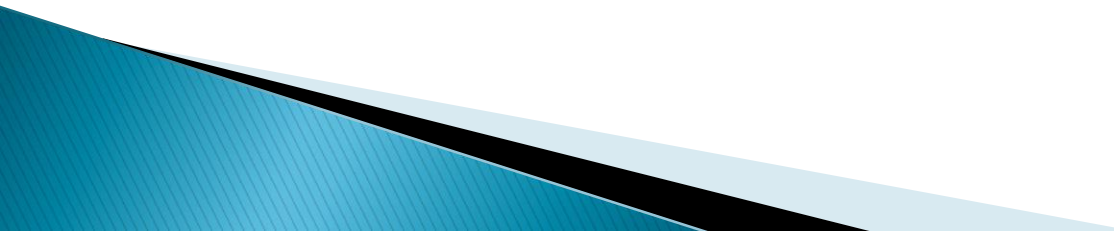


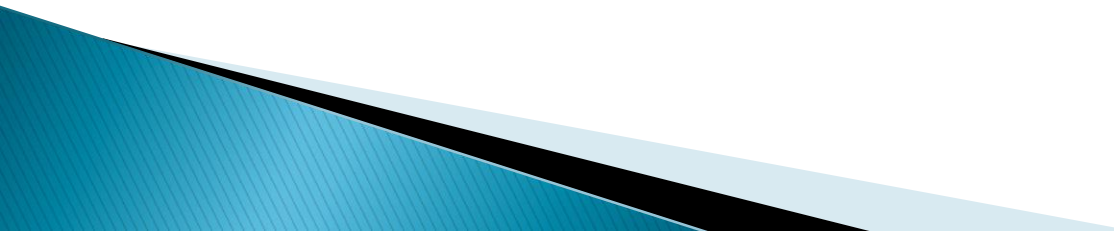
Estrutura de Dados

Prof. MSc. David Alain do Nascimento
IFPE Campus Garanhuns

Conteúdo programático

- ▶ Revisão geral sobre Orientação a Objetos (OO) e programação
 - ▶ Tipos de estruturas de dados
 - ▶ Arrays, listas, Filas, Pilhas e Árvores
 - ▶ Algoritmos de manipulação das estruturas
 - ▶ Algoritmos de Ordenação
 - ▶ Algoritmos de Busca
- 

Revisão geral sobre OO e programação

- ▶ Condicional (if else e switch case)
 - ▶ Loop (for, while e do while)
 - ▶ Função/Método e parâmetro
 - ▶ Classe
 - ▶ Herança
 - ▶ Interface
- 

Revisão geral sobre OO e programação

► Condicional (if else e switch case)

```
if (valor == 2){  
    /* faça alguma coisa aqui */  
}else{  
    /* faça outra coisa aqui */  
}
```

```
switch (valor){  
case 2:  
    /* faça alguma coisa aqui */  
    break;  
default:  
    /* faça outra coisa aqui */  
    break;  
}
```

Revisão geral sobre OO e programação

- ▶ Loop (for, while e do while)

```
for (int i = 0 ; i < 10 ; i++){  
    /* faça alguma coisa aqui */  
}
```

```
while(i < 10){  
    /* faça alguma coisa aqui */  
}
```

```
do{  
    /* faça alguma coisa aqui */  
}while(i < 10);
```



Revisão geral sobre OO e programação

- ▶ Função/Método e parâmetro

```
int soma(int a, int b){  
    return a + b;  
}
```

Revisão geral sobre OO e programação

- ▶ Classe
 - Tipo primitivo e objeto
 - Atributo
 - Método
- ▶ Tipos primitivos:
 - boolean, char, byte, short, int, long, float e double
- ▶ Objetos:
 - Tipos representado por uma classe

