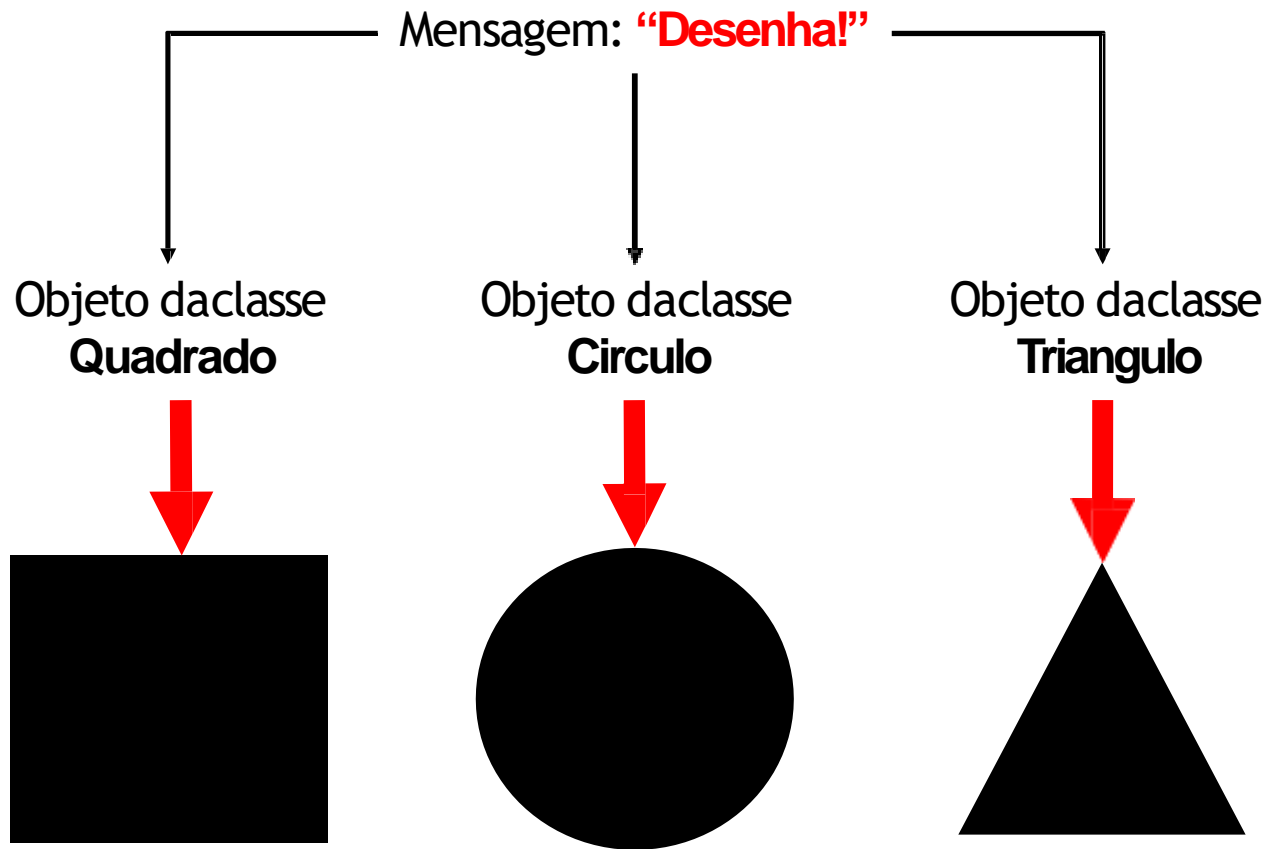


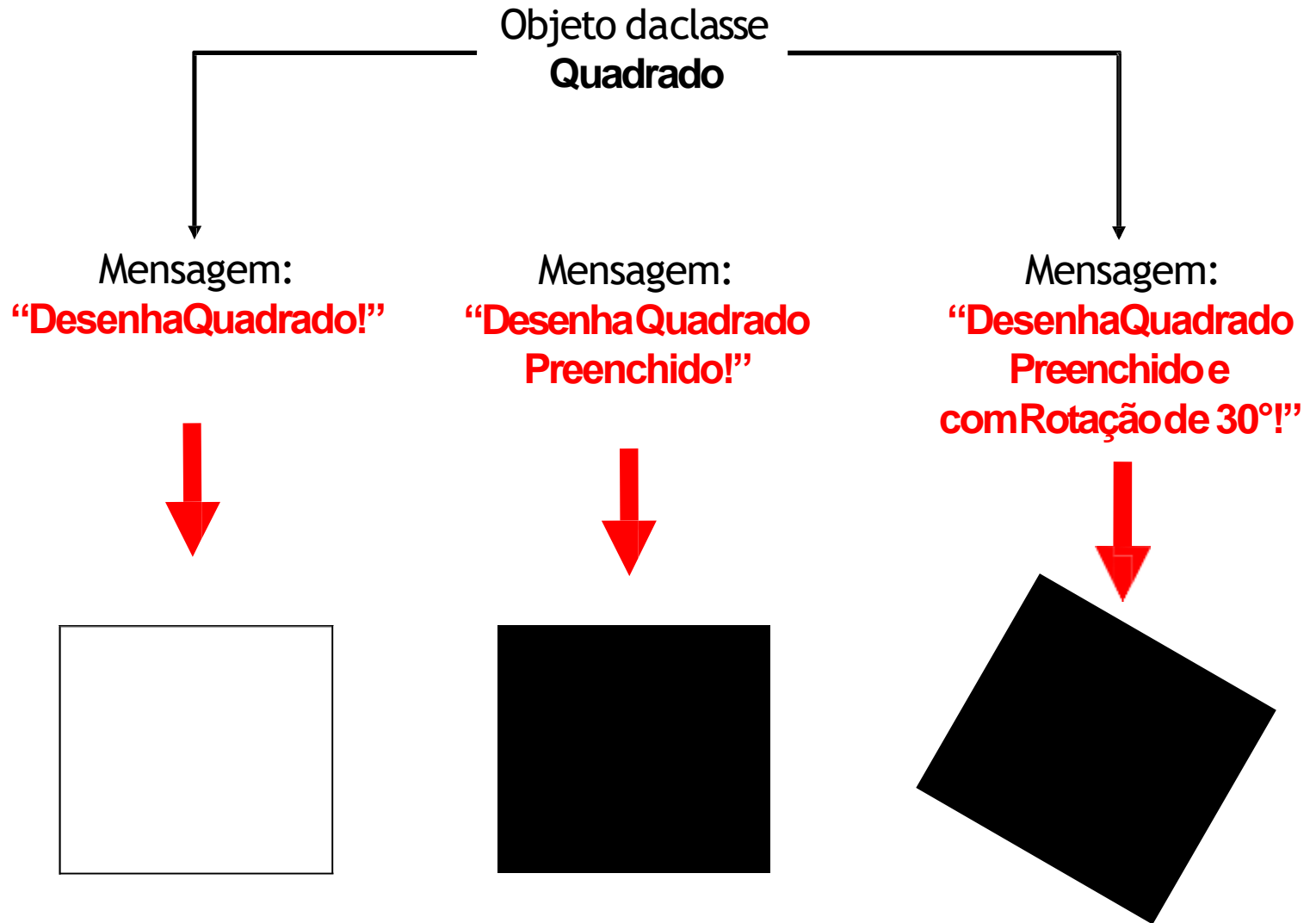
Polimorfismo em Java

Prof. Me. Gustavo Molina
2021

Introdução



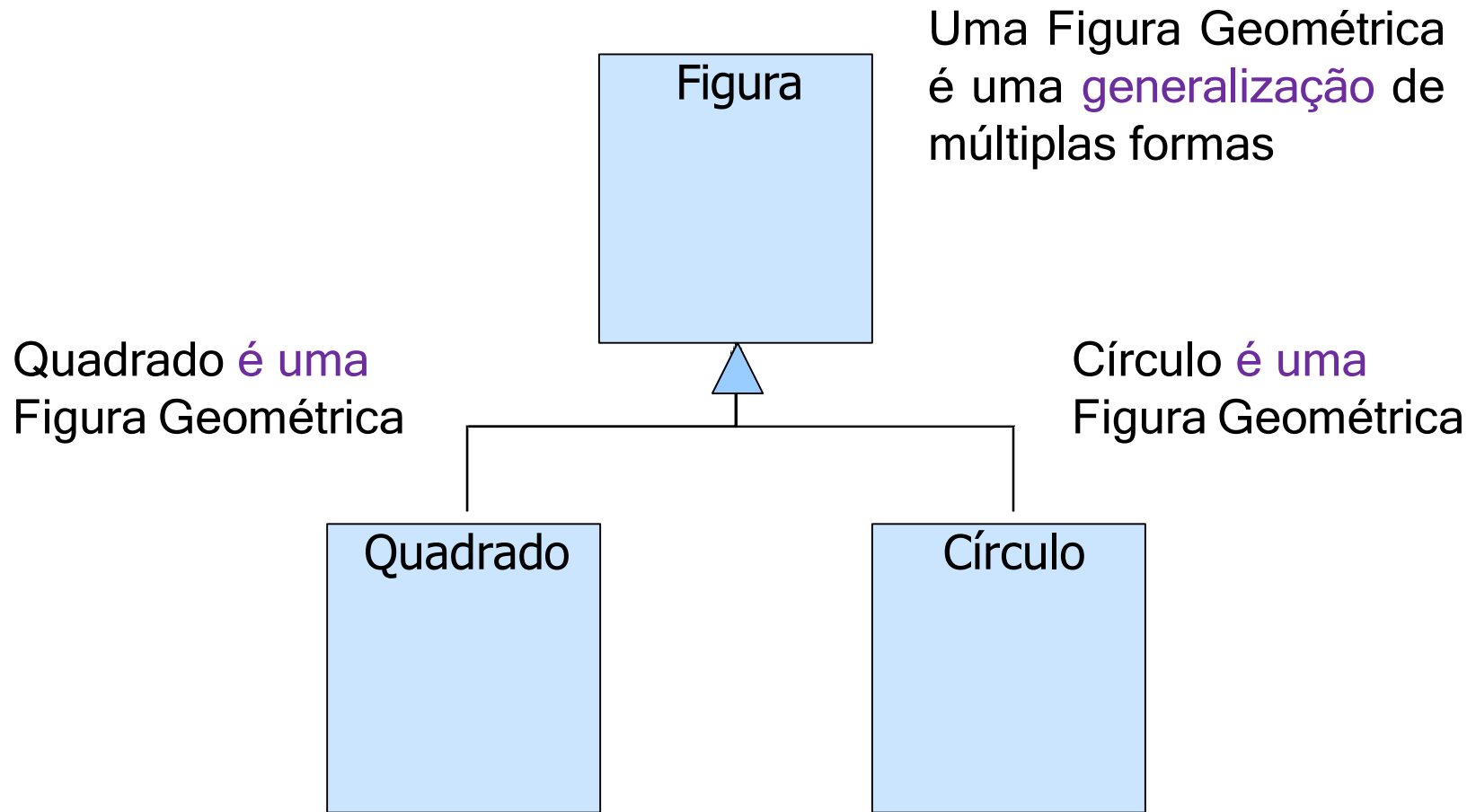
Introdução



Polimorfismo e Herança

- O mecanismo de **herança** permite a criação de classes a partir de outras já existentes desde que exista a relação “**é um**” entre a subclasse e a superclasse.
- Dessa forma é possível criar classes mais especializadas a partir de uma classe genérica.
- A relação “**é um**” entre classes também permite a existência de outra característica fundamental das linguagens OO que é o **polimorfismo**.

