
Programação Orientada a Objetos

- Tipo abstrato de dado
- Classes
- Atribuição de valores
- Referência para objetos na memória
- Tipo de métodos
 - Construtores
 - Operacionais

Programação Orientada a Objetos

Tipo Abstrato de Dados (TAD):

Conjunto de dados e operações sobre esses dados, que permite que você defina novos tipos na linguagem, ocultando dados internos e o estado, atrás de uma interface bem definida.

Tipos

Os tipos definem diferentes espécies de valores que estão disponíveis em seus programas.

Ex.: `String x;`

A `String` é um tipo de variável, definida dentro do Java como uma classe. Esse tipo – ou classe – reúne dados e operações que podem ser realizadas pela usuário.

Dependendo do domínio da aplicação aonde se encontra o seu problema, você também pode criar tipos abstratos de dados.

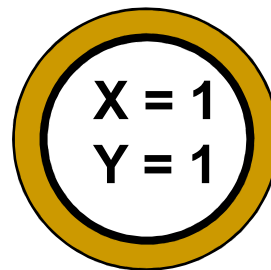
Programação Orientada a Objetos

- Declaração:

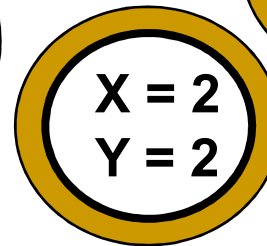
```
class Ponto {  
    ...  
}
```

- Propriedades: dados que as instâncias da classe conterão:

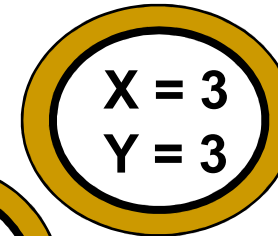
```
class Ponto {  
    float x, y;  
}
```



objPonto1



objPonto2



objPonto3

Qualquer objeto do tipo **Ponto** possuirá dois atributos denominados x e y



Programação Orientada a Objetos

Classe executadora (que contém o método main) pode utilizar classe criada (TAD) criando objetos a partir da mesma.

Declaração das variáveis:

```
class Editor {  
    public static void main (String args[]) {  
        Ponto p;  
        int a = 2;  
    }  
}
```

Memória

2

X = ?
Y = ?

Declaração de uma
variável a do tipo inteiro.

Declaração de uma variável p
do tipo da classe Ponto.

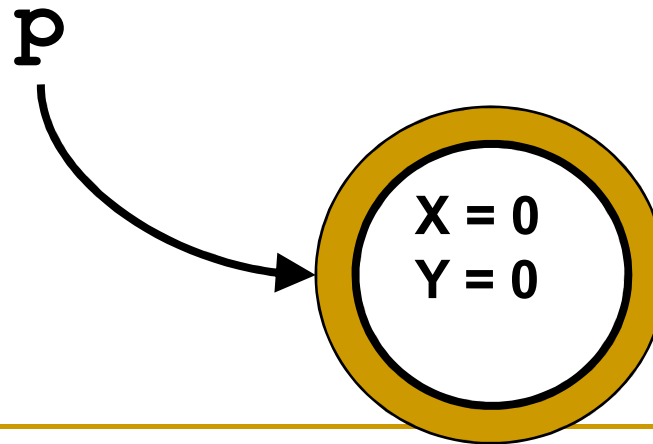
Programação Orientada a Objetos

Criação de objeto (instância da classe Ponto) :
→ utilização do comando **new**.

```
class Editor {  
    public static void main (String arg[])  
    {  
        Ponto p = new Ponto();  
    }  
}
```

Programação Orientada a Objetos

```
class Editor {  
    public static void main (String arg[]) {  
        Ponto p = new Ponto();  
    }  
}
```



