

# Programação Orientada a Objetos

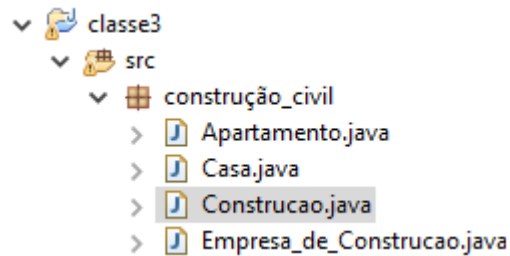
## *Conceitos de Polimorfismo*

*profº Mauricio Conceição Mario*

# Polimorfismo

Polimorfismo é o princípio pelo qual duas ou mais **classes derivadas** de uma mesma **superclasse** podem invocar **métodos que têm a mesma identificação** (assinatura) mas **comportamentos distintos**, especializados para cada classe derivada, usando para tanto uma referência a um objeto do tipo da superclasse. Esse mecanismo é fundamental na programação orientada a objetos, permitindo definir funcionalidades que operem genericamente com objetos, abstraindo-se de seus detalhes particulares quando esses não forem necessários.

# Polimorfismo



A screenshot of the code editor showing the source code of the `Construcao.java` file. The editor has several tabs open: `Calcula_are...`, `Construcao.java` (active), `Casa.java`, `Apartamento...`, and `Empresa_de_C...`. The code is as follows:

```
1 package construção_civil;
2
3 public class Construcacao {
4     private String tipo;
5
6     public void setTipo (String tipo){
7         this.tipo = tipo;
8     }
9
10    public String getTipo (){
11        return tipo;
12    }
13
14    public void constroi(){
15        System.out.println("FAZ-SE QUALQUER TIPO DE CONSTRUÇÃO");
16    }
17 }
```

# Polimorfismo

Calcula\_are... Construcao.java Casa.java Apartamento.... Empresa\_de

```
1 package construço_civil;
2
3 public class Casa extends Construcao {
4
5     Construcao casa = new Construcao();
6
7     public Casa(){
8         casa.setTipo("CASA");
9     }
10
11     public void constroi(){
12         System.out.println("FAZ-SE " + casa.getTipo());
13     }
14
15 }
16
```

# Polimorfismo



The image shows a screenshot of an IDE with several open files: 'Calcula\_are...', 'Construcao.java', 'Casa.java', '\*Apartament...', and 'Empresa\_de\_C...'. The 'Construcao.java' file is active, displaying the following Java code:

```
1 package construção_civil;
2
3 public class Apartamento extends Construção {
4     Construção apartamento = new Construção();
5
6     Apartamento(){
7         apartamento.setTipo("APARTAMENTO");
8     }
9
10    public void constroi(){
11        System.out.println("FAZ-SE " + apartamento.getTipo());}
12
13 }
14
15
16
17
```

