Roteiro 9

Neste roteiro trabalharemos com o conceito de **Interface**.

Interface é um recurso da linguagem Java que apresenta inúmeras vantagens no sentido da modelagem e instanciação de objetos, porém deve-se entender claramente os conceitos básicos da orientação a objetos a fim de utilizá-la plenamente.

Uma interface é **similar a um contrato**. Com o uso de uma interface podemos “obrigar” que os métodos descritos nesta interface sejam implementados em um grupo de classes que desejamos.

Pode-se dizer, de forma ilustrativa, que uma interface é um contrato que quando assumido por uma classe deve ser implementado.

Dentro de uma interface existem apenas assinaturas de métodos (sem implementação). Podem conter também propriedades em forma de constantes. Então, pode-se dizer que uma classe que “assume o contrato” com uma determinada interface, tem “obrigação” de implementar os métodos descritos nesta interface.

Agora iremos aplicar estes conceitos durante o roteiro.

Parte 1 (roteiro9.parte1) – Interface

Cenário :

Vamos imaginar a modelagem de um sistema onde seja necessário criar diversos componentes visuais no formato de figuras geométricas como quadrado, retângulo, círculo e etc.

Sabe-se que estas figuras possuem características especificas de cada uma delas (atributos). Exemplo : lado, altura, largura, raio.

Utilizando-se dos atributos de cada uma das figuras queremos implementar métodos comuns a todas elas como por exemplo : calcular a área e calcular o perímetro. Além de implementar estes métodos, queremos que todas as figuras possuam obrigatoriamente estes métodos.

1 – Crie o pacote **roteiro9.parte1** com a classes **Principal**.

2 – Crie agora a interface **FiguraGeometrica** conforme o código abaixo.

Uma interface é um arquivo com a extensão .java semelhante a uma classe qualquer. Diferenciamos uma da outra apenas pela sintaxe dentro do código.
No lugar de `public class` temos `public interface`

Na IDE do Netbeans podemos criar diretamente em New -> Java Interface

```
public interface FiguraGeometrica {

    public String getNomeFigura();
    public double getArea();
    public double getPerimetro();

}
```

Observe que nesta interface temos 3 métodos sem implementação.

3 – Crie as classes Quadrado e Retangulo conforme o código abaixo

```
public class Quadrado {

    private double lado;

    public Quadrado (int lado){
        this.lado = lado;
    }


    {Gets e Sets}

}
```

```
public class Retangulo {

    private double altura;
    private double largura;


    public Retangulo (double altura, double largura){
        this.altura = altura;
        this.largura = largura;
    }

    {Gets e Sets}

}
```

4 – Faça com que a classe **Retangulo** implemente a interface **FiguraGeometrica.**

Neste momento é a “assinatura do contrato”. A classe Retangulo obrigatoriamente terá que implementar as classes da interface.

Ao fazer a alteração conforme no código abaixo, provavelmente um erro será indicado em tela, pois os métodos ainda não foram implementados.

OBS.: Algumas IDEs geram os métodos da interface automaticamente e o desenvolvedor precisa apenas implementá-los. Vale a pena conferir.

```
public class Quadrado implements FiguraGeometrica {

    private double lado;

    public Quadrado (int lado){
        this.lado = lado;
    }


    {Gets e Sets}

}
```

5 – Implemente na classe **Quadrado** os métodos indicados na interface conforme o trecho de código abaixo.

```
public class Quadrado implements FiguraGeometrica {

    private double lado;

    public Quadrado (int lado){
        this.lado = lado;
    }


    {Gets e Sets}


    @Override
    public String getNomeFigura() {
        return "Quadrado";
    }

    @Override
    public double getArea() {
        return this.lado*this.lado;
    }

    @Override
    public double getPerimetro() {
       return this.lado * 4;
    }


}
```

6 – Podemos dizer que esta técnica faz uso de Polimorfismo ? Sim/Não ? Explique

7 – Faça as adaptações necessárias para que a classe **Retangulo** implemente a interface **FiguraGeometrica.**

8 – Faça os testes na classe **Principal** utilizando os métodos implementados para as 2 figuras criadas.

Parte 2 (roteiro9.parte2) – Exercício

1 – Crie o pacote **roteiro9.parte2** com a cópia das classes implementados na parte1.

2 – Crie uma nova classe chamada **Circulo** em que tenha como atributo o raio e faça com que ela implemente a interface **FiguraGeometrica.**

3 – Faça os testes na classe **Principal** utilizando os métodos implementados para as 3 figuras criadas.

4 – Utilize o software StarUML para construir o diagrama de classes fazendo engenharia reversa do código.