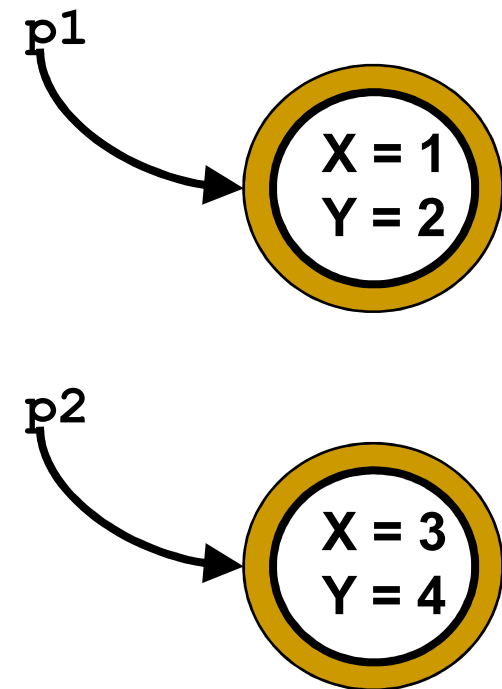
Programação Orientada a Objetos

Atribuição de valores

As propriedades dos objetos podem ser manipuladas diretamente:

```
Ponto p1 = new Ponto();  
p1.x = 1;  
p1.y = 2;  
// p1 representa o ponto (1,2)
```

```
Ponto p2 = new Ponto();  
p2.x = 3;  
p2.y = 4;  
// p2 representa o ponto (3,4)
```

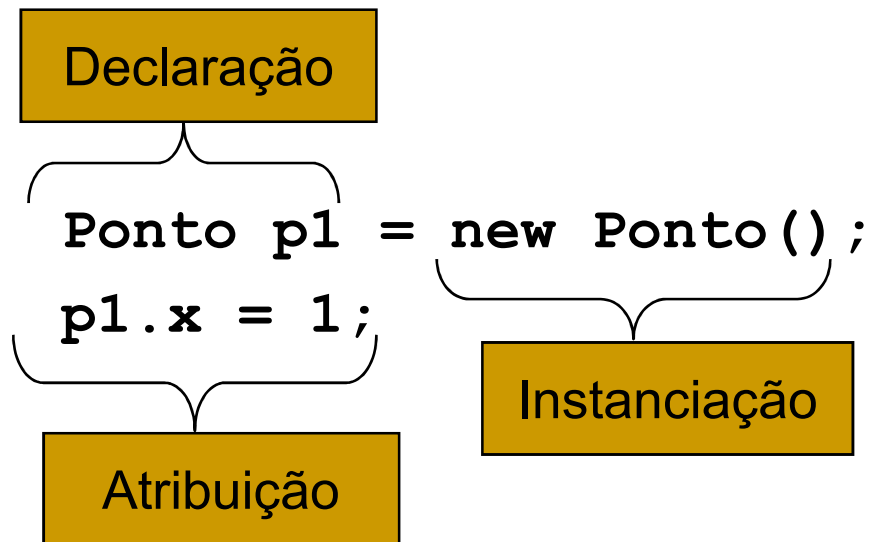


Mas dentro da OO, os atributos de outras classes são sempre acessados por métodos operacionais de acesso, preservando o encapsulamento.

Programação Orientada a Objetos

Atribuição de valores

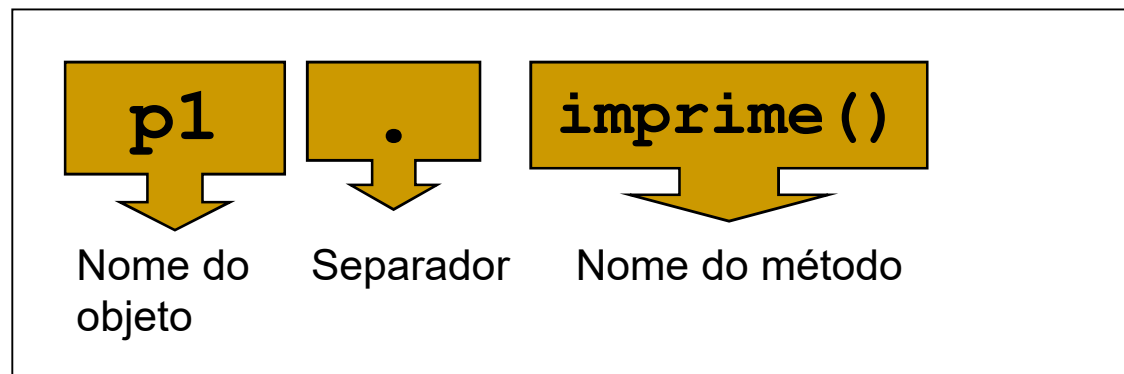
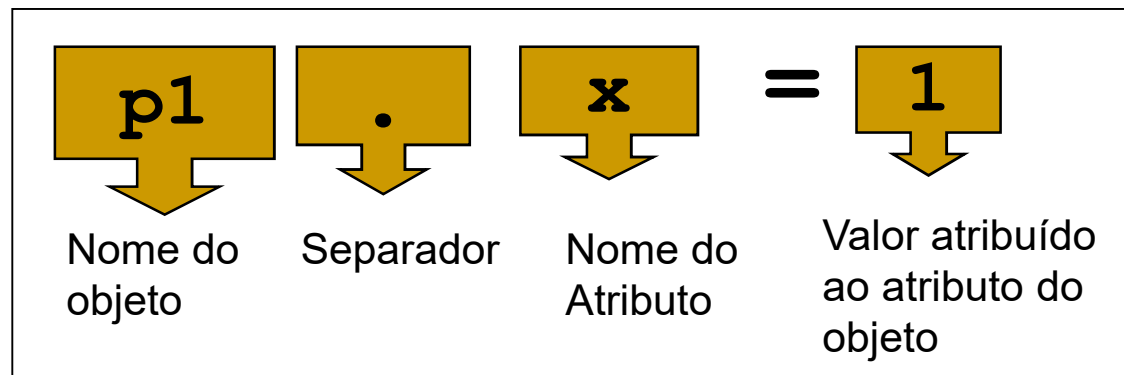
Atenção !!!



