

LPOO I - JAVA

Construtores

CONSTRUTORES

Quando usamos a palavra chave new, estamos construindo um objeto. Sempre quando o new é chamado, ele executa o construtor da classe. O construtor da classe é um bloco declarado com o mesmo nome que a classe:

CONSTRUTORES

```
class Conta {  
    int numero;  
    String titular;  
    double saldo;  
    double limite;  
    // construtor  
    Conta() {  
        System.out.println("Construindo uma conta.");  
    }  
    // ..  
}
```

CONSTRUTORES

```
Conta c = new Conta();
```

A mensagem “construindo uma conta” aparecerá. É como uma rotina de inicialização que é chamada sempre que um novo objeto é criado. Um construtor pode parecer, mas não é um método.

CONSTRUTOR DEFAULT

O CONSTRUTOR DEFAULT

Até agora, as nossas classes não possuíam nenhum construtor. Então como é que era possível dar `new`, se todo `new` chama um construtor obrigatoriamente?

Quando você não declara nenhum construtor na sua classe, o Java cria um para você. Esse construtor é o **construtor default**, ele não recebe nenhum argumento e o corpo dele é vazio.

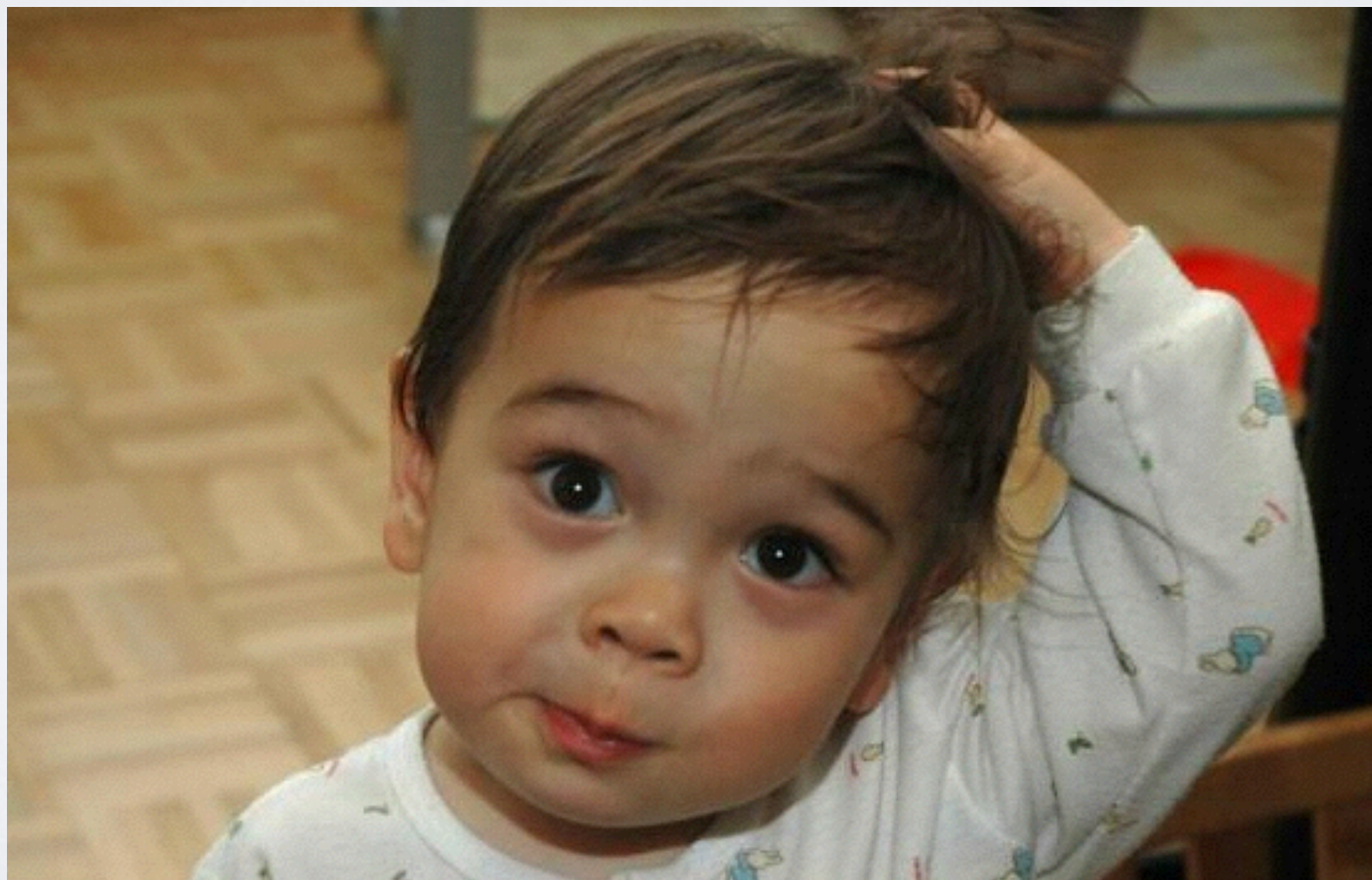
A partir do momento que você declara um construtor, o construtor default não é mais fornecido.

