TEMA 8 – CONSTRUTORES

Um **construtor determina as ações que podem ser executadas quando um objeto é criado** em JAVA. Ele é definido **como um método**, que deve ter o mesmo nome da classe e sem definição do tipo de retorno, **nem mesmo void**.
O construtor é **unicamente invocado no momento da criação do objeto através do operador new**. O retorno do operador new é uma referência para o objeto recém-criado.
O construtor pode receber argumentos, como qualquer método. **Usando o mecanismo de sobrecarga**, **mais de um construtor pode ser definido para uma classe**.
Um construtor é um método de grande importância dentro de uma classe Java. Ele possui algumas características próprias:

• É um método que tem o mesmo nome que a classe
• O padrão (default) não possui um valor de retorno, nem mesmo void.
• Pode conter parâmetros (argumentos)
• Toda classe deve possuir ao menos um construtor
• Se nenhum construtor for definido em uma classe, a JVM irá prover um construtor padrão, sem argumentos, conhecido como default.

Ao ser criada, uma classe inicializa seus atributos da seguinte maneira:

• Tipos primitivos numéricos (int, short, byte, long, float, double) são inicializados **com valor 0 (Zero)**
• Tipos primitivos booleanos (boolean) **são inicializados com valor false**
• Tipos primitivos caracteres (char) **são inicializar com valor ” (vazio)**
• Objetos são inicializados **como null**.

O que é um Construtor?
Um construtor em Java é um bloco de código que inicializa um objeto recém-criado. Ele tem o mesmo nome da classe e parece-se com um método, mas não tem um tipo de retorno.

Por que precisamos de um Construtor?
Quando criamos um objeto, muitas vezes queremos configurá-lo com valores iniciais. Os construtores permitem que façamos isso. Eles garantem que o objeto esteja em um estado inicial válido assim que ele for criado.

Exemplos no Código

Construtor padrão:

```
public Car() {
    this.color = "Unknown";
    this.year = 0;
}
```

Quando criamos um objeto e não fornecemos informações, o construtor padrão é invocado. Ele define a cor como "Unknown" e o ano como 0.

Construtor sobrecarregado:

```
public Car(String color, int year) {
    this.color = color;
    this.year = year;
}
```

Este é um exemplo de sobrecarga de construtor. Quando sabemos a cor e o ano do carro no momento da criação, podemos fornecer essas informações e este construtor será chamado.

Como os Construtores são usados?

No programa MainCar, temos dois objetos sendo criados:

Objeto defaultCar:

Car defaultCar = new Car();

Neste caso, o construtor padrão é chamado porque não fornecemos informações. Portanto, as propriedades desse objeto serão:

Color: Unknown
Year: 0

Objeto customCar:

Car customCar = new Car("Red", 2020);

Aqui, fornecemos cor e ano, então o construtor sobrecarregado é chamado. As propriedades desse objeto serão:

Color: Red
Year: 2020

Abaixo os códigos de "Car" e "MainCar":

public class Car {

```
    private String color;
    private int year;

    // Construtor padrão
    public Car() {
        this.color = "Unknown";
        this.year = 0;
    }

    // Construtor sobrecarregado
    public Car(String color, int year) {
        this.color = color;
        this.year = year;
    }

    // Getter para color
    public String getColor() {
        return this.color;
    }

    // Getter para year
    public int getYear() {
        return this.year;
    }
```

```
}


public class MainCar {
    public static void main(String[] args) {
        // Criando objetos da classe Car
        Car defaultCar = new Car(); // Usará o construtor padrão
        Car customCar = new Car("Red", 2020); // Usará o construtor sobrecarregado

        // Exibindo informações dos carros
        System.out.println("Default Car:");
        System.out.println("Color: " + defaultCar.getColor());
        System.out.println("Year: " + defaultCar.getYear());

        System.out.println("\nCustom Car:");
        System.out.println("Color: " + customCar.getColor());
        System.out.println("Year: " + customCar.getYear());
    }
}
```

Resultado de MainCar:

```
Default Car:
Color: Unknown
Year: 0

Custom Car:
Color: Red
Year: 2020
```

Agora vamos usar o conceito de uma classe "Book" (Livro) para demonstrar outro exemplo de construtores.

Classe Book:
Imagine que queremos modelar um livro. Um livro tem um título, um autor e um número de páginas. Aqui está como poderíamos representar isso:

```
public class Book {

    private String title;
    private String author;
    private int pages;

    // Construtor padrão
    public Book() {
        this.title = "Unknown";
        this.author = "Unknown";
        this.pages = 0;
    }

    // Construtor sobrecarregado
    public Book(String title, String author, int pages) {
        this.title = title;
        this.author = author;
        this.pages = pages;
```

```
    }

    // Getter para title
    public String getTitle() {
        return this.title;
    }

    // Getter para author
    public String getAuthor() {
        return this.author;
    }

    // Getter para pages
    public int getPages() {
        return this.pages;
    }
}
```

Classe MainBook:

A classe principal para criar objetos da classe Book e mostrar seus detalhes:

```
public class MainBook {
    public static void main(String[] args) {
        // Criando objetos da classe Book
        Book defaultBook = new Book(); // Usará o construtor padrão
        Book customBook = new Book("1984", "George Orwell", 328); // Usará o construtor sobrecarregado

        // Exibindo informações dos livros
        System.out.println("Default Book:");
        System.out.println("Title: " + defaultBook.getTitle());
        System.out.println("Author: " + defaultBook.getAuthor());
        System.out.println("Pages: " + defaultBook.getPages());

        System.out.println("\nCustom Book:");
        System.out.println("Title: " + customBook.getTitle());
        System.out.println("Author: " + customBook.getAuthor());
        System.out.println("Pages: " + customBook.getPages());
    }
}
```

Resultado de MainBook:

```
Default Book:
Title: Unknown
Author: Unknown
Pages: 0

Custom Book:
Title: 1984
Author: George Orwell
Pages: 328
```

Neste exemplo, você pode ver o uso do construtor padrão (sem argumentos) e do construtor sobrecarregado (com argumentos) de maneira similar ao exemplo da classe Car.

Conclusão

Construtores são fundamentais para inicializar objetos em Java. Eles garantem que um objeto comece sua vida em um estado adequado. A capacidade de sobrecarregar construtores permite flexibilidade na maneira como inicializamos objetos, proporcionando diferentes formas de criar uma instância de uma classe, dependendo das informações que temos no momento da criação.