

Linguagem de Programação

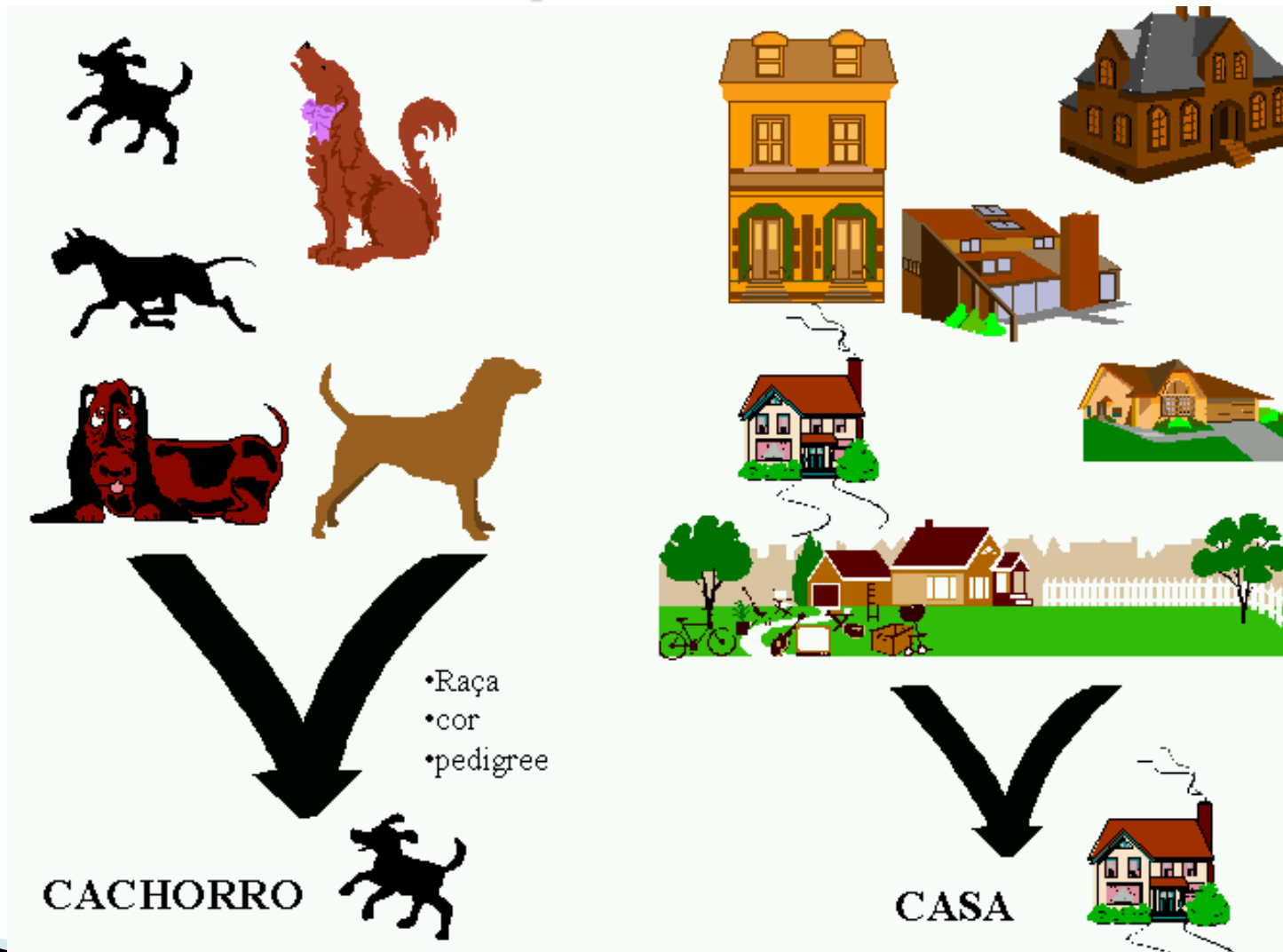
Considerações sobre Orientação a Objetos
(Criando uma classe e instâncias da classe criada)

Conceito de Objetos

Objetos

- Um objeto pode ser concreto ou abstrato, tal como um carro, uma reserva de passagem aérea, uma organização, uma planta de engenharia, um componente de uma planta de engenharia, etc.
- “Qualquer coisa, real ou abstrata, a respeito da qual armazenamos dados e métodos que os manipulam” Martin, Odell (1995)
- Abstração ou conceito do mundo real
- Encapsulamento de atributos e serviços