# Polimorfismo

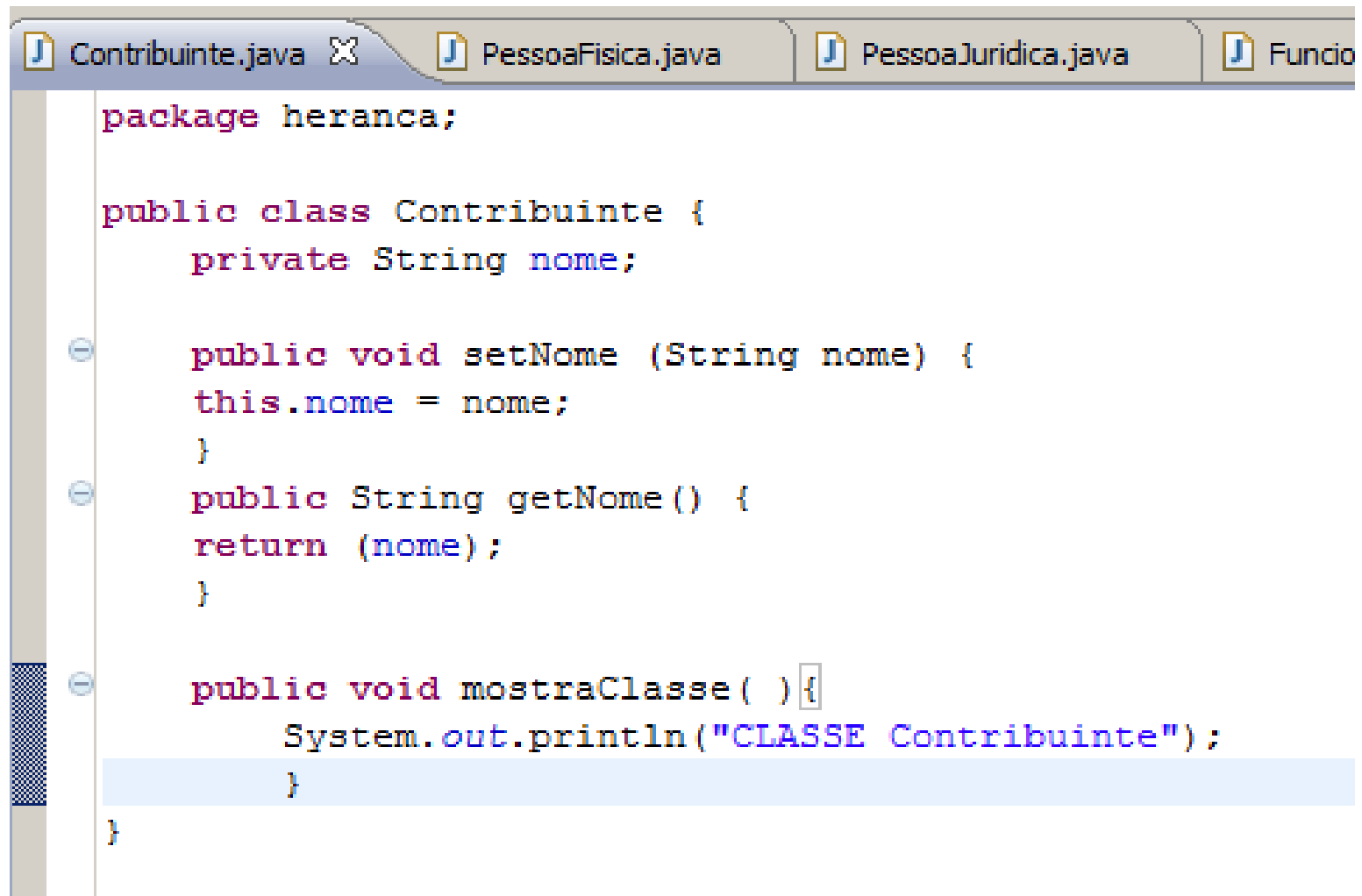
Figuras\_geom... Figuras\_1.java Figuras\_2.java Calcula\_are... Quarto\_Cicl... Construc

```
1 package construção_civil;
2
3 import javax.swing.*;
4 public class Empresa_de_Construcao {
5     public static void main (String args[]){
6
7         Construcao pedreiro = null;
8
9         int tipo_construcao = Integer.parseInt(JOptionPane.showInputDialog("digitar o tipo "
10             + "de construção:" + "\n" + "1 = CASA e 2 = APARTAMENTO ou 3 = tudo" ));
11
12         switch(tipo_construcao){
13             case 1:
14                 pedreiro = new Casa(); break;
15
16             case 2:
17                 pedreiro = new Apartamento(); break;
18
19             case 3:
20                 pedreiro = new Construcao(); break;
21
22             default:
23                 System.out.println("CONSTRUÇÃO INDEFINIDA");
24                 System.exit(0); } //fim do switch case
25
26         pedreiro.constroi();
27     } //fim do void main()
28 }
29
```

# Polimorfismo

1. Construir a aplicação de “construção de imóveis” e executá-la. Em que método da aplicação é utilizado o conceito de Polimorfismo?
2. Construir as classes Contribuinte, PessoaFisica, PessoaJuridica, Funcionario e Polimorfismo. Executar a classe Polimorfismo com as diferentes opções de tipo e verificar os resultados.