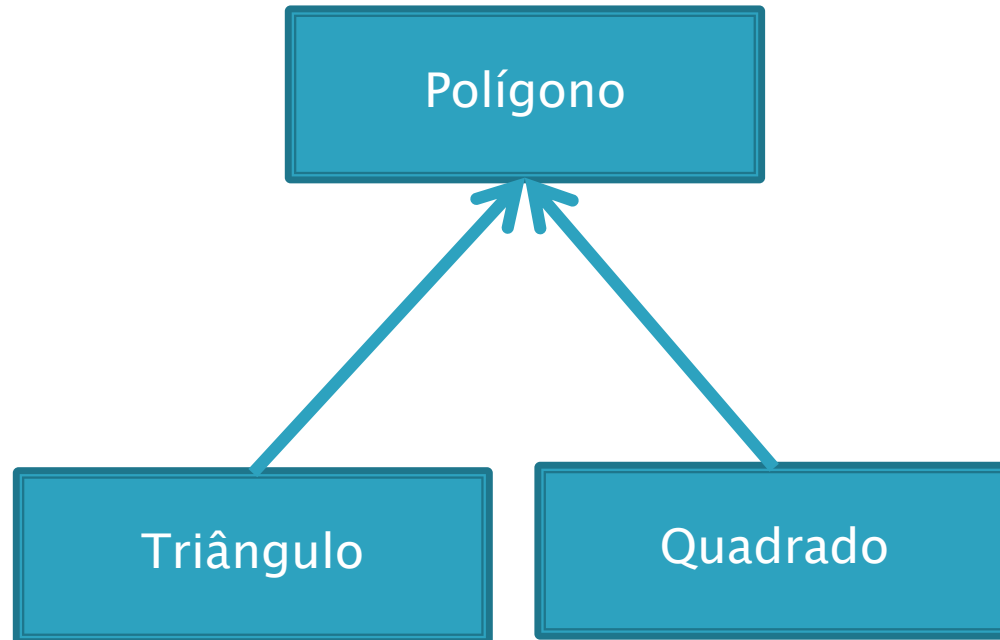
Revisão geral sobre OO e programação

► Exemplo:

```
public class Poligono {  
  
    protected int lados;  
  
    public Poligono(int lados){  
        this.lados = lados;  
    }  
  
    public int getLados() {  
        return lados;  
    }  
    public void setLados(int lados) {  
        this.lados = lados;  
    }  
}
```


Revisão geral sobre OO e programação

► Herança

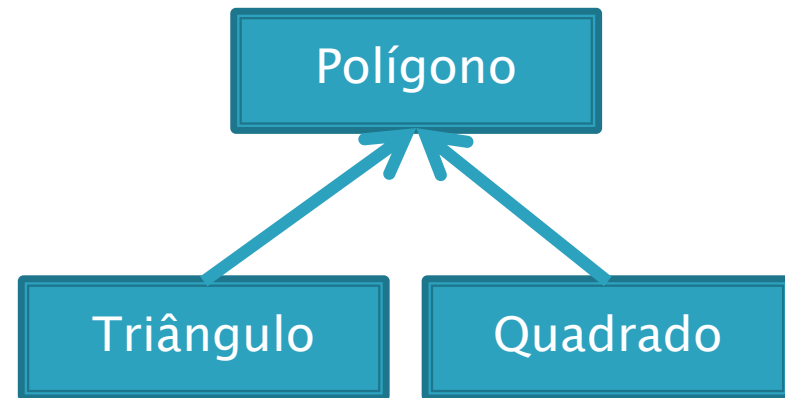


Revisão geral sobre OO e programação

► Herança

```
public class Triangulo extends Poligono{  
    public Triangulo(){  
        super(3);  
    }  
}
```

```
public class Quadrado extends Poligono{  
    public Quadrado(){  
        super(4);  
    }  
}
```



Revisão geral sobre OO

- ▶ Nova necessidade:
 - Método:
 - `String getNome()`
 - Se 3 lados, deverá retornar: “Triângulo”
 - Se 4 lados, deverá retornar: “Quadrado”
 - Como implementar?

Revisão geral sobre OO

- ▶ Nova necessidade:
 - Método:
 - String getNome()

```
public abstract class Poligono {  
  
    protected int lados;  
  
    public Poligono(int lados){  
        this.lados = lados;  
    }  
  
    public int getLados() {  
        return lados;  
    }  
  
    public void setLados(int lados) {  
        this.lados = lados;  
    }  
  
    public String getNome(){  
  
        if(lados == 3){  
            return "Triângulo";  
        }else if(lados == 4){  
            return "Quadrado";  
        }else{  
            return "Erro";  
        }  
  
    }  
  
}
```

Revisão geral sobre OO

- ▶ Nova necessidade:
 - Método:
 - String getNome()

Errado!!

```
public abstract class Poligono {  
    protected int lados;  
    public Poligono(int lados){  
        this.lados = lados;  
    }  
    public int getLados(){  
        return lados;  
    }  
    public void setLados(int lados) {  
        this.lados = lados;  
    }  
    public String getNome(){  
        if(lados == 3){  
            return "Triângulo";  
        }else if(lados == 4){  
            return "Quadrado";  
        }else{  
            return "Erro";  
        }  
    }  
}
```