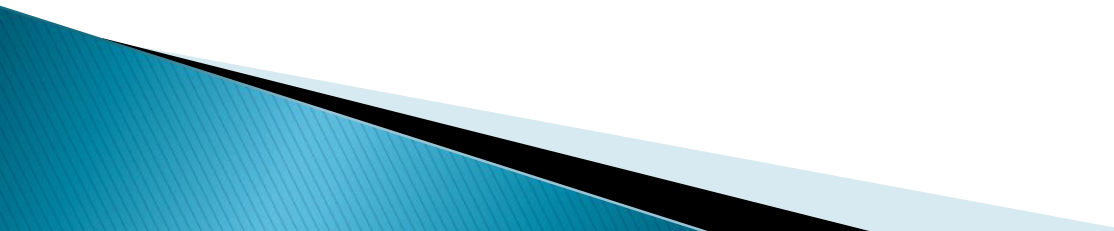
Objetos (Exemplos)



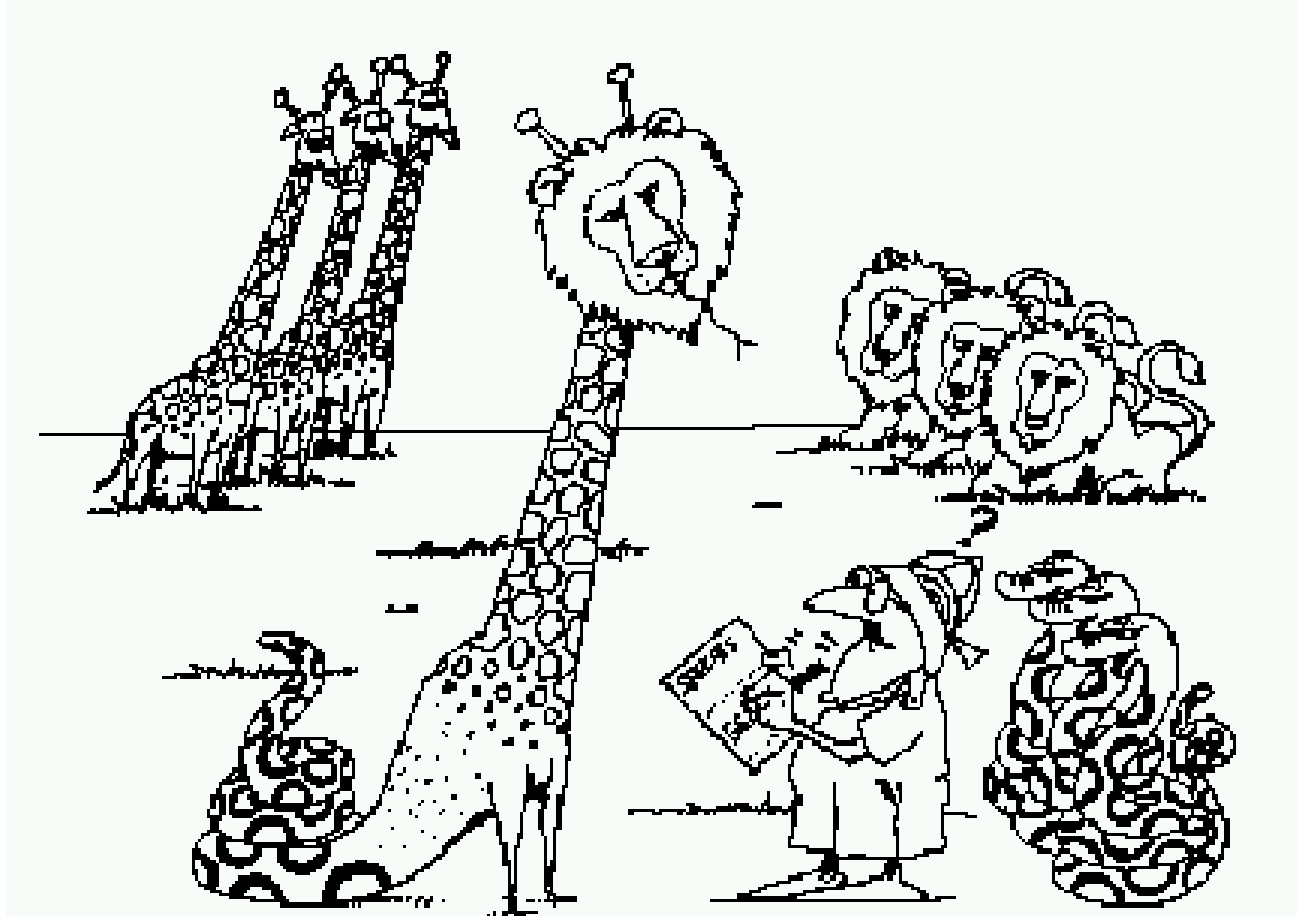
Conceito de Classes

Classes

Uma classe descreve um conjunto de objetos com:

- as mesmas propriedades (atributos),
 - o mesmo comportamento (funcionalidades definidas em métodos),
 - os mesmos relacionamentos com outros objetos e a mesma semântica.
- 

Classes (Exemplo)



Classificação é o meio pelo qual ordenamos conhecimento.

Conceito de Instância

Instância

- Um instância representa uma ocorrência de uma classe.
- Objetos que se comportam da maneira especificada pela classe são ditos *instâncias* dessa classe.
- Assim, cada objeto é uma instância de uma classe. Instanciar uma classe é criar um novo objeto da classe.

Criando uma Classe em Java

- Para declarar uma classe com seus atributos e métodos.

```
qualificador-de-acesso class NomeDaClasse
{
    /* ... colocar uma lista de atributos se necessário */
    qualificador_de_acesso tipo_de_dado_do_atributo nomeAtributo;

    qualificador_de_acesso tipo_de_retorno nomeMétodo (lista de
                        parâmetros)
    {
        /* corpo do método */
    }
    /* ... colocar quantos métodos forem necessários.
}
```

Obs: Os qualificadores de acesso dos atributos serão usados mais adiante para garantir o uso do conceito de encapsulamento.

Exemplo1: Produto.java

Exemplo: Especificando uma classe de produtos em Java. Essa classe contém três atributos correspondentes a dados importantes sobre produtos e um método para imprimir um produto instanciado (que é um determinado objeto).

```
/* arquivo Produto.java */
```

```
public class Produto {
```

```
    public int codProd;
```

```
    public String descricaoProd;
```

```
    public double precoProd;
```

atributos

método

```
    public void imprimirProduto() {
```

```
        System.out.println("Codigo do produto:" + this.codProd);
```

```
        System.out.println("Descricao do produto:" + this.descricaoProd);
```

```
        System.out.println("Preco do produto:" + this.precoProd); }
```

```
}
```

Obs: (não se usa a palavra static para esse método)

Exemplo 1: Criando um objeto

Para criar um objeto que é um produto, ou seja pertencente à classe Produto, utiliza-se uma variável em Java para armazenar o objeto, como se Produto fosse um tipo.

Produto p1;

para acessar um atributo ou método de p1, utilizamos notações parecidas como a de registro (struct em Linguagem C com o acesso aos membros).

```
p1.codProd  
p1.descricaoProd  
p1.precoProd  
p1.imprimirProduto()
```

atributos

A notação ponto (.)

- Para acessar o atributo de instância precisamos usar uma referência.
- O ponto concatena a referência (objeto) e o atributo:
 - Objeto.atributo
 - Ex: `p1.codProd = 123;`
 - Obs: o acesso direto só é possível quando o atributo foi definido como `public` (+)

Exemplo 1: Produto.java

Obs: quando usamos apenas `Produto p1;` entende-se que p1 foi declarado, porém o objeto ainda não foi criado.

É preciso criar um objeto para poder acessar seus atributos e métodos.

Isso é feito utilizando-se a palavra reservada **new**. Isso na verdade está associado ao uso de um construtor (**que consiste em um bloco de código especial, dentro de uma classe, capaz de inicializar novos objetos**).

Para instanciar o objeto deve-se usar:

```
NomeClasse nomeObjeto = new NomeClasse();
```

ex: `Produto p1 = new Produto();`

Exemplo 1: Criando um objeto

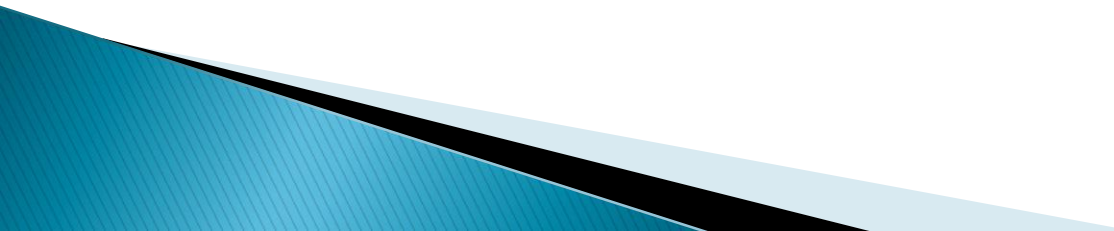
Após criar o objeto p1 a partir de:

```
Produto p1 = new Produto();
```

