Curso
Programação Orientada a Objetos com Java

Capítulo: Construtores, palavra this, sobrecarga, encapsulamento

http://educandoweb.com.br

Prof. Dr. Nelio Alves

Construtores

http://educandoweb.com.br

Prof. Dr. Nelio Alves

Construtor

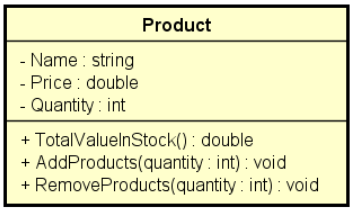
- É uma operação especial da classe, que executa no momento da instanciação do objeto (Comando New)
- Usos comuns:
  - Iniciar valores dos atributos
  - Permitir ou obrigar que o objeto receba dados / dependências no momento de sua instanciação (injeção de dependência)
- Se um construtor customizado não for especificado, a classe disponibiliza o construtor padrão:

```
Product p = new Product();
```

- É possível especificar mais de um construtor na mesma classe (sobrecarga)

Problema exemplo

```
Enter product data:
Name: TV
Price: 900.00
Quantity in stock: 10

Product data: TV, $ 900.00, 10 units, Total: $ 9000.00

Enter the number of products to be added in stock: 5

Updated data: TV, $ 900.00, 15 units, Total: $ 13500.00

Enter the number of products to be removed from stock: 3

Updated data: TV, $ 900.00, 12 units, Total: $ 10800.00
```



```java
package application;

import java.util.Locale;
import java.util.Scanner;

import entities.Product;

public class Program {
	public static void main(String[] args) {
		Locale.setDefault(Locale.US);
		Scanner sc = new Scanner(System.in);

		Product product = new Product();
		System.out.println("Enter product data: ");
		System.out.print("Name: ");
		product.name = sc.nextLine();
		System.out.print("Price: ");
		product.price = sc.nextDouble();
		System.out.print("Quantity in stock: ");
		product.quantity = sc.nextInt();

		System.out.println();
		System.out.println("Product data: " + product);

		System.out.println();
		System.out.print("Enter the number of products to be added in stock: ");
		int quantity = sc.nextInt();
		product.addProducts(quantity);

		System.out.println();
		System.out.println("Updated data: " + product);

		System.out.println();
		System.out.print("Enter the number of products to be removed from stock: ");
		quantity = sc.nextInt();
		product.removeProducts(quantity);

		System.out.println();
		System.out.println("Updated data: " + product);

		sc.close();
	}
}
```

```java
package entities;

public class Product {

	public String name;
	public double price;
	public int quantity;

	public double totalValueInStock() {
		return price * quantity;
	}

	public void addProducts(int quantity) {
		this.quantity += quantity;
	}

	public void removeProducts(int quantity) {
		this.quantity -= quantity;
	}

	public String toString() {
		return name
			+ ", $ "
			+ String.format("%.2f", price)
			+ ", "
			+ quantity
			+ " units, Total: $ "
			+ String.format("%.2f", totalValueInStock());
	}
}
```

Proposta de melhoria

Quando executamos o comando abaixo, instanciamos um produto "**product**" com seus atributos “vazios”:

```
product = new Product();
```

Entretanto, faz sentido um produto que não tem nome? Faz sentido um produto que não tem preço?

Com o intuito de evitar a existência de produtos sem nome e sem preço, é possível fazer com que seja “obrigatória” a iniciação desses valores?

```java
package entities;

public class Product {

    public String name;
    public double price;
    public int quantity;

    public Product(String name, double price, int quantity) {
        this.name = name;
        this.price = price;
        this.quantity = quantity;
    }
    (...)
```

```java
System.out.println("Enter product data: ");
System.out.print("Name: ");
String name = sc.nextLine();
System.out.print("Price: ");
double price = sc.nextDouble();
System.out.print("Quantity in stock: ");
int quantity = sc.nextInt();
Product product = new Product(name, price, quantity);
```