# Polimorfismo



The image shows a screenshot of a Java IDE with four tabs: Contribuinte.java, PessoaFisica.java, PessoaJuridica.java, and Funcao. The main editor window displays the code for the Contribuinte class. The code is as follows:

```
package heranca;

public class Contribuinte {
    private String nome;

    public void setNome (String nome) {
        this.nome = nome;
    }

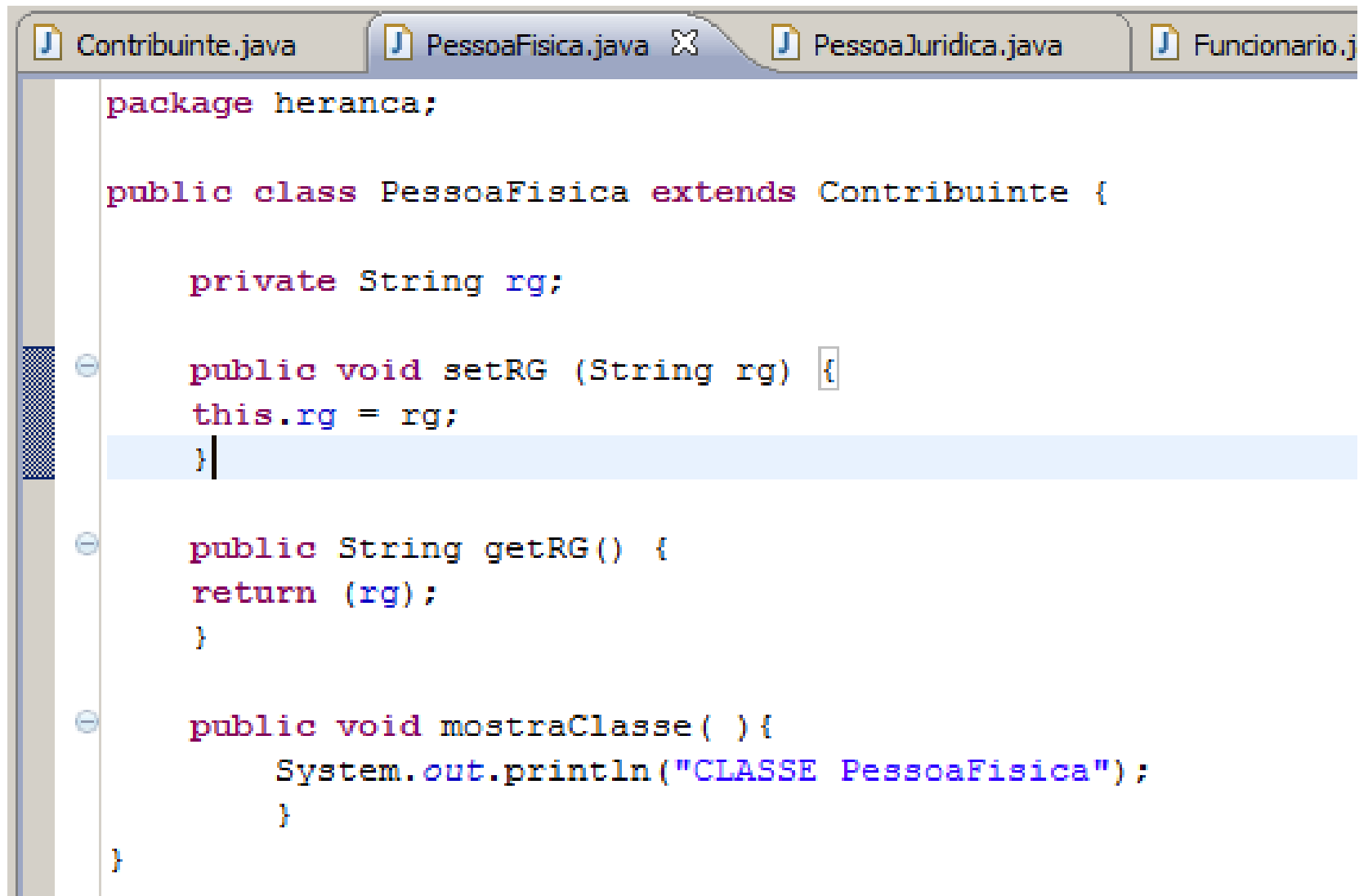
    public String getNome() {
        return (nome);
    }

    public void mostraClasse( ){
        System.out.println("CLASSE Contribuinte");
    }
}
```