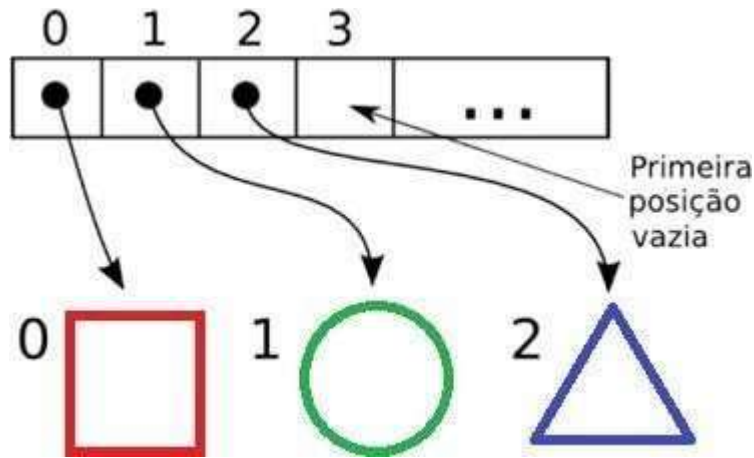
Polimorfismo e Herança



Polimorfismo

- Polimorfismo, que significa “**múltiplas formas**”, permite a manipulação de instâncias de classes que herdam de uma mesma classe ancestral de forma unificada:
 - Assim, é possível escrever métodos que recebam instâncias de uma classe C, e os mesmos métodos serão capazes de processar instâncias de qualquer classe que herde de C, já que qualquer classe que herde de C “é um” C.

Exemplo de Polimorfismo



Quadrado q = **new** Quadrado(2.0);

Circulo c = **new** Circulo(2.0);

if(vetor.contem(q))

...

if(vetor.contem(c))

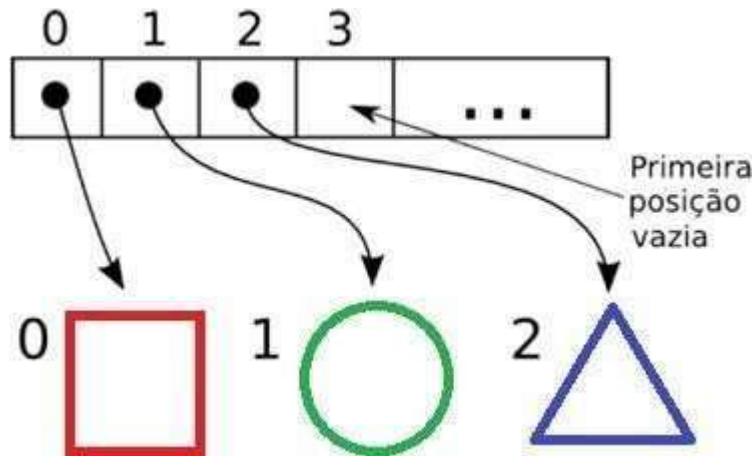
...

```
public class VetorFiguras{
```

```
    private Figura[] figuras = new Figura[10];  
    private int totalDeElementos;
```

```
    public boolean contem(Figura fig) {  
        boolean resultado = false;  
        for(int i = 0; i < this.totalDeElementos; i++){  
            if(fig.equals(this.figuras[i])){  
                resultado = true;  
                break;  
            }  
        }  
        return resultado;  
    }  
}
```

Exemplo de Polimorfismo



```
public class Vetor {
```

```
    private Object[ ] objetos = new Object[10];  
    private int totalDeElementos;
```

```
    public boolean contem(Object obj) {  
        boolean resultado = false;  
        for(int i = 0; i < this.totalDeElementos; i++){  
            if(obj.equals(this.objetos[i])){  
                resultado = true;  
                break;  
            }  
        }  
        return resultado;  
    }  
}
```


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.

Polimorfismo

- Em outras palavras, polimorfismo é a capacidade de se enviar a **mesma mensagem** a **objetos de classes diferentes**, por meio de um mesmo tipo base comum a estes objetos.

Exemplo de Polimorfismo

```
public class Figura {  
    public double calcularArea( ) {  
        return 0;  
    }  
}
```

```
public class Quadrado extends Figura { → Herança  
    double lado;  
  
    public Quadrado(double lado) {  
        this.lado = lado;  
    }
```

```
    public double calcularArea( ) { → Sobreposição do método da  
        double area = 0;           superclasse  
        area = lado * lado;  
        return area;  
    }  
}
```

Exemplo de Polimorfismo

```
public class Circulo extends Figura{
```

 → Herança

```
    double raio;
```

```
    public Circulo (double raio){  
        this.raio = raio;  
    }
```

```
    public double calcularArea( ){  
        double area = 0;  
        area = 3.14 * raio * raio;  
        return area;  
    }
```

```
}
```

 → Sobreposição do método da superclasse.

