Lógica de **P**rogramação

Unidade 11 – Método construtor

QI E SCOLAS E FACULDADES
Curso Técnico em Informática

SUMÁRIO

INTRODUÇÃO

Método construtor é um método especial responsável pela ação de construir o objeto na memória. Ele é responsável por criar o objeto e definir seu **estado inicial**.

> *O estado inicial corresponde ao valor dos atributos logo que o objeto é instanciado.*

Quando não definimos o método construtor, o Java utiliza um construtor padrão, presente em todas as classes. Este construtor cria um objeto cujo estado inicial é nulo ou vazio. Cada tipo de atributo é inicializado com um valor padrão:

- String: **null**
- double, float, int e byte: **0**
- boolean: **false**

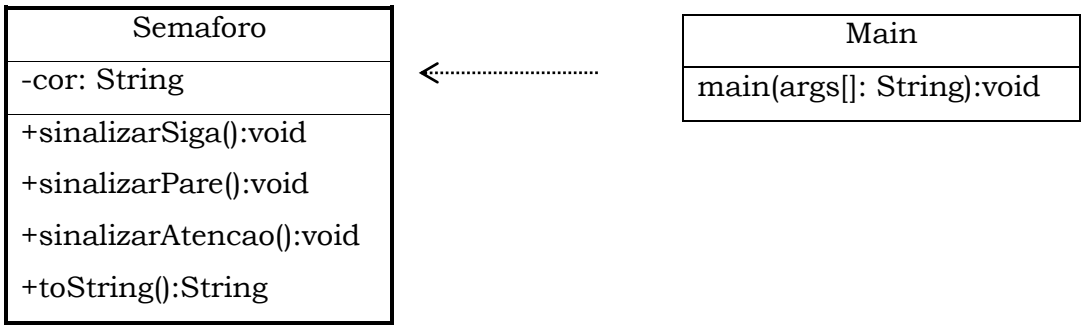
Uma classe pode ter mais de um método construtor, conforme a necessidade. O construtor é acionado quando utilizamos o comando **new** para criar um novo objeto.

CRIANDO UM MÉTODO CONSTRUTOR

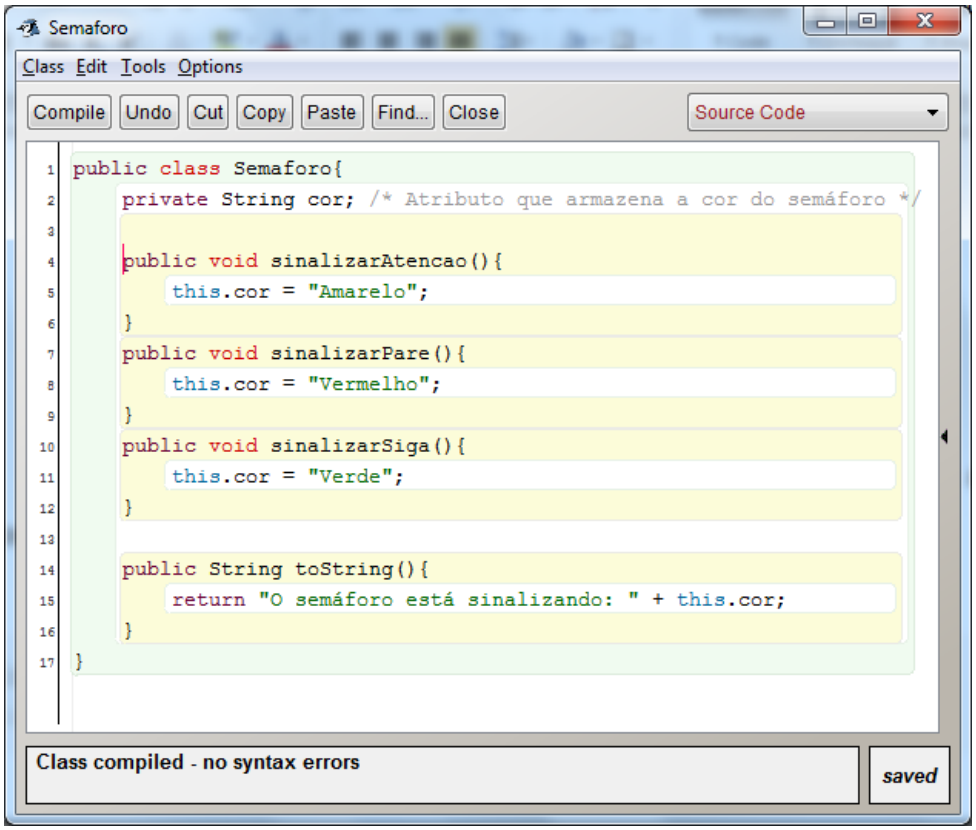
Um método construtor deve possuir exatamente o **mesmo nome da classe**, não tem tipo, e deve ter visibilidade pública. Pode ou não ter argumentos.