The code defines a package named 'heranca' and a public class named 'Contribuinte'. The class has a private attribute 'nome' of type 'String'. It includes three methods: 'setNome' which takes a 'String' parameter and assigns it to 'this.nome'; 'getNome' which returns the value of 'nome'; and 'mostraClasse' which prints 'CLASSE Contribuinte' to the console. The 'mostraClasse' method is currently selected, indicated by a light blue highlight.



# Polimorfismo



```
Contribuinte.java  PessoaFisica.java  PessoaJuridica.java  Funcionario.j

package heranca;

public class PessoaFisica extends Contribuinte {

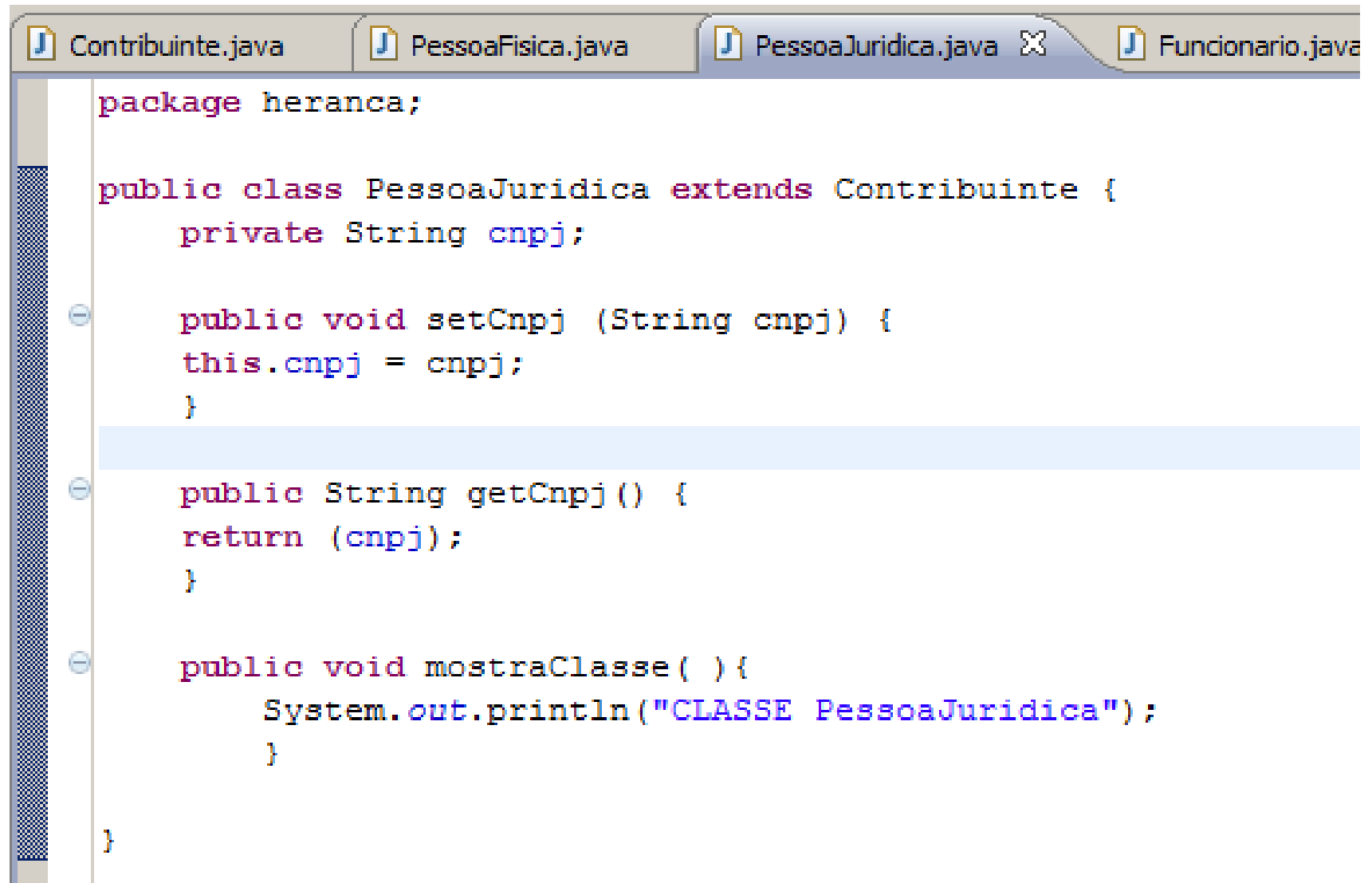
    private String rg;

    public void setRG (String rg) {
        this.rg = rg;
    }

    public String getRG() {
        return (rg);
    }

    public void mostraClasse( ){
        System.out.println("CLASSE PessoaFisica");
    }
}
```

