

# Programação Orientada a Objetos

*Programação Baseada em Objetos*

Rone Ilídio  
Thiago Oliveira



# Padrão de Nome

- Todas as palavras de uma classe devem ser iniciadas por maiúsculo.
  - Ex. de classes: PessoaFisica, VendaPrazo
- Com exceção da primeira letra, todas as palavras de um método devem ser iniciadas por maiúsculo.
  - Ex. de métodos: buscarVenda(), apagarRegistro()
- Variáveis, uma palavra toda em minúsculo.



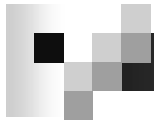
# Escopo de Classe

- As variáveis de instância e métodos da classe pertencem ao escopo dela.
- Dentro do escopo da classe, os membros estão acessíveis para todos os seus métodos e podem ser chamados simplesmente pelo nome.
- Fora do escopo da classe, os membros visíveis (ex: membros public) são chamados através de:
  - nomeDoObjeto.nomeDoMembro

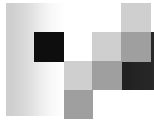


# Escopo de Classe

- Variáveis são visíveis somente dentro do escopo do método.
- Uma variável local a um método sobrescreve uma variável de instância com o mesmo nome.
- Para acessar a variável de instância, usa-se a palavra reservada “this”:
  - `this.nomeVariável`




```
import javax.swing.*;
public class Escopo extends JApplet{
    int x = 0;
    public void init (){
        int x = 1;
        JOptionPane.showMessageDialog(null, "x local: " + x);
        JOptionPane.showMessageDialog(null,
                                     "x variavel de instancia: " + this.x);
        {
            int y = x;
            JOptionPane.showMessageDialog(null,
                                     "Valor de y: " + y);
        }
        System.exit(0);
    }
}
```

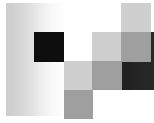


# Programação Baseada em Objetos

- Até agora, o objetivo foi fornecer uma base concreta de programação estruturada.
- Alguns conceitos já foram abordados, como classe e método.
- O exemplo a seguir utiliza os conceitos de orientação a objetos.
  - Para tanto, essas classes devem estar no mesmo diretório.



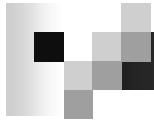
```
public class Pessoa{  
    private String nome;  
    private int idade;  
    public Pessoa(){  
        nome = "";  
        idade = 0;  
    }  
    public String getNome(){  
        return this.nome;  
    }  
    public int getIdade(){  
        return this.idade;  
    }  
    public void setNome(String nome){  
        this.nome = nome;  
    }  
    public void setIdade(int idade){  
        this.idade = idade;  
    }  
}
```



# Programação Baseada em Objetos

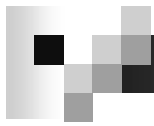
- Foi declarada a classe Pessoa, que não é um *applet* nem um aplicativo.
  - Não possui nem o método *main*, nem os métodos típicos de um *applet*.
- Possui duas variáveis de instância declaradas como *private* (atributos) e cinco métodos declarados como *public*.
- Possui um método com o mesmo nome da classe, método construtor.





# Programação Baseada em Objetos

- Os atributos são declarados como *private* para evitar que outras classes diretamente façam acesso a seus valores.
  - Só podem ser acessados pelos métodos da classe.
- Os métodos são declarados como *public*, pois podem ser acessados em qualquer lugar.



```
public class UsaPessoa{
    public static void main(String[] args){
        Pessoa pessoa = new Pessoa();
        System.out.println("Nome="+ pessoa.getNome());
        System.out.println("Idade="+ pessoa.getIdade());

        pessoa.setNome("Ze");
        pessoa.setIdade(18);

        System.out.println("Nome="+ pessoa.getNome());
        System.out.println("Idade="+ pessoa.getIdade());

        //System.out.println("Nome="+ pessoa.nome);
    }
}
```



# Programação Baseada em Objetos

- Foi declarada a classe UsaPessoa, que é um aplicativo.
- `Pessoa pessoa = new Pessoa();`
  - Cria uma instância de Pessoa na memória, ou seja, cria na memória um objeto denominado pessoa que é do tipo Pessoa.
- No momento da criação, é executado o método construtor `Pessoa()`, sempre!