Palavra this

http://educandoweb.com.br

Prof. Dr. Nelio Alves

Palavra this

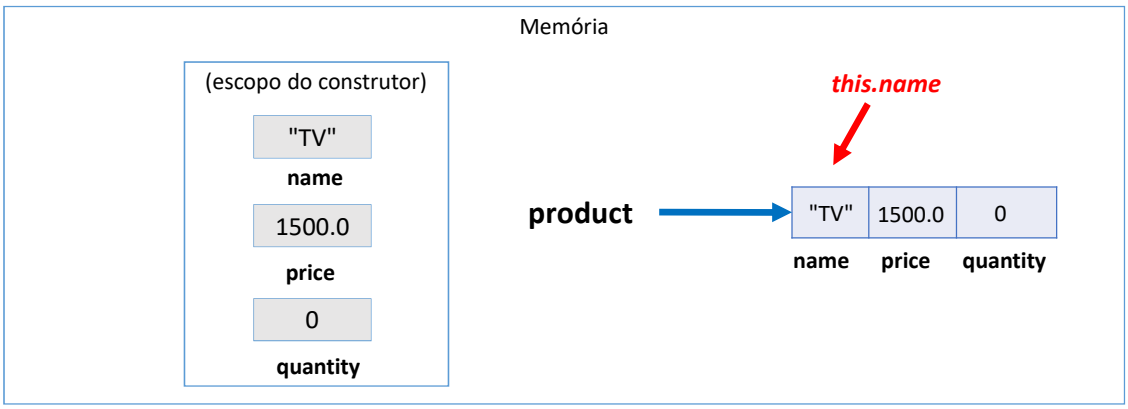
- É uma referência para o próprio objeto

- Usos comuns:
  - Diferenciar atributos de variáveis locais
  - Passar o próprio objeto como argumento na chamada de um método ou construtor

Diferenciar atributos de variáveis locais

```
Product product = new Product("TV", 1500.0, 0);
```

```
public Product(String name, double price, int quantity) {
    this.name = name;
    this.price = price;
    this.quantity = quantity;
}
```



Passar o próprio objeto como argumento na chamada de um método ou construtor

```
public class ChessMatch {

    (...)

    placeNewPiece('e', 1, new King(board, Color.WHITE, this));

    (...)
```

Sobrecarga

http://educandoweb.com.br

Prof. Dr. Nelio Alves

Sobrecarga

- É um recurso que uma classe possui de oferecer mais de uma operação com o mesmo nome, porém com diferentes listas de parâmetros.

Proposta de melhoria

- Vamos criar um construtor opcional, o qual recebe apenas nome e preço do produto. A quantidade em estoque deste novo produto, por padrão, deverá então ser iniciada com o valor zero.

- Nota: é possível também incluir um construtor padrão

```
package entities;

public class Product {

    public String name;
    public double price;
    public int quantity;

    public Product() {
    }

    public Product(String name, double price, int quantity) {
        this.name = name;
        this.price = price;
        this.quantity = quantity;
    }

    public Product(String name, double price) {
        this.name = name;
        this.price = price;
    }
    (...)
```

Encapsulamento

http://educandoweb.com.br

Prof. Dr. Nelio Alves

Encapsulamento

- É um princípio que consiste em esconder detalhes de implementação de uma classe, expondo apenas operações seguras e que mantenham os objetos em um estado consistente.
- Regra de ouro: o objeto deve sempre estar em um estado consistente, e a própria classe deve garantir isso.

Analogia:

Regra geral básica

- Um objeto **NÃO** deve expor nenhum atributo (modificador de acesso **private**)
- Os atributos devem ser acessados por meio de métodos get e set
  - Padrão JavaBeans: https://en.wikipedia.org/wiki/JavaBeans

Padrão para implementação de getters e setters

```
private String name;
private double price;

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public double getPrice() {
    return price;
}

public void setPrice(double price) {
    this.price = price;
}
```

```java
package entities;

public class Product {

	private String name;
	private double price;
	private int quantity;

	public Product() {
	}

	public Product(String name, double price, int quantity) {
		this.name = name;
		this.price = price;
		this.quantity = quantity;
	}

	public Product(String name, double price) {
		this.name = name;
		this.price = price;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public double getPrice() {
		return price;
	}

	public void setPrice(double price) {
		this.price = price;
	}

	public int getQuantity() {
		return quantity;
	}

	(...)
```

Gerando automaticamente construtores, getters e setters com Eclipse

http://educandoweb.com.br

Prof. Dr. Nelio Alves

Comandos

- Botão direito -> Source -> Generate Constructor using Fields
- Botão direito -> Source -> Generate Getters and Setters