# Polimorfismo



```
package heranca;

public class PessoaJuridica extends Contribuinte {
    private String cnpj;

    public void setCnpj (String cnpj) {
        this.cnpj = cnpj;
    }

    public String getCnpj() {
        return (cnpj);
    }

    public void mostraClasse( ){
        System.out.println("CLASSE PessoaJuridica");
    }
}
```

# Polimorfismo

Contribuinte.java PessoaFisica.java PessoaJuridica.java Funcionario.java X

```
package heranca;

public class Funcionario extends PessoaFisica {
    private String cartao;

    public void setCartao (String cartao) {
        this.cartao = cartao;
    }

    public String getCartao() {
        return (cartao);
    }

    public void mostraClasse( ){
        System.out.println("CLASSE Funcionario");
    }
}
```

# Polimorfismo

```
package heranca;
```

```
import javax.swing.*;
```

```
public class Polimorfismo {
```

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

```
        /*declaração de objeto do tipo da superclasse  
inicializado como "vazio"  
*/
```

```
        Contribuinte pessoa = null;
```

```
        //leitura de variável que definirá como o objeto será instanciado  
        int tipo = Integer.parseInt(JOptionPane.showInputDialog  
            ("digitar opção: valor numérico entre 1 e 4"));
```

```
        //definição de como o objeto será instanciado
```

```
        switch(tipo){
```

```
        case 1:
```

```
            | pessoa = new Contribuinte(); break;//instância pela superclasse
```

```
        case 2:
```

```
            pessoa = new PessoaFisica(); break;//instância pela classe PessoaFisica
```

```
        case 3:
```

```
            pessoa = new PessoaJuridica(); break;//instância pela classe PessoaJuridica
```

```
        case 4:
```

```
            pessoa = new Funcionario(); break;//instância pela classe Funcionario
```

```
            default: {System.out.println("tipo não pertence à família");
```

```
                System.exit(0);}
```

```
        }//fim do switch()  
        pessoa.mostraClasse();
```

```
    }//fim do void main()
```

```
}
```