Construtor sem argumentos (vazio)

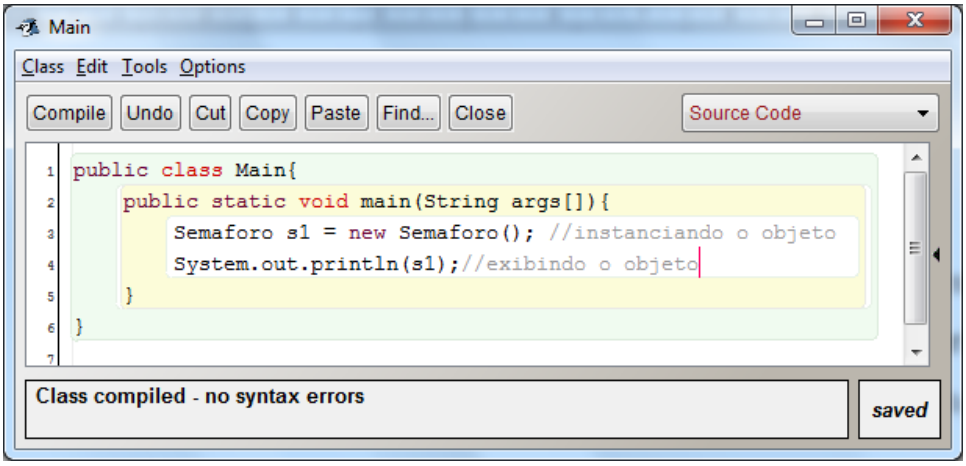
Vamos usar como exemplo um semáforo, observe então o diagrama de classe a seguir.



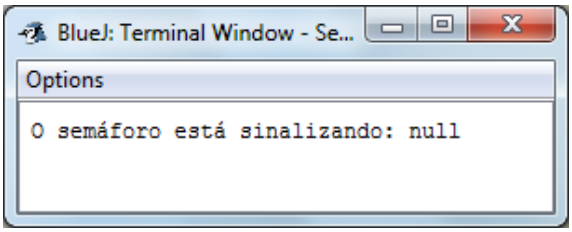
Nesta classe, temos o atributo "cor" que armazena a cor que o Semáforo mostrará. Vamos ver como fica o código da classe:



Na principal, vamos criar o objeto e exibi-lo.



No momento que instanciamos o objeto (linha 3), o construtor padrão do Java é acionado. Isto significa que o valor do atributo "**cor**", no momento que o objeto é criado, valerá "**null**", que é o valor padrão inicial para *Strings*. Se rodarmos o programa, vemos no terminal a mensagem "O semáforo está sinalizando: null".



Para que o atributo, ao invés de inicializar com "*null*", inicialize com uma das cores do semáforo, como vermelho, podemos criar um construtor e determinar que sempre que um novo objeto for criado seu valor inicial será vermelho.

Primeiramente, acrescentamos o método no diagrama de classe, e em seguida no código.

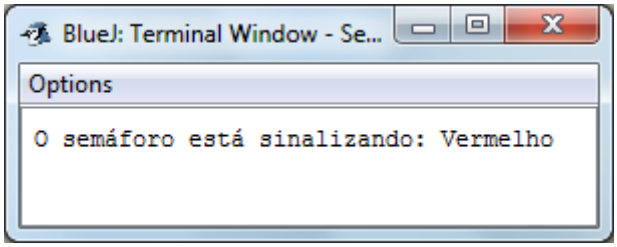
Semaforo
-cor: String
+Semaforo() +sinalizarSiga():void +sinalizarPare():void +sinalizarAtencao():void +toString():String

```
public class Semaforo{
    private String cor; /* Atributo que armazena a cor do semáforo */

    public Semaforo(){
        this.cor = "Vermelho";
    }
    public void sinalizarAtencao(){
        this.cor = "Amarelo";
    }
    public void sinalizarPare(){
        this.cor = "Vermelho";
    }
    public void sinalizarSiga(){
        this.cor = "Verde";
    }

    public String toString(){
        return "O semáforo está sinalizando: " + this.cor;
    }
}
```

Assim, ao executarmos novamente a classe Main, a saída será "O semáforo está sinalizando: Vermelho".



IMPORTANTE: O método construtor é acionado quando criamos uma instância do objeto.

IMPORTANTE 2: O método construtor, apesar de não retornar nada, não utiliza a palavra void!

Construtor com argumentos

Ao invés de definirmos no código o valor inicial de um atributo, podemos fazer com que ele seja enviado ao objeto no momento da sua criação, ou seja, ao instanciá-lo, através dos argumentos.