Exemplo de Polimorfismo

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

```
        Figura f1 = new Quadrado(4);
```

```
        Figura f2 = new Circulo(2);
```

```
        System.out.println("Área da Figura 1 é:"  
                            + f1.calcularArea( ) + "\n"  
                            + "Área da Figura 2 é:"  
                            + f2.calcularArea( ));
```

→ Comportamento polimórfico.

```
    }  
}
```

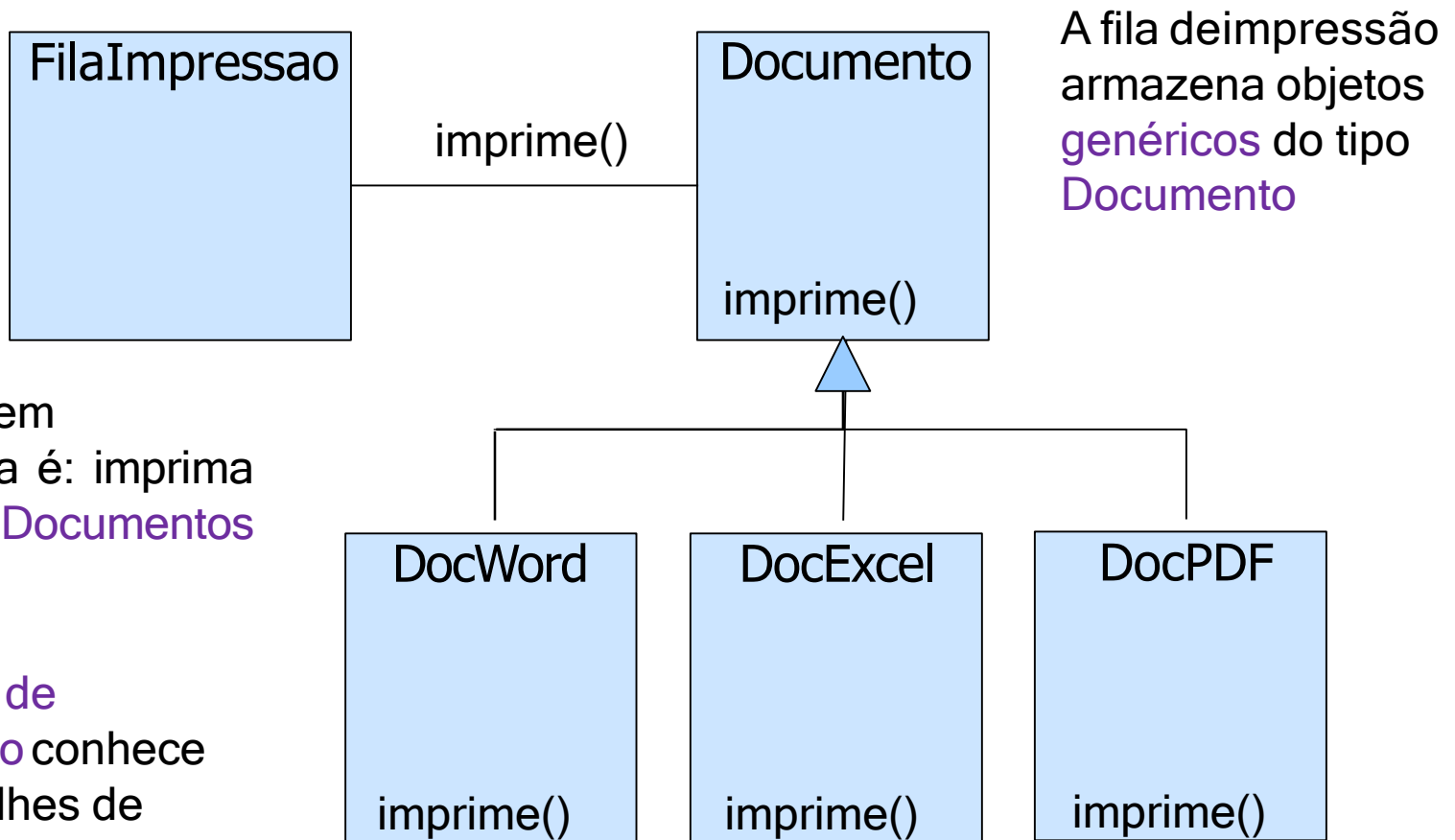
Um mesmo tipo base Figura, por meio das variáveis f1 e f2, é utilizado para enviar uma mesma mensagem calcularArea para objetos de tipos diferentes Quadrado e Circulo e o comportamento executado será distinto.