Modificadores de acesso

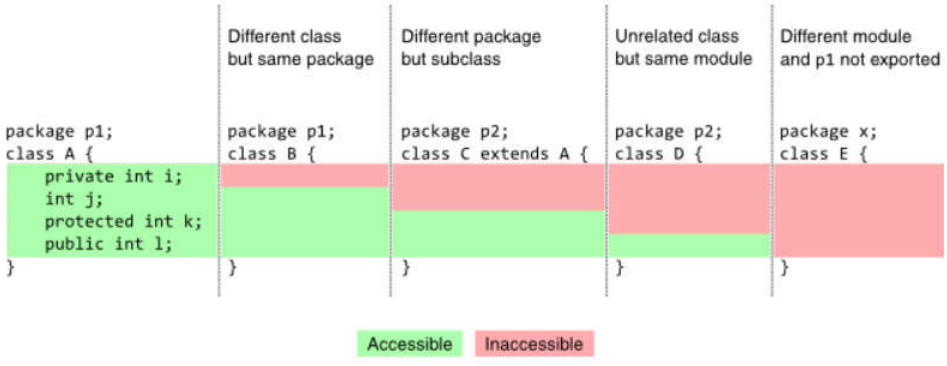
http://educandoweb.com.br

Prof. Dr. Nelio Alves

Modificadores de acesso

- https://docs.oracle.com/javase/tutorial/java/javaOO/accesscontrol.html
- **private**: o membro só pode ser acessado na **própria classe**
- (nada): o membro só pode ser acessado nas classes do **mesmo pacote**
- **protected**: o membro só pode ser acessado no **mesmo pacote**, bem como em **subclasses de pacotes diferentes**
- **public**: o membro é acessado por todas classes (ao menos que ele resida em um módulo diferente que não exporte o pacote onde ele está)

https://stackoverflow.com/questions/215497/in-java-difference-between-package-private-public-protected-and-private



Exercício de fixação

http://educandoweb.com.br

Prof. Dr. Nelio Alves

Em um banco, para se cadastrar uma conta bancária, é necessário informar o número da conta, o nome do titular da conta, e o valor de depósito inicial que o titular depositou ao abrir a conta. Este valor de depósito inicial, entretanto, é opcional, ou seja: se o titular não tiver dinheiro a depositar no momento de abrir sua conta, o depósito inicial não será feito e o saldo inicial da conta será, naturalmente, zero.

Importante: uma vez que uma conta bancária foi aberta, o número da conta nunca poderá ser alterado. Já o nome do titular pode ser alterado (pois uma pessoa pode mudar de nome por ocasião de casamento, por exemplo).

Por fim, o saldo da conta não pode ser alterado livremente. É preciso haver um mecanismo para proteger isso. O saldo só aumenta por meio de depósitos, e só diminui por meio de saques. Para cada saque realizado, o banco cobra uma taxa de $ 5.00. Nota: a conta pode ficar com saldo negativo se o saldo não for suficiente para realizar o saque e/ou pagar a taxa.

Você deve fazer um programa que realize o cadastro de uma conta, dando opção para que seja ou não informado o valor de depósito inicial. Em seguida, realizar um depósito e depois um saque, sempre mostrando os dados da conta após cada operação.

(exemplos nas próximas páginas)

EXAMPLE 1

```
Enter account number: 8532
Enter account holder: Alex Green
Is there na initial deposit (y/n)? y
Enter initial deposit value: 500.00

Account data:
Account 8532, Holder: Alex Green, Balance: $ 500.00

Enter a deposit value: 200.00
Updated account data:
Account 8532, Holder: Alex Green, Balance: $ 700.00

Enter a withdraw value: 300.00
Updated account data:
Account 8532, Holder: Alex Green, Balance: $ 395.00
```

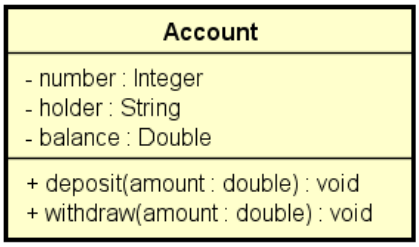
EXAMPLE 2

```
Enter account number: 7801
Enter account holder: Maria Brown
Is there na initial deposit (y/n)? n

Account data:
Account 7801, Holder: Maria Brown, Balance: $ 0.00

Enter a deposit value: 200.00
Updated account data:
Account 7801, Holder: Maria Brown, Balance: $ 200.00

Enter a withdraw value: 198.00
Updated account data:
Account 7801, Holder: Maria Brown, Balance: $ -3.00
```

Correção do exercício de fixação

http://educandoweb.com.br

Prof. Dr. Nelio Alves



https://github.com/acenelio/encapsulation1-java