CONSTRUTOR PADRÃO

```
class Conta {  
    int numero;  
    String titular;  
    double saldo;  
    double limite;  
    // construtor  
    public Conta (int numero) {  
        this.numero= numero;  
    }  
  
    public Conta () {  
        super ();  
    }  
}
```

A NECESSIDADE DE UM CONSTRUTOR

Tudo estava funcionando até agora. Para que utilizamos um construtor?



A NECESSIDADE DE UM CONSTRUTOR

A ideia é bem simples. Se toda conta precisa de um titular, como obrigar todos os objetos que forem criados a ter um valor desse tipo?

Basta criar um único construtor que recebe essa String!

O construtor se resume a isso! Dar possibilidades ou obrigar o usuário de uma classe a passar argumentos para o objeto durante o processo de criação do mesmo.

Por exemplo, não podemos abrir um arquivo para leitura sem dizer qual é o nome do arquivo que desejamos ler! Portanto, nada mais natural que passar uma String representando o nome de um arquivo na hora de criar um objeto do tipo de leitura de arquivo, e que isso seja obrigatório.

Você pode ter mais de um construtor na sua classe e, no momento do new, o construtor apropriado será escolhido.

CONSTRUTOR É UM MÉTODO?



NÃO

Construtor é um construtor!
Tanto que:
Não possui retorno!
E só usando na criação do objeto

QUANTOS CONSTRUTORES?

```
class Conta {  
    int numero;  
    String titular;  
    double saldo;  
    double limite;  
    public Conta (int numero) {  
        this.numero= numero;  
    }  
    public Conta (int numero, String titular) {  
        this.numero= numero;  
        this.titular=titular;  
    }  
    public int criaConta(){  
        //faz alguma coisa  
        return 1;  
    }  
    public Conta add(){  
        ///faz alguma coisa....  
        return null;  
    }  
}
```



O LULA NÃÃO SABE NADA DE OO

```
class Conta {  
    int numero;  
    String titular;  
    double saldo;  
    double limite;  
    public Conta (int numero) {  
        this.numero= numero;  
    }  
    public Conta (int numero, String titular) {  
        this.numero= numero;  
        this.titular=titular;  
    }  
    public int criaConta(){  
        //faz alguma coisa  
        return 1;  
    }  
    public Conta add(){  
        ///faz alguma coisa....  
        return null;  
    }  
}
```

Dois
construtores

Dois
Métodos

ASSINATURA DE MÉTODOS

The screenshot shows the Eclipse IDE with a Java file named 'Ex.java'. The code defines a class 'Conta' with the following methods:

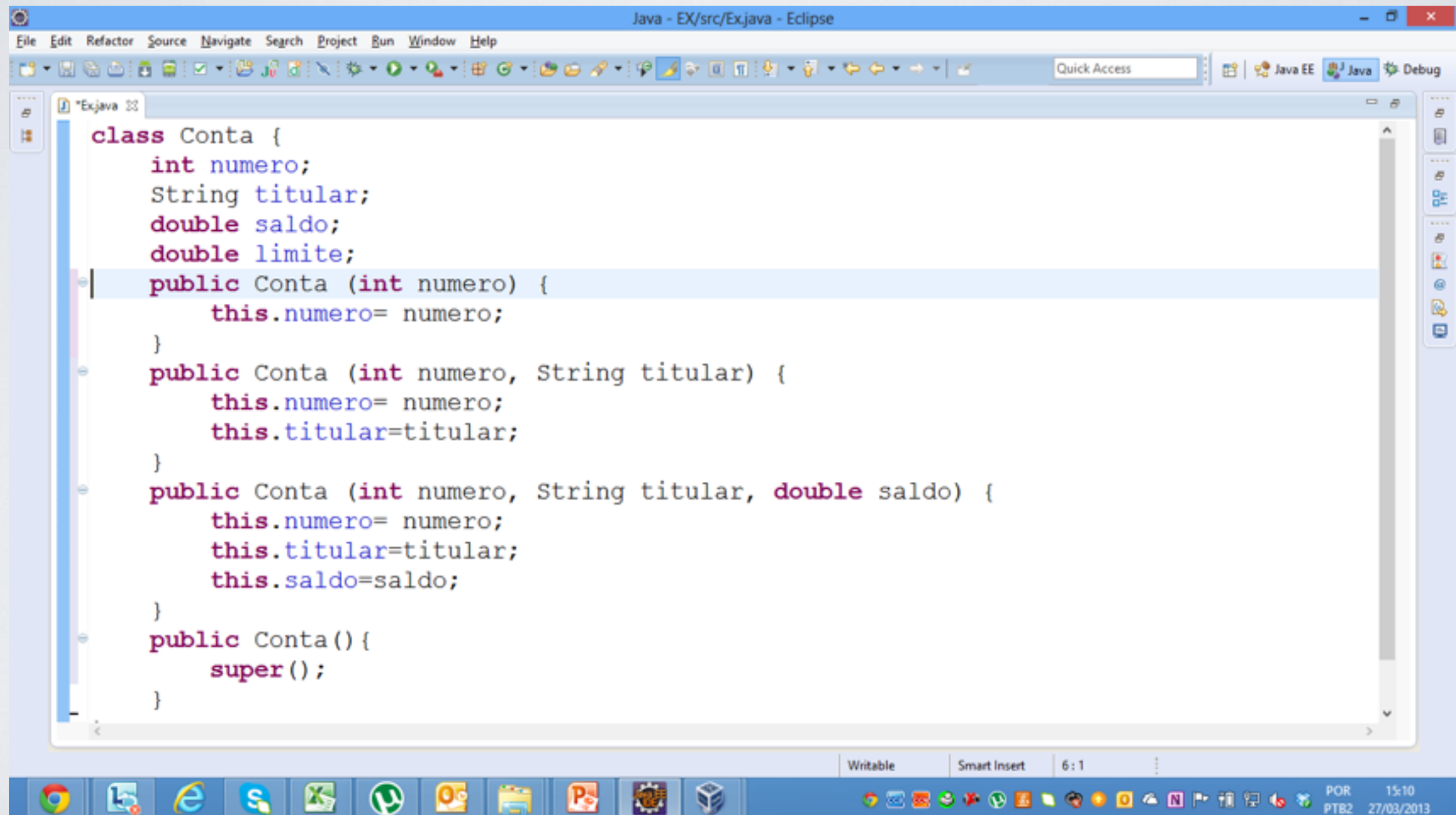
```
class Conta {  
    int numero;  
    String titular;  
    double saldo;  
    double limite;  
    public Conta (int numero) {  
        this.numero= numero;  
    }  
    public Conta (int numero, String titular) {  
        this.numero= numero;  
        this.titular=titular;  
    }  
    public Conta add(int i){  
        //faz alguma coisa  
        return null;  
    }  
    public Conta add(){  
        ///faz alguma coisa....  
        return null;  
    }  
}
```

Annotations on the right side of the image point to specific parts of the code:

- Assinatura**: Points to the signature of the second constructor, `public Conta (int numero, String titular)`.
- Nome**: Points to the name of the first `add` method, `add(int i)`.