Programação Orientada a Objetos

Acessando membros de uma classe

Membros de uma classe são seus atributos e métodos.



Ponto
float x foat y
imprimeX()

Exercícios

Programação:

Crie a classe Ponto em Java e um aplicativo que cria tres objetos do tipo ponto e imprima suas coordenadas como na saída abaixo:

Ponto	CoordenadasX	Coordenadas Y
1	1	1
2	5	2
3	2	1

Os métodos setX e setY recebem ambos uma variavel do tipo float que e atribuida ao atributo X ou Y do objeto



Ponto
float x foat y
imprimeX() setX(float novoValor) setY(float novoValor)

Programação Orientada a Objetos

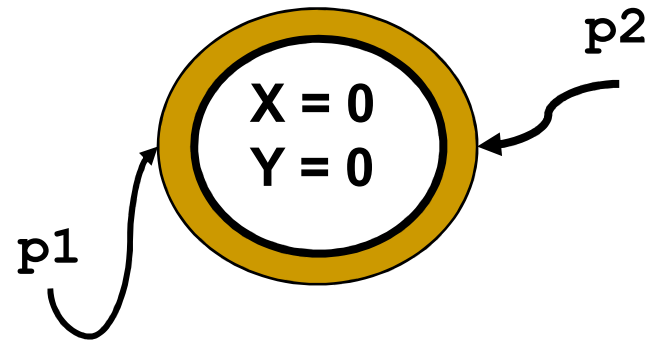
Referência para Objetos

Toda vez que uma variável recebe um objeto, ela está recebendo a sua referência:

```
Ponto p1 = new Ponto();
```

O que ocorre quando um outro objeto do tipo Ponto aponta para a mesma referência ??

```
Ponto p2 = p1;
```



Programação Orientada a Objetos

Métodos

Definem o comportamento da classe.

```
class Ponto
{
    int x, y;

    void mover ( int dx, int dy)
    {
        x += dx;
        y += dy;
    }
}
```

Programação Orientada a Objetos

Tipos de Métodos

Método
Construtor

Método
Operacional

```
class Ponto
{
    float X;
    float Y ;

    Ponto ()
    {
        ...
    }

    void Mover ()
    {
        ...
    }
}
```


Programação Orientada a Objetos

Tipos de Métodos



Método Construtor

O método construtor é o primeiro método chamado quando instanciamos um objeto. Ele normalmente é usado para inicializar os atributos do objeto.



Método Operacional

Os métodos operacionais são todos os outros métodos da classe, excluindo o construtor. Dentre os métodos operacionais encontramos os métodos de acesso aos atributos do objeto, métodos esses que retornam ou alteram o valor de algum atributo do objeto.