# Programação Baseada em Objetos

- Foram chamados os métodos do objeto pessoa.
  - Métodos iniciados com *set* modificam os valores dos atributos do objeto.
  - Métodos iniciados com *get* retornam os valores contidos nos atributos do objeto.
- Se tentarmos acessar o valor de um atributo declarado *private*, ocorre erro de compilação.
- A compilação de UsaPessoa automaticamente compila Pessoa.




# Métodos Construtores

- São executados na criação dos objetos.
- Possuem o mesmo nome da classe.
- Não possuem tipo de retorno.
- Normalmente utilizados para inicializar atributos.
- Se não forem criados, o compilador cria um construtor padrão.
- Chamada do construtor:
  - **Pessoa** pessoa = **new Pessoa()**;



# Métodos Construtores

```
public class Pessoa{  
    private String nome;  
    private int idade;  
    public Pessoa() { }  
    public String getNome(){  
        return nome;  
    }  
    public int getIdade(){  
        return idade;  
    }  
    public void setNome(String nome){  
        this.nome = nome;  
    }  
    public void setIdade(int idade){  
        this.idade = idade;  
    }  
}
```

- 
- Construtor vazio
  - Se não for declarado o compilador cria



# Construtores com Parâmetros

```
public class Pessoa{  
    private String nome;  
    private int idade;  
    public Pessoa(String nome, int idade) {  
        this.nome = nome;  
        this.idade = idade;  
    }  
    ...  
}
```

- O construtor recebe dois parâmetros, uma String e um inteiro.
  - Os valores desses parâmetros são passados para os atributos.

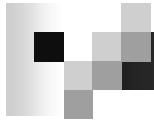


# Construtores com Parâmetros

- Na chamada do construtor, torna-se obrigatória a passagem dos parâmetros.


```
public class Principal{  
    public static void main(String[] args){  
  
        Pessoa pessoa;  
        pessoa = new Pessoa("José" , 18);  
  
        System.out.println("Nome="+ pessoa.getNome());  
        System.out.println("Idade="+ pessoa.getIdade());  
    }  
}
```



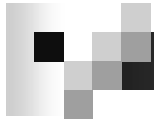


# Referência *this*

- A referência *this* é utilizada para acessar métodos ou variáveis de instância, dentro do corpo de uma classe.
- A classe Pessoa poderia, por exemplo, ter um método para exibir a alteração do nome.




```
public class Pessoa{
    private String nome;
    private int idade;
    public int getIdade(){
        return idade;
    }
    public void setIdade(int idade){
        this.idade = idade;
    }
    public String getNome(){
        return nome;
    }
    public void setNome(String nome){
        mostrarAlteracao(nome);
        this.nome = nome;
    }
    public String mostrarAlteracao(String nome){
        JOptionPane.showMessageDialog(null,
            "Nome atual: " + this.nome + "e novo nome:" + nome);
    }
}
```



# Método toString

- Esse método é executado toda vez que chamamos um objeto somente pelo identificador, sem especificarmos um método ou atributo public.
- Pode ser definido de acordo com a forma de exibição desejada para o objeto.



```
public class Pessoa{  
    private String nome;  
    private int idade;  
    public Pessoa(){  
        nome = "";  
        idade = 0;  
    }  
    public String getNome(){  
        return this.nome;  
    }  
    public int getIdade(){  
        return this.idade;  
    }  
    public void setNome(String nome){  
        this.nome = nome;  
    }  
    public void setIdade(int idade){  
        this.idade = idade;  
    }  
    public String toString(){  
        return nome + idade;  
    }  
}
```



```
public class UsaPessoa
```

```
{
```

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

```
    {
```

```
        Pessoa pessoa = new Pessoa();
```

```
        System.out.println("Nome: " + pessoa.getNome());
```

```
        System.out.println("Idade: " + pessoa.getIdade());
```

```
        pessoa.setNome("Ze");
```

```
        pessoa.setIdade(18);
```

```
        System.out.println("Nome: " + pessoa.getNome());
```

```
        System.out.println("Idade: " + pessoa.getIdade());
```

```
        System.out.println("Dados da pessoa:" + pessoa);
```

```
    }
```

```
}
```