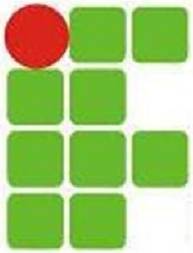
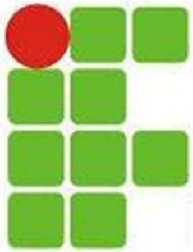
Programação Orientada a Objetos

Professor: Diego Oliveira

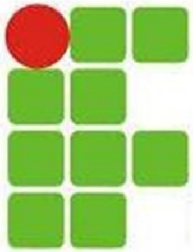


Conteúdo 12: Interfaces



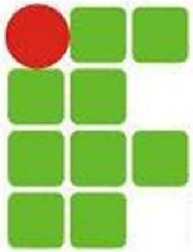
Interfaces

- É um recurso utilizado para definir os métodos e propriedades de um determinado grupo de classes
- Funcionam como um **contrato**, e todas as classes que participam do contrato deverão ter aqueles métodos e propriedades
- O funcionamento do método em cada classe pode ser diferente, contanto que o método exista da maneira definida



Interfaces

- Dentro das interfaces existem apenas as assinaturas dos métodos e propriedades
- Cabe à classe realizar a implementação das assinaturas, dando comportamento concreto aos métodos
- Mais de uma classe pode implementar a mesma interface
- Uma classe pode implementar mais de uma interface



Interfaces

- Exemplo de interface de uma Figura Geométrica:

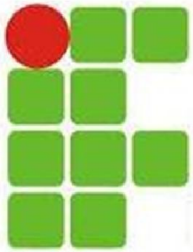
```
1   package interfaces;
2
    public interface FiguraGeometrica {
        public String getNomeFigura();
        public int getArea();
        public int getPerimetro();
7   }
8
```

- Sim, os métodos são vazios! As classes que implementarão esta interface é que dirão como os métodos funcionarão. Vejamos:

Interfaces

- Exemplo de Quadrado que implementa uma Figura Geométrica:

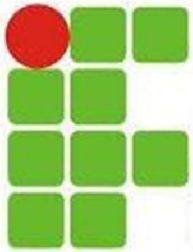
```
5   public class Quadrado implements FiguraGeometrica{
6       private String nomeQuadrado;
7       private int lado;
8       @Override
9       public String getNomeFigura() {
10          return this.nomeQuadrado;
11      }
12      @Override
13      public int getArea() {
14          return lado*lado;
15      }
16      @Override
17      public int getPerimetro() {
18          return lado*4;
19      }
20  }
```



Interfaces

- Outro exemplo, agora um Triângulo que implementa uma Figura Geométrica:

```
5   public class Triangulo implements FiguraGeometrica{
6       private String nomeTriangulo;
7       private int ladoA;
8       private int ladoB;
9       private int ladoC;
10      private int base;
11      private int altura;
12      @Override
        public String getNomeFigura() {
14          return nomeTriangulo;
15      }
16      @Override
        public int getArea() {
18          return (base*altura)/2;
19      }
20      @Override
        public int getPerimetro() {
22          return ladoA+ladoB+ladoC;
23      }
24  }
```

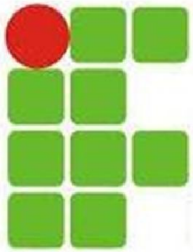


Interfaces

- Agora vejamos um exemplo que implementa mais de uma interface:

```
1     package interfaces;
      public interface Motor {
          public int getRPM();
          public double getPotencia();
5     }
```

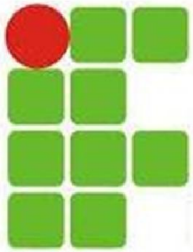
```
1     package interfaces;
      public interface Veiculo {
          public String getTipo();
          public double getPeso();
5     }
```



Interfaces

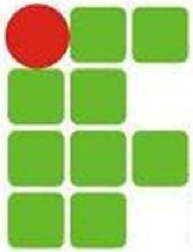
- O carro modelo CIVIC é um ***Veículo*** e também possui um ***Motor***:

```
1   package classes;
2   import interfaces.Motor;
3   import interfaces.Veiculo;
4   public class Civic implements Motor, Veiculo{
5       @Override
6       public int getRPM() {
7           return 8000;
8       }
9       @Override
10      public double getPotencia() {
11          return 139.0;
12      }
13      @Override
14      public String getTipo() {
15          return "veículo urbano";
16      }
17      @Override
18      public double getPeso() {
19          return 1650.0;
20      }
21  }
```



Exercício

- Crie três interfaces que possuam relação entre si, de alguma maneira
- Crie duas assinaturas de método para cada uma das interfaces
- Crie uma classe que implementa as três interfaces
- Reescreva os métodos para que eles possuam uma função concreta
- Imprima os resultados



Perguntas?