Exemplo de Polimorfismo

```
public class VetorFiguras {  
  
    private Figura[] figuras = new Figura[10];  
    private int totalDeElementos;  
  
    public double calcularAreaTotal() {  
        double areaTotal = 0;  
        for (int i = 0; i < figuras.length; i++){  
            if (figuras[i] != null) {  
                areaTotal = areaTotal + figuras[i].calcularArea();  
            }  
        }  
        return areaTotal;  
    }  
}
```

→ Comportamento polimórfico.

Outro Exemplo



Amensagem transmitida é: imprima todos os **Documentos** da fila

Cada **tipo de documento** conhece seus detalhes de impressão

Sobrecarga

- **Sobrecarga:** Permite que um “nome de função” possa ser usado mais de uma vez com diferentes tipos de parâmetros.
 - Exemplo: uma função soma com 2 parâmetros inteiros e uma função soma com 2 parâmetros reais. A informação sobre os tipos dos parâmetros é usada para selecionar a função apropriada.

Sobrecarga - Exemplos em Java

- Sobrecarga de métodos construtores:
 - `public ContaCorrente (); // construtor default`
 - `public ContaCorrente (String nome, float val, int num, int pwd) { ... }`
- Sobrecarga de Operadores: quando um operador da linguagem pode ter diferentes significados, dependendo do tipo do parâmetro aplicado.
- Exemplo: `a + = b`
 - *Significado (1): “adiciono valor b ao atributo a”.*
 - *Significado (2): “incluo elemento b no conjunto a”.*
- Java não permite sobrecarga de operadores, apenas de métodos.

Sobrecarga

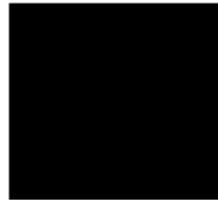
Quadrado



```
public void desenhaQuadrado() { (...) }
```

```
public void desenhaQuadrado(boolean preenchido) { (...) } public
```

```
void desenhaQuadrado(boolean preenchido, double rotacionado)  
{ (...) }
```



Sobrecarga X Redefinição de Métodos

- **Redefinição:** o novo método deve ter a mesma assinatura do método herdado, isto é, eles devem ser idênticos quanto ao nome da operação e à lista de parâmetros (mesmo número de parâmetros, com os mesmos tipos e declarados na mesma ordem).
- O tipo do resultado de retorno não faz parte da assinatura do método e não pode ser mudado.
- **Sobrecarga:** Ocorre quando existe apenas coincidência nos nomes dos métodos; isto é, as listas de parâmetros não são idênticas.

Vantagens do Polimorfismo

- Já vimos que o grande benefício do polimorfismo é permitir que vários objetos de um mesmo tipo base sejam tratados da mesma maneira.
- Uma outra vantagem é permitir aumentar um software de maneira mais controlada, mais localizada.

Vantagens do Polimorfismo

- Considere o exemplo da fila de impressão. Se quisermos incrementar o software e permitir que novos tipos de documentos sejam impressos, a classe **FilaImpressao** não precisa ser alterada.
- Somente novas classes precisam ser criadas para implementar os novos tipos de documentos. Assim, o trabalho é menor e mais localizado, evitando que erros de programação sejam inseridos na classe FilaImpressao.

Vida de Programador



QUANDO VÁRIOS POWER RANGERS
MORFAM AO MESMO TEMPO



CHAMAMOS ISSO DE
POLIMORFISMO