É possível acessar atributos e chamar a execução do método de p1, conforme pode ser observado:

```
p1.codProd=123;  
p1.descricaoProd="Protetor Solar";  
p1.precoProd=79.9;
```

p1.imprimirProduto(); -> isso corresponde à ativação do método imprimirProduto, ou seja, ocorre a chamada do método a partir de uma mensagem.



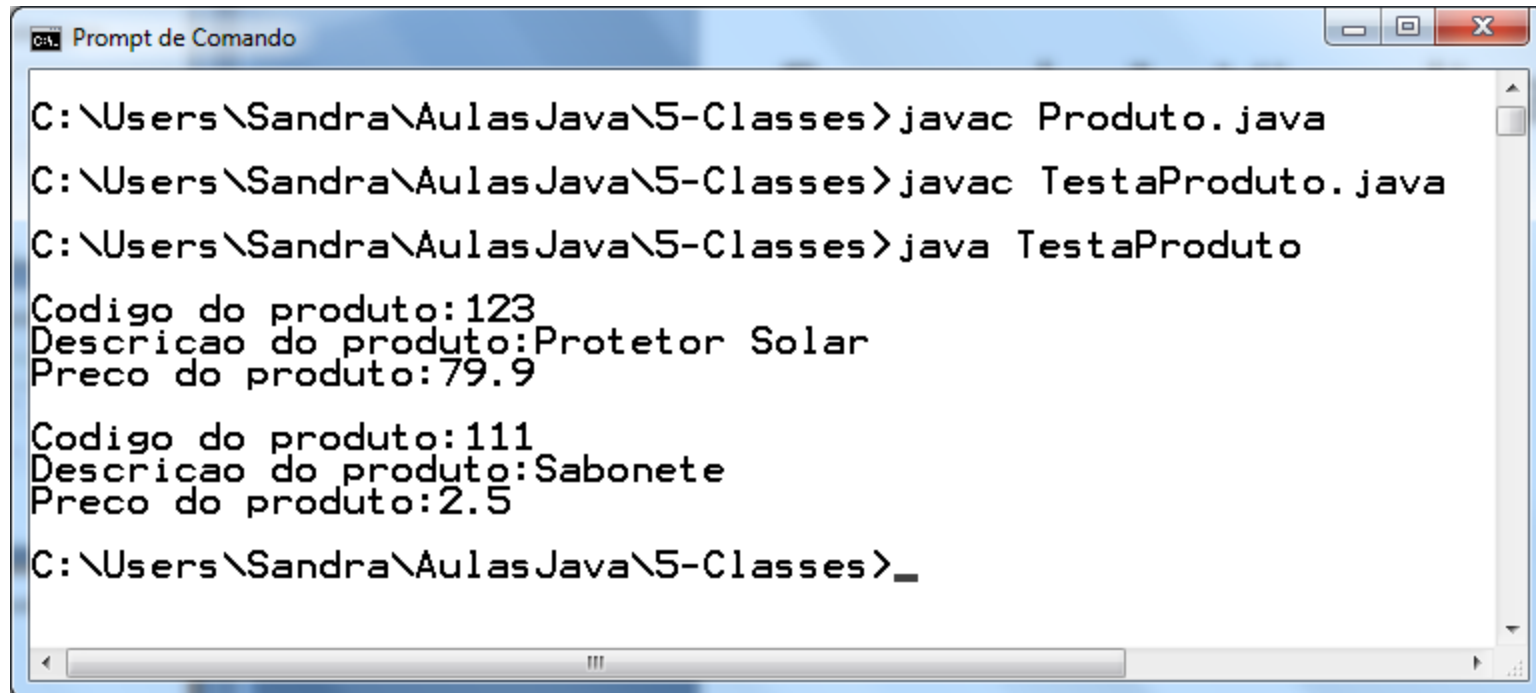
Exemplo 1: Produto.java

```
Produto.java x TestaProduto.java x
1  /* arquivo Produto.java */
2  public class Produto    {
3      public int codProd;
4      public String descricaoProd;
5      public double precoProd;
6
7      public void imprimirProduto()    {
8          System.out.println("");
9          System.out.println("Codigo do produto:"+ this.codProd);
10         System.out.println("Descricao do produto:"+ this.descricaoProd);
11         System.out.println("Preco do produto:"+ this.precoProd);
12     }
13 }
```