Revisão geral sobre OO

- ▶ Solução com **Classe abstrata e Método abstrato**
 - Classe: Poligono
 - Método: String getNome()

Revisão geral sobre OO

- ▶ Classe abstrata e Método abstrato
 - Classe: Poligono
 - Método: String getNome()

```
public abstract class Poligono {  
  
    protected int lados;  
  
    public Poligono(int lados){  
        this.lados = lados;  
    }  
  
    public int getLados() {  
        return lados;  
    }  
    public void setLados(int lados) {  
        this.lados = lados;  
    }  
  
    public abstract String getNome();  
  
}
```

```
public class Quadrado extends Poligono{  
  
    public Quadrado(){  
        super(4);  
    }  
  
    public String getNome(){  
        return "quadrado";  
    }  
  
}
```

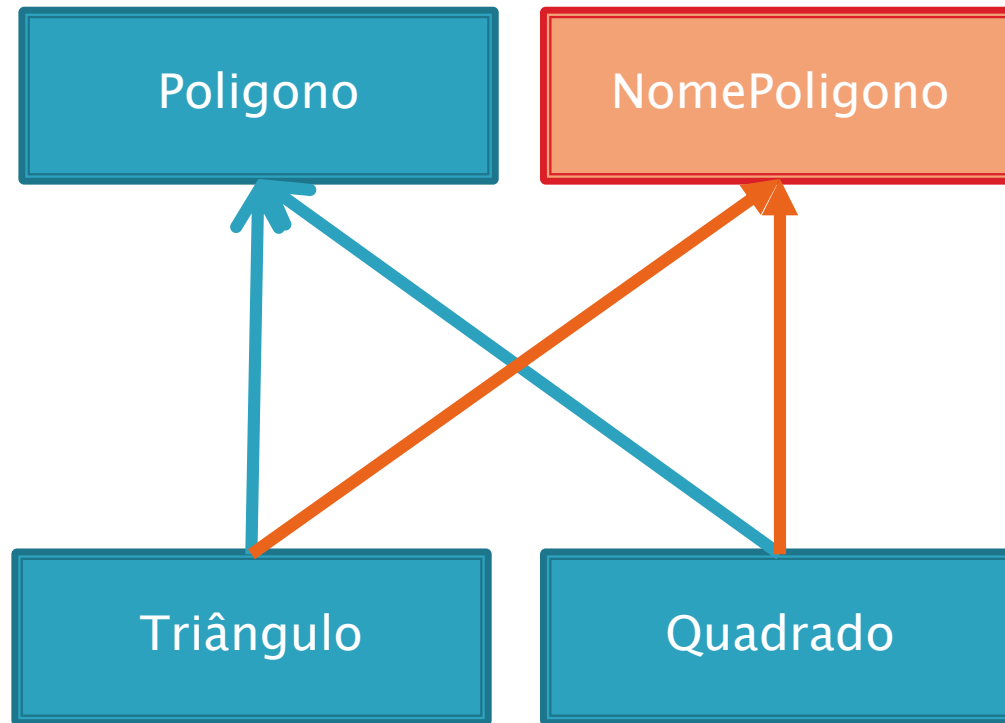
```
public class Triangulo extends Poligono{  
  
    public Triangulo(){  
        super(3);  
    }  
  
    public String getNome(){  
        return "Triangulo";  
    }  
  
}
```

Revisão geral sobre OO

- ▶ Solução com **Interface**
 - Interface: NomePoligono
 - Método: String getNome()

Revisão geral sobre OO

- ▶ Interface NomePoligono
 - Método: String getNome()



Revisão geral sobre OO

- ▶ Interface NomePoligono
 - Método: String getNome()

```
public interface NomePoligono {  
    public String getNome();  
}
```

```
public abstract class Poligono {  
    protected int lados;  
  
    public Poligono(int lados){  
        this.lados = lados;  
    }  
  
    public int getLados() {  
        return lados;  
    }  
  
    public void setLados(int lados) {  
        this.lados = lados;  
    }  
}
```

```
public class Triangulo extends Poligono implements NomePoligono{  
    public Triangulo(){  
        super(3);  
    }  
  
    public String getNome(){  
        return "Triângulo";  
    }  
}
```

```
public class Quadrado extends Poligono implements NomePoligono{  
    public Quadrado(){  
        super(4);  
    }  
  
    public String getNome(){  
        return "Quadrado";  
    }  
}
```