Programação Orientada a Objetos

Métodos Construtores

Devemos usar os métodos construtores quando queremos atribuir valores aos atributos de um objeto no momento de sua criação.

```
class Ponto {
```

```
    int x=0;
```

```
    int y=0;
```

```
    Ponto (int x, int y)
```

```
    {
```

```
        this.x = x;
```

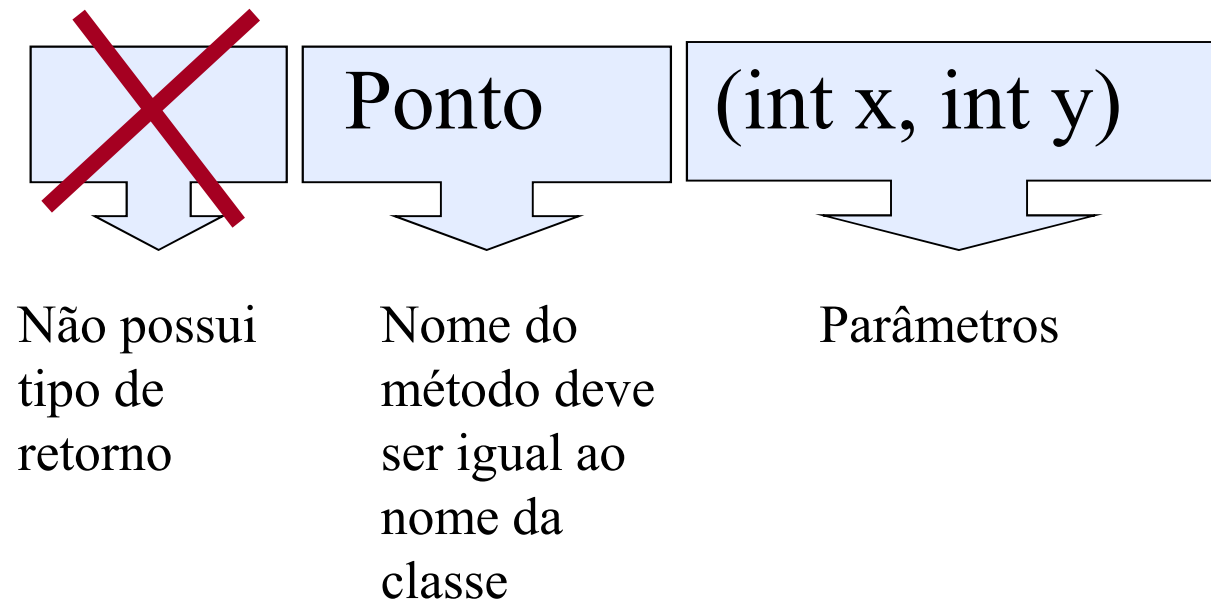
```
    }
```

```
}
```

Método Construtor

Programação Orientada a Objetos

Assinatura de um método Construtor



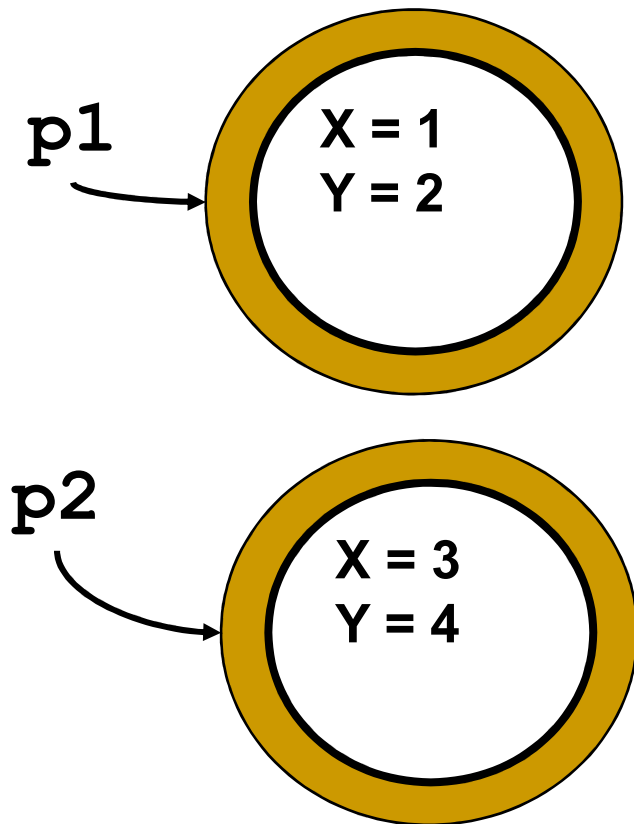
Programação Orientada a Objetos

Uso de Construtores

Deve ser usado no momento da criação do objeto:

```
Ponto p1 = new Ponto(1,2);
```

```
Ponto p2 = new Ponto(3,4);
```