Exemplo 1: TestaProduto.java

```
Produto.java x TestaProduto.java x
1  /* Classe principal arquivo TestaProduto.java */
2  public class TestaProduto
3  {
4      public static void main (String[] args)
5      {
6          Produto p1 = new Produto();      /* instancia o objeto p1*/
7          p1.codProd=123;
8          p1.descricaoProd="Protetor Solar";
9          p1.precoProd=79.9;
10         p1.imprimirProduto();
11
12         Produto p2 = new Produto();
13         p2.codProd=111;
14         p2.descricaoProd="Sabonete";
15         p2.precoProd=2.5;
16         p2.imprimirProduto();
17     }
18 }
```

Exemplo 1: Visualizando



```
C:\Users\Sandra\AulasJava\5-Classes>javac Produto.java
C:\Users\Sandra\AulasJava\5-Classes>javac TestaProduto.java
C:\Users\Sandra\AulasJava\5-Classes>java TestaProduto
Codigo do produto:123
Descricao do produto:Protetor Solar
Preco do produto:79.9

Codigo do produto:111
Descricao do produto:Sabonete
Preco do produto:2.5
C:\Users\Sandra\AulasJava\5-Classes>_
```

The image shows a Windows Command Prompt window titled "Prompt de Comando". The window contains the following text:
C:\Users\Sandra\AulasJava\5-Classes>javac Produto.java
C:\Users\Sandra\AulasJava\5-Classes>javac TestaProduto.java
C:\Users\Sandra\AulasJava\5-Classes>java TestaProduto
Codigo do produto:123
Descricao do produto:Protetor Solar
Preco do produto:79.9

Codigo do produto:111
Descricao do produto:Sabonete
Preco do produto:2.5
C:\Users\Sandra\AulasJava\5-Classes>_

Construtor

- Conforme mencionado anteriormente um construtor **consiste em um bloco de código especial, dentro de uma classe, capaz de inicializar novos objetos.**
- Assim, construtores são usados para criar instâncias (um objeto/uma ocorrência) de uma classe.
- Criar uma instância significa criar um objeto ou ocorrência de uma classe.
- Em Java, para chamar um construtor usamos a palavra **new**.

Construtor

O exemplo abaixo usa a classe Produto em uma outra classe definida como TestaProduto, que contém o método main e que poderá ser executada. Percebe-se que p1 é uma instância da classe Produto.

```
/* Classe principal arquivo TestaProduto.java */
```

```
public class TestaProduto{
```

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

```
        Produto p1 = new Produto(); /* instancia o objeto p1*/
```

```
        p1.codProd=123;
```

```
        p1.descricaoProd="Protetor Solar";
```

```
        p1.precoProd=79.9;
```

```
        p1.imprimirProduto();
```

```
    }
```

```
}
```

Construtor

Observe no exemplo que declaramos e criamos ao mesmo tempo em um único comando:

```
Produto p1 = new Produto();
```

Ao invés de usar os comandos abaixo:

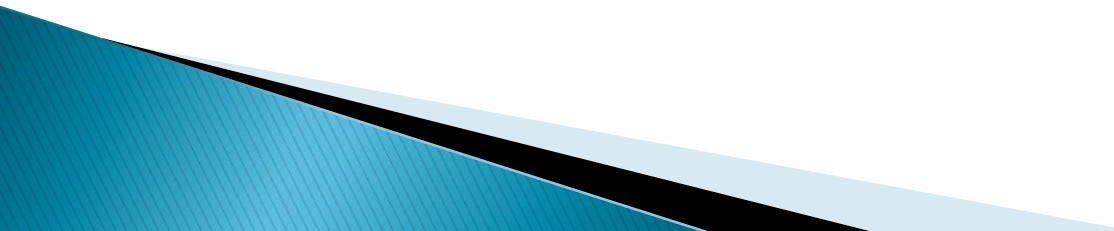
```
Produto p1;
```

```
p1 = new Produto();
```



Construtor – Regras

Para definir um construtor utilize as seguintes regras:

- um construtor nunca retorna nada, não tem tipo de retorno;
 - tem sempre o mesmo nome da classe (obrigatoriamente);
 - deve ser sempre público;
 - é chamado para instanciar um objeto; e
 - nunca é do tipo abstrato.
- 

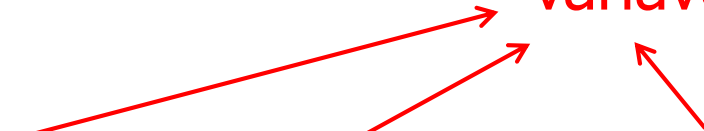
Construtores (Resumo)

- Construtores são usados para criar (iniciar) objetos.
- Um construtor não possui tipo de retorno, tem o mesmo nome da classe, e pode possuir parâmetros.
- O operador **new** é o responsável pela criação de um novo objeto.

• Ex:

```
public Produtos(int cod, String desc, double pr){  
    /* construtor com parâmetros da classe Produtos */  
    this.codProd = cod;  
    this.descricaoProd = desc;  
    this.precoProd = pr;  
}
```

variáveis locais



Tipos de Construtores

- Os construtores podem ser de três tipos:
 - Construtor sem parâmetro: cria um objeto com valores especificados no corpo do construtor, não permitindo que o usuário passe parâmetros ao construtor;
 - Construtor com parâmetro: construtor que cria um objeto com valores especificados a partir da passagem de parâmetros;
 - Construtor cópia: gera uma cópia de um outro objeto. Para isso esse tipo de construtor precisa de uma referência vinculada a uma instância.

Tipos de Construtores

```
Produtos.java x TestaProduto.java x
1  /* arquivo Produtos.java */
2  public class Produtos {
3      public int codProd;
4      public String descricaoProd;
5      public double precoProd;
6
7      public Produtos() {
8          /* construtor sem parâmetros da classe Produtos */
9          codProd = 0;
10         descricaoProd = "AAA";
11         precoProd = 0.0;
12     }
13     public Produtos(int cod, String desc, double pr) {
14         /* construtor com parâmetros da classe Produtos */
15         codProd = cod;
16         descricaoProd = desc;
17         precoProd = pr;
18     }
19     public Produtos(Produtos p) {
20         /* construtor cópia da classe Produtos, também chamado de Copy constructor */
21         codProd = p.codProd;
22         descricaoProd = p.descricaoProd;
23         precoProd = p.precoProd;
24     }
25     public void imprimirProduto() {
26         System.out.println("Codigo do produto:" + this.codProd);
27         System.out.println("Descricao do produto:" + this.descricaoProd);
28         System.out.println("Preco do produto:" + this.precoProd);
29     }
30 }
```

Construtor – Exemplo (continuação)

```
/* arquivo Produtos.java */
```

```
public class Produtos {  
    public int codProd;  
    public String descricaoProd;  
    public double precoProd;
```

Construtor sem parâmetro

```
    public Produtos(){  
        /* construtor sem parâmetros da classe Produtos */  
        codProd = 0;  
        descricaoProd = "AAA";  
        precoProd = 0.0;  
    }
```

Construtor – Exemplo (continuação)

/* continuação do arquivo Produtos.java */

Construtor com parâmetro

```
public Produtos(int cod, String desc, double pr){  
    /* construtor com parâmetros da classe Produtos */  
    codProd = cod;  
    descricaoProd = desc;  
    precoProd = pr;  
}
```

```
public Produtos(Produtos p){  
    /* construtor cópia da classe Produtos,  
    também chamado de Copy constructor */  
    codProd = p.codProd;  
    descricaoProd = p.descricaoProd;  
    precoProd = p.precoProd;  
}
```

Construtor cópia

Construtor – Exemplo (continuação)

```
/* continuação do arquivo Produtos.java */
```

```
    public void imprimirProduto()    {  
        System.out.println("Codigo do produto:" + this.codProd);  
        System.out.println("Descricao do produto:" + this.descricaoProd);  
        System.out.println("Preco do produto:" + this.precoProd);  
    }  
}
```

Construtor – Exemplo (instanciação)

```
1  /* Classe principal arquivo TestaProdutos.java */
2  public class TestaProdutos
3  {
4      public static void main (String[] args)
5      {
6          /* instancia o objeto p1*/
7          Produtos p1 = new Produtos();
8          p1.imprimirProduto();
9
10         /* instancia o objeto p2*/
11         Produtos p2 = new Produtos(123,"Protetor Solar",79.9);
12         p2.imprimirProduto();
13
14         /* instancia o objeto p3*/
15         Produtos p3 = new Produtos(p2);
16         p3.imprimirProduto();
17     }
18 }
```

