {
  int indice = 0;
  boolean achou = false;

  while(!achou && indice < proximaPosicaoVaga) {
       if(c.equals(contas[indice])) {
              achou = true;
       } else {
              indice = indice + 1;
       }
  }
  return indice;
}
```

Coleções

- Agora sim, a remoção:
  - Nós não podemos simplesmente apagar a posição.
  - O array ficaria todo ‘furado’.
  - O que fazer?

```
void remover(Conta c) {
  int indice = buscarIndice(c);
  contas[indice] = contas[proximaPosicaoVaga – 1];
  proximaPosicaoVaga = proximaPosicaoVaga -1;
}
```

Coleções

- Agora, para buscarmos um objeto é muito fácil:

```
Conta buscar(Conta c) {
  int indice = buscarIndice(c);

  return contas[indice];
}
```

E agora...

A miniprova!