Programação Orientada a Objetos

Método Construtor

Exemplo – Aplicativo criando objetos Ponto

```
class CriaPontos
{
    public static void main (String args[])
    {
        Ponto p1, p2;
        p1 = new Ponto (1,2) ;
        p2 = new Ponto (3,4) ;
    }
}
```

Programação Orientada a Objetos

Construtor Padrão

- A linguagem Java declara um construtor padrão, vazio, que não recebe nenhum parâmetro
- Quando declaramos um novo construtor, esse construtor padrão deixa de existir e é substituído pelo novo construtor

Programação Orientada a Objetos

Métodos Construtores

- Usados na criação de um objeto através do comando *new*
- **Possuem o mesmo nome da classe**
- Podem receber parâmetros que servirão para inicialização dos atributos da classe
- Uma classe pode ter vários métodos construtores

Programação Orientada a Objetos

Métodos Construtores

```
class Funcionario  
{ String nomefunc;  
  float salario;
```

```
    public Funcionario (String nome,float sal)  
    {  
        salario = sal;  
        nomefunc = nome;  
    }
```

Método
Construtor

```
    public double calcularSalario() {...}  
    public String getNomeFunc() {...}  
}
```

Métodos
Operacionais

Programação Orientada a Objetos

Exemplo

```
class ProgramaPrincipal
{
    public static void main (String args[])
    {
        Funcionario func;
        func = new Funcionario("Juca",102);
    }
}
```



Método
Construtor

Programação Orientada a Objetos

Métodos Operacionais

- Implementam as funções de uma classe
- Possuem sintaxe semelhante à sintaxe de definição das funções de um programa procedural
- Determinam o comportamento da classe e a troca de mensagens com outras classes

Programação Orientada a Objetos

Métodos Operacionais

```
class Funcionario  
{ String nomeFunc;  
  float salario;
```

```
  ...  
  float calcularSalario (int horas)
```

Assinatura

```
{   float salMes = 0;  
    if (horas < 220)  
    {  
        salMes =(salario/220)*horas;  
    }  
    return salMes;
```

Corpo

```
}
```

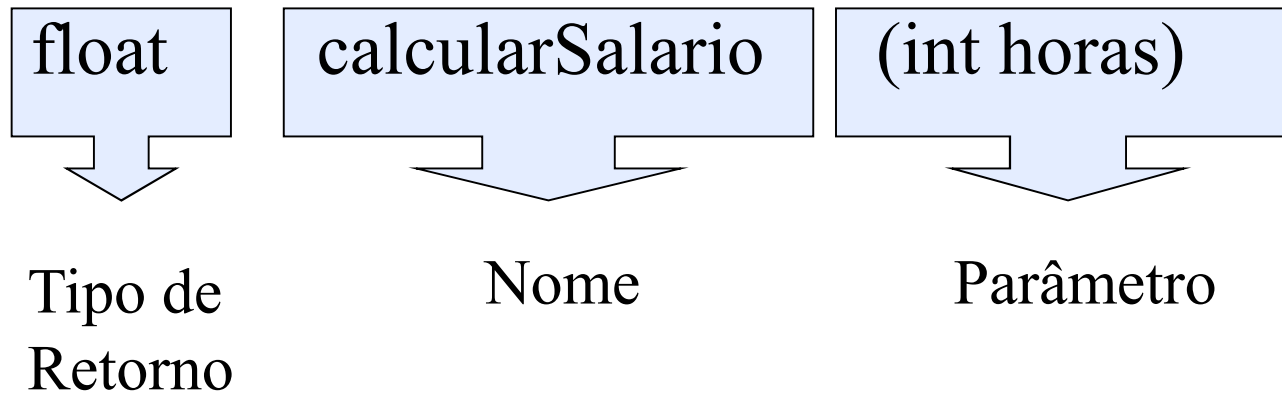
Retorno

```
  ...
```

```
}
```


Programação Orientada a Objetos

Assinatura de um método Operacional



Programação Orientada a Objetos

Exemplo

```
class ProgramaPrincipal
{
    public static void main (String arg[])
    { float sal;
      Funcionario func;
      func = new Funcionario("Juca",2200);
      sal = func.calcularSalario(80);
      System.out.println (sal);
    }
}
```

Executando
o Método
calcularSalario da
classe Funcionario

func

nomeFunc = "Juca"
salario = 2200

calcularSalario(horas)

sal = func.calcularSalario(80);

**sal = (salario/220)*horas;
sal = (2200/220)*80
sal = 800**