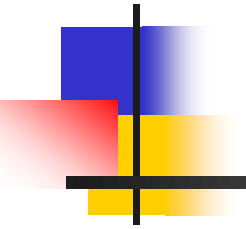


# Construtores

Profª. Rachel Reis

rachel@inf.ufpr.br

# Classe

```
public class <Nome da classe>
{
    // Atributos
    // Construtores
    //  Métodos
}
```

- Toda classe possui um construtor.
- Uma classe pode ter vários construtores.

# O que são Construtores?

- São chamados automaticamente quando instâncias de objetos são criadas por meio da palavra chave ***new***.

```
public class Principal{
        public static void main(String[ ] args)
        {
             Funcionario objeto1 = new Funcionario();
             Funcionario objeto2 = new Funcionario();
         }
}
```

```
public class Funcionario{
    //Atributos
    String nome;
    int ano;
    double salario;


    //Métodos
    public void cadastrar(String nome, int ano, double salario) {
          this.nome = nome;
          this.ano = ano;
          this.salario = salario;
    }

    public void imprimir( ) {
          System.out.println(“Nome: ” + this.nome);
          System.out.println(“Ano: ” + this.ano);
          System.out.println(“Salario: ” + this.salario);
    }
}
```

- Onde está o construtor?

# Construtor padrão (*default*)

- Se nenhum construtor é definido na classe, Java irá definir um construtor padrão (também conhecido como construtor *default*)

```
public class Funcionario{
        // Atributos
        ....
        // Construtor padrão
         public Funcionario() { }
}
```

# Construtor padrão (*default*)

- Características:
  1. Não possui parâmetro.
  2. Não possui conteúdo.
  3. Não possui tipo de retorno.
  4. Possui o mesmo nome da classe.
  5. Possui o mesmo modificador de acesso da classe.

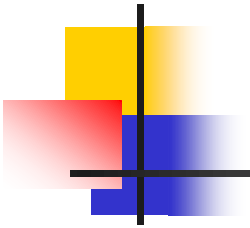
```
public class Funcionario{
        ...
        // Construtor padrão
        public Funcionario( ){ }
}
```

# Construtor padrão (*default*)

- Características:
  1. Não possui parâmetro.
  2. Não possui conteúdo.
  3. Não possui tipo de retorno.
  4. Possui o mesmo nome da classe.
  5. Possui o mesmo modificador de acesso da classe.

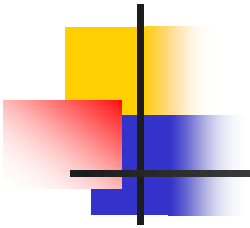
Vale para qualquer tipo de construtor

```
public class Funcionario{
        ...
        // Construtor padrão
        public Funcionario( ){ }
}
```



# Construtor padrão (*default*)

- Curiosidade: Depois de criar um objeto usando o construtor padrão, o que acontece se imprimirmos o conteúdo dos seus campos?



# Construtor padrão (*default*)

- Curiosidade: Depois de criar um objeto usando o construtor padrão, o que acontece se imprimirmos o conteúdo dos seus campos?

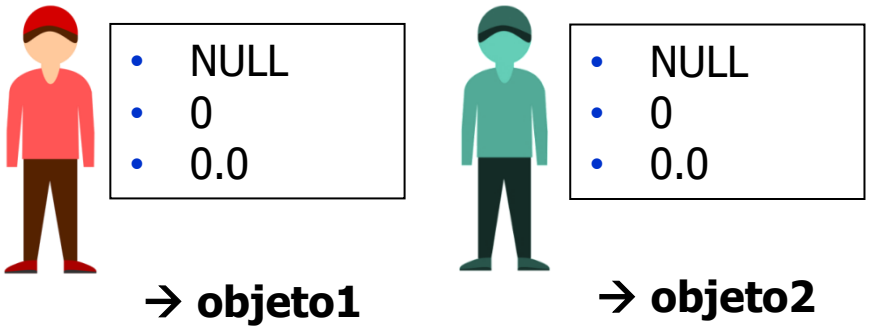
  → São inicializados com valores *default*:

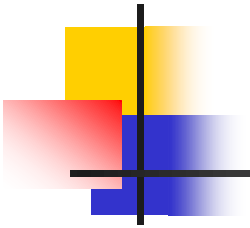
  - Campos do tipo boolean: false
  - Campos do tipo char: vazio
  - Campos do tipo inteiro ou ponto flutuante: zero
  - Instâncias de qualquer classe: NULL

# Construtor padrão (*default*)

- Ex.:

```
public class Principal{
      public static void main(String[ ] args){
            Funcionario objeto1 = new Funcionario( );
            Funcionario objeto2 = new Funcionario( );
      }
}
```



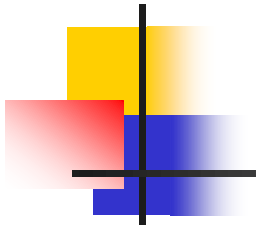


# Outros Construtores

- Construtores diferentes do padrão são úteis para inicializar os “campos de uma instância” (atributos) com valores específicos.