The bottom of the image shows the Windows taskbar with various application icons and the system clock indicating 16:44 on 27/03/2013.

POLIMORFISMO - SOBRECARGA



```
Java - EX/src/Ex.java - Eclipse
File Edit Refactor Source Navigate Search Project Run Window Help
Quick Access
Java EE Java Debug

class Conta {
    int numero;
    String titular;
    double saldo;
    double limite;
    public Conta (int numero) {
        this.numero= numero;
    }
    public Conta (int numero, String titular) {
        this.numero= numero;
        this.titular=titular;
    }
    public Conta (int numero, String titular, double saldo) {
        this.numero= numero;
        this.titular=titular;
        this.saldo=saldo;
    }
    public Conta() {
        super();
    }
}
```

Writable Smart Insert 6:1

POR PTB2 15:10 27/03/2013

POLIMORFISMO DE MÉTODOS

SOBRECARGA

```
public Conta add(int i, int a) {  
    //faz alguma coisa  
    return null;  
}  
public int add(int i) {  
    ///faz alguma coisa....  
    return 1;  
}  
public int add(int i, char b) {  
    ///faz alguma coisa....  
    return 1;  
}
```

ATIVIDADE PROPOSTA

Criar 3 construtores para nossos livros.

O primeiro realiza a leitura de todos os dados no construtor.

o Segundo recebe por parâmetro o autor, e faz a leitura no construtor dos demais dados do livro.

o terceiro recebe os dados do livro por parâmetro e um autor.

ARRAYLIST

ARRAYLISTS

O Java, por padrão, possui uma série de recursos prontos (APIs) para que possamos tratar de estrutura de dados, também chamados de coleções (collections).

Podemos dizer que ArrayList é uma classe para coleções. Uma classe genérica (generic classes), para ser mais exato.

Coleções mesmo, de qualquer tipo de 'coisa', desde que seja um objeto.

Você pode criar seus objetos - através de uma classe - e agrupá-los através de ArrayList e realizar, nessa coleção, várias operações, como: adicionar e retirar elementos, ordená-los, procurar por um elemento específico, apagar um elemento específico, limpar o ArrayList dentre outras possibilidades.

COMO DECLARAR E USAR ARRAYLIST EM JAVA

```
ArrayList< Objeto > nomeDoArrayList = new ArrayList< Objeto >();
```

No exemplo a seguir, vamos usar um ArrayList de String

```
ArrayList<String> nomes = new ArrayList<String> ();
```

API

<http://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html>

Method Summary

Methods

Modifier and Type	Method and Description
boolean	add(E e) Appends the specified element to the end of this list.
void	add(int index, E element) Inserts the specified element at the specified position in this list.
boolean	addAll(Collection<? extends E> c) Appends all of the elements in the specified collection to the end of this list, in the order that they are returned by the specified collection's Iterator.
boolean	addAll(int index, Collection<? extends E> c) Inserts all of the elements in the specified collection into this list, starting at the specified position.
void	clear() Removes all of the elements from this list.
Object	clone() Returns a shallow copy of this ArrayList instance.
boolean	contains(Object o) Returns true if this list contains the specified element.

ALGUNS EXEMPLOS

```
ArrayList<String> lista = new ArrayList<String>();
```

```
lista.add("ramon");  
lista.add("Fulano");
```

```
lista.add(0, "Raquel");  
lista.add(2, "Beltrano");
```

```
System.out.println(lista.size());  
lista.remove(0);  
lista.remove("Beltrano");
```

```
System.out.println(lista.get(0));
```

```
System.out.println(lista.indexOf("Fulano"));
```


ATIVIDADE

Criar um `arrayList` para armazenar livro, crie um menu que seja possível adicionar livros, remover, listar e verificar se um livro já existe na lista.