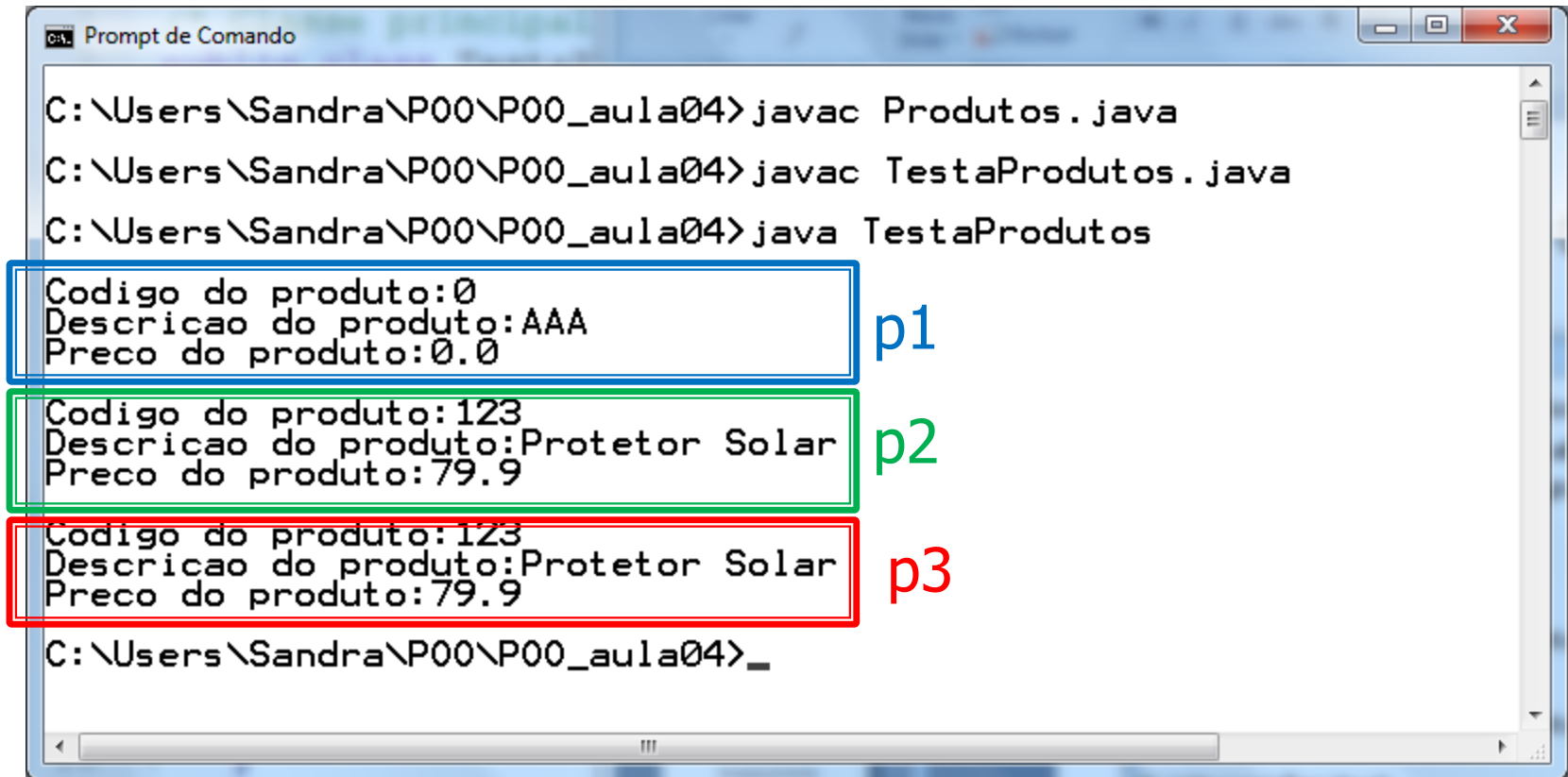
O construtor pode conter parâmetros e sua chamada é como uma função normal.

O copy constructor copia p2 em p3.