- Formato:

```
<modificador de acesso> <nome do construtor>(<parâmetros>)
{
    //corpo do construtor
}
```



# Outros Construtores

- Ex1:

```
public Funcionario(String nome, int ano, double salario)
{
        this.nome = nome;
        this.ano = ano;
        this.salario = salario;
}
```

# Outros Construtores

- Ex2:

```
public Funcionario(String nome, int ano)
{
        this.nome = nome;
        this.ano = ano;
}
```

- Ex3:

```
public Funcionario(int ano)
{
        this.ano = ano;
}
```

```
public class Funcionario{
    //Atributos
    String nome;
    int ano;
    double salario;

    //Construtores
    public Funcionario(String nome, int ano, double salario){
        this.nome = nome;
        this.ano = ano;
        this.salario = salario;
    };

    public Funcionario(int ano){
        this.ano = ano;
    }

    //Métodos
    …
}
```

- Quantos construtores possui esta classe?

```
public class Principal
{
    public static void main(String[ ] args)
    {
        Funcionario objeto1 = new Funcionario( );
        Funcionario objeto2 = new Funcionario( );
    }
}
```

- O que acontece se tentarmos criar dois objetos usando o construtor padrão?

```
public class Principal
{
    public static void main(String[ ] args)
    {
        Funcionario objeto1 = new Funcionario("Robin", 2021, 2500);

    }

}
```

- Usando o construtor completo para criar o objeto 1

- Robin
- 2021
- 2500

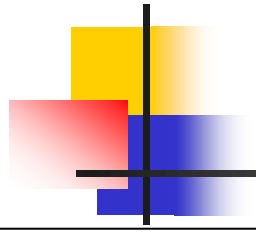
→ **objeto1**

```
public class Principal
{
    public static void main(String[ ] args)
    {
        Funcionario objeto1 = new Funcionario("Robin", 2021, 2500);
        Funcionario objeto2 = new Funcionario(2020);

    }
}
```

- Usando o construtor com um parâmetro para criar o objeto 2

- NULL
- 2020
- 0

→ **objeto2**



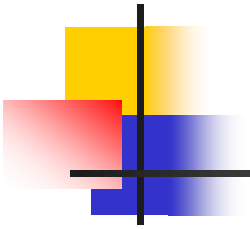
# Diferença - Construtor x Método

```
public Funcionario(String nome, int ano, double salario){
        this.nome = nome;
        this.ano = ano;
        this.salario = salario;
}
```

construtor

```
public void cadastrar(String nome, int ano, double salario){
        this.nome = nome;
        this.ano = ano;
        this.salario = salario;
}
```

método



# Diferença - Construtor x Método

- Construtores devem ter **exatamente** o mesmo nome da classe a que pertencem.
- Construtores devem ser declarado **sem tipo de retorno**.
- Construtores devem usar o modificador de acesso public **somente** se a classe for public.

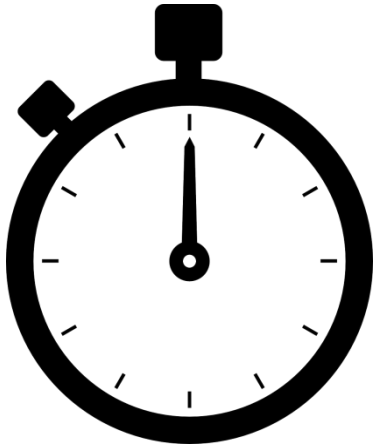
# Diferença - Construtor x Método

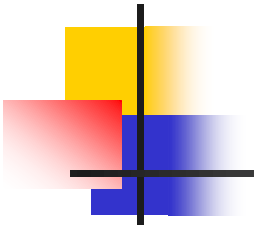
- Construtores **não podem** ser chamados diretamente, somente quando uma instância for inicializada com new.

```
public class Principal{
   public static void main (String[] args){

      Funcionario objeto1.Funcionario();

      Funcionario objeto1 = new Funcionario();


}
```

# Construtor

- Por meio dos construtores podemos garantir que o código que eles contém será executado antes de qualquer outro código.

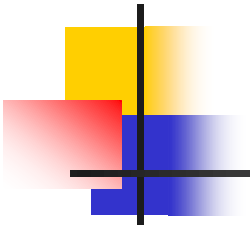




# Praticar...

- Implemente uma classe para representar um eletrônico de sua casa. Defina 4 atributos e 5 construtores. Em seguida, crie objetos para validar os 5 construtores que vocês criou.

- Descreva uma situação do mundo real em que seja necessário criar uma classe em que o código do construtor tenha que ser executado antes de qualquer outro código.



# Referências


- Deitel, P. J.; Deitel, H. M. (2017). Java como programar. 10a edição. São Paulo: Pearson Prentice Hall.

- Barnes, D. J. (2009). Programação orientada a objetos com Java: uma introdução prática usando o BlueJ (4. ed.). São Paulo, SP: Prentice Hall.

- Boratti, I. C. (2007). Programação orientada a objetos em Java. Florianópolis, SC: Visual Books.