Vamos utilizar o mesmo exemplo da classe Semaforo, porém agora nosso construtor receberá a cor via argumento. Observe a alteração no diagrama e no código da classe:

Semaforo
-cor: String
+Semaforo(cor:String) +sinalizarSiga():void +sinalizarPare():void +sinalizarAtencao():void +toString():String

```
1  public class Semaforo{
2      private String cor; /* Atributo que armazena a cor do semáforo */
3
4      public Semaforo(String cor){
5          this.cor = cor;
6      }
7      public void sinalizarAtencao(){
8          this.cor = "Amarelo";
9      }
10     public void sinalizarPare(){
11         this.cor = "Vermelho";
12     }
13     public void sinalizarSiga(){
14         this.cor = "Verde";
15     }
16
17     public String toString(){
18         return "O semáforo está sinalizando: " + this.cor;
19     }
20 }
```

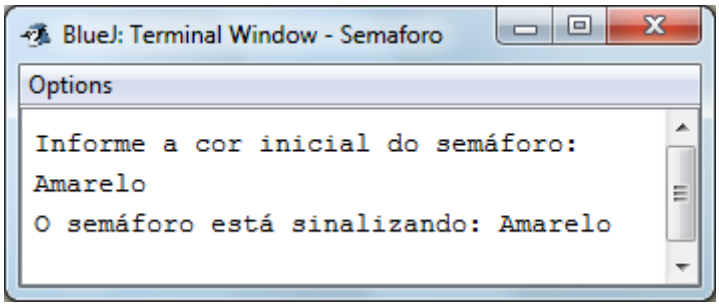
Assim, também teremos alteração na principal, na forma com que criaremos o objeto, **pois uma vez o construtor criado, seu uso é obrigatório**. Logo, antes de criar o objeto precisamos ter em mãos os dados necessários, no caso a cor.

```java
import java.util.Scanner;
public class Main{
    public static void main(String args[]){
        Scanner ler = new Scanner(System.in);
        //lendo a cor do semáforo
        System.out.println("Informe a cor inicial do semáforo: ");
        String cor = ler.next();

        //criando o objeto passando a cor para o construtor
        Semaforo s1 = new Semaforo(cor); //instanciando o objeto

        System.out.println(s1);//exibindo o objeto
    }
}
```

Testando no terminal:



UTILIZANDO MAIS DE UM CONSTRUTOR NA CLASSE

De acordo com a necessidade, podemos ter mais de um construtor na mesma classe, porém eles precisam diferenciar em quantidade e tipos de argumentos. Podemos ter, por exemplo, um construtor vazio e outro com argumento, mas não podemos ter dois construtores vazios.

Vamos observar a classe Semaforo com os dois construtores:

Semaforo
-cor: String
+Semaforo()
+Semaforo(cor:String)
+sinalizarSiga():void
+sinalizarPare():void
+sinalizarAtencao():void
+toString():String

```java
public class Semaforo{
    private String cor; /* Atributo que armazena a cor do semáforo */

    public Semaforo(){
        this.cor = "Vermelho";
    }
    public Semaforo(String cor){
        this.cor = cor;
    }
    public void sinalizarAtencao(){
        this.cor = "Amarelo";
    }
    public void sinalizarPare(){
        this.cor = "Vermelho";
    }
    public void sinalizarSiga(){
        this.cor = "Verde";
    }

    public String toString(){
        return "O semáforo está sinalizando: " + this.cor;
    }
}
```

Assim, na principal, **podemos criar o objeto usando qualquer um dos construtores**.

```java
import java.util.Scanner;
public class Main{
    public static void main(String args[]){
        Scanner ler = new Scanner(System.in);
        //lendo a cor do semáforo
        System.out.println("Informe a cor inicial do primeiro semáforo: ");
        String cor = ler.next();

        //criando o objeto passando a cor para o construtor
        Semaforo s1 = new Semaforo(cor); //instanciando o objeto

        //criando o objeto com o construtor vazio
        Semaforo s2 = new Semaforo();

        System.out.println("Primeiro semáforo: " + s1);//exibindo o objeto s1
        System.out.println("Segundo semáforo: " + s2);//exibindo o objeto s2
    }
}
```

```
BlueJ: Terminal Window - Semaforo
Options
Informe a cor inicial do primeiro semáforo:
Verde
Primeiro semáforo: O semáforo está sinalizando: Verde
Segundo semáforo: O semáforo está sinalizando: Vermelho
```

IMPORTÂNCIA DE USAR MÉTODO CONSTRUTOR

Como vimos o método construtor define como será criado nosso objeto, o que ele conterá na sua criação.

Se pensarmos no dia a dia, ao criarmos uma conta bancária não criamos primeiro a conta e depois informamos os dados para o gerente. Primeiro informamos todos os dados ao gerente, após o mesmo ter todas as informações necessárias, a conta será criada. Se compararmos com o nosso programa, acontece a mesma coisa, até então criávamos objetos sem atribuições, sem dados, alocávamos um espaço na memória com valores nulos, vazios, e a partir de agora ocuparemos um espaço na memória com valores definidos no momento da criação do objeto.