Não está certo fazer `p3 = p2`, pois p3 e p2 funcionam como apontadores para posição de memória e você poderá ter problemas em utilizar esse comando, assim como compará-los em expressões `p3 == p2`.

Usando tipos
diferentes de
construtores

Visualizando o uso de construtores



```
ch: Prompt de Comando

C:\Users\Sandra\P00\P00_aula04>javac Produtos.java
C:\Users\Sandra\P00\P00_aula04>javac TestaProdutos.java
C:\Users\Sandra\P00\P00_aula04>java TestaProdutos

Codigo do produto:0
Descricao do produto:AAA
Preco do produto:0.0
Codigo do produto:123
Descricao do produto:Protetor Solar
Preco do produto:79.9
Codigo do produto:123
Descricao do produto:Protetor Solar
Preco do produto:79.9
C:\Users\Sandra\P00\P00_aula04>_
```

p1

p2

p3

Estrutura do programa Java

Declaração
Da Classe

Corpo da Classe

Atributos

Construtores

Métodos

```
/* arquivo Produtos.java */
public class Produtos {

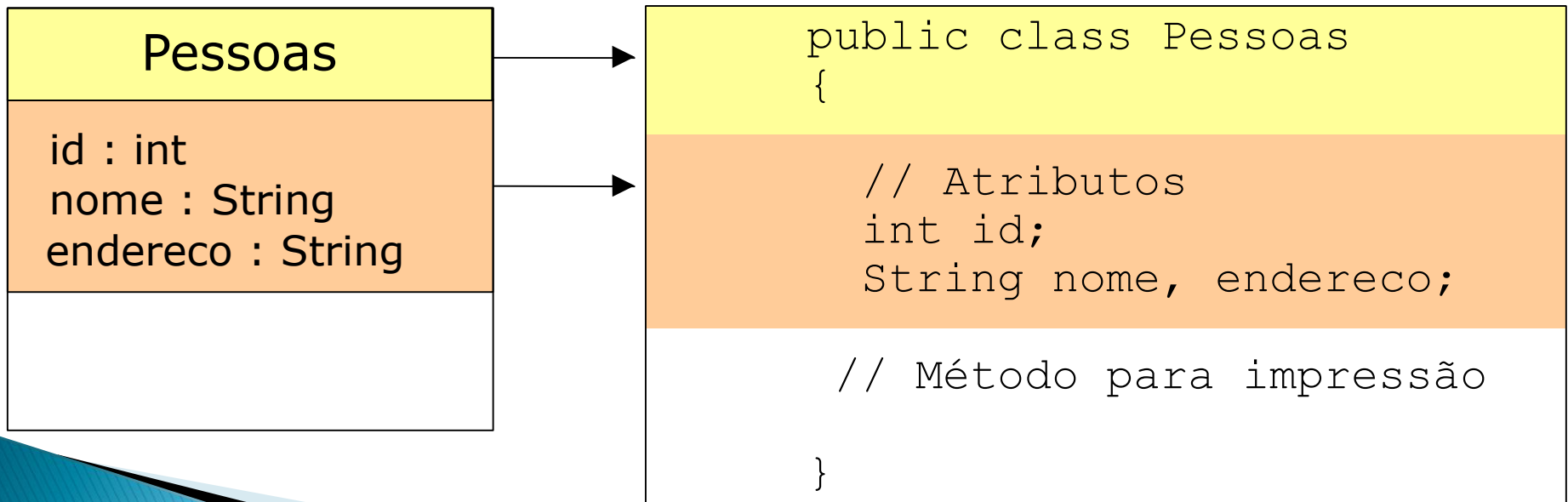
    public int codProd;
    public String descricaoProd;
    public double precoProd;

    public Produtos() {
    }
    public Produtos(int cod, String desc, double pr) {
    }
    public Produtos(Produtos p) {
    }

    public void imprimirProduto() {
        System.out.println("");
        System.out.println("Codigo do produto:" + this.codProd);
        System.out.println("Descricao do produto:" + this.descricaoProd);
        System.out.println("Preco do produto:" + this.precoProd);
    }
}
```

Exercício

- 1) Considere a necessidade de criar uma classe definida pelo nome Pessoas. Os atributos dessa classe serão os seguintes: id, nome e endereço. Crie um método para impressão.



Exercício

Implemente a classe Pessoas. Crie uma outra classe para inserir a main; chame essa classe de TestaPessoas e instancie dois objetos definidos como: p1 e p2. Defina valores para os atributos dos objetos e os imprima.