Conteúdo Adicional
Java

Interface

- Recurso muito utilizado para forçar um determinado grupo de classes a ter métodos ou propriedades em comum para existir em um determinado contexto

- Contudo, os métodos podem ser implementados em cada classe de uma maneira diferente

- Uma interface é como um contrato que quando assumido por uma classe deve ser implementado.

Interface

- Dentro das interfaces existem somente assinaturas de métodos e propriedades
- cabendo à classe que utiliza a interface realizar a implementação das assinaturas, dando comportamentos práticos aos métodos.
- interface FiguraGeometrica com três assinaturas de métodos que virão a ser implementados pelas classes referentes às figuras geométricas.

```
public interface FiguraGeometrica
{
        public String getNomeFigura();
        public int getArea();
        public int getPerimetro();
}
```

Interface

- Necessário adicionar a palavra-chave implements ao final da assinatura da classe que irá implementar a interface escolhida.

- **Sintaxe:**
  public class nome_classe implements nome_interface

Interface

- Pode-se ver duas classes que implementam a interface FiguraGeometrica, uma chamada Quadrado e outra Triangulo.

```
public class Quadrado implements FiguraGeometrica {
    private int lado;
    public int getLado() {
        return lado;
    }
    public void setLado(int lado) {
        this.lado = lado;
    }
    public int getArea() {
        int area = 0;
        area = lado * lado;
        return area;
    }
     public int getPerimetro() {
        int perimetro = 0;
        perimetro = lado * 4;
        return perimetro;
    }
     public String getNomeFigura() {
        return "quadrado";
    }
}
```

Interface

```
public class Triangulo implements FiguraGeometrica {


    private int base;
    private int altura;
    private int ladoA;
    private int ladoB;
    private int ladoC;


    public int getAltura() {
        return altura;
    }
    public void setAltura(int altura) {
        this.altura = altura;
    }
public int getBase() {
        return base;
    }
    public void setBase(int base) {
        this.base = base;
    }
    public int getLadoA() {
        return ladoA;
    }
    public void setLadoA(int ladoA) {
        this.ladoA = ladoA;
    }
    public int getLadoB() {
        return ladoB;
    }
public void setLadoB(int ladoB) {
        this.ladoB = ladoB;
    }
    public int getLadoC() {
        return ladoC;
    }
    public void setLadoC(int ladoC) {
        this.ladoC = ladoC;
    }
     public String getNomeFigura() {
        return "Triangulo";
    }
     public int getArea() {
        int area = 0;
        area = (base * altura) / 2;
        return area;
    }
    public int getPerimetro() {
        int perimetro = 0;
        perimetro = ladoA + ladoB +
ladoC;


        return perimetro;
    }
}
```

Interface

- Ambas as classes seguiram o contrato da interface FiguraGeometrica
- Porém, cada uma delas a implementou de maneira diferente.
- É possível que uma classe implemente varias interfaces ao mesmo tempo.
- Exemplo:

```
public class Carro implements Veiculo, Motor {
```

Variável Static

- modificador usado em variáveis e métodos dentro de uma classe
- Parecido com o conceito de variável global
- Pode-se acessar variáveis de uma classe sem ter de instanciar o objeto.

```
public class Classe1{

static int valor;

}
```

```
public class Classe2{

    public static void main(String[] args) {

    Classe1.valor = 10;

    }

}
```

Classes Abstratas

- Classes abstratas não permitem realizar qualquer tipo de instância
- Feitas especialmente para serem modelos para suas classes derivadas
- As classes derivadas devem **sobrescrever os métodos** para realizar a implementação dos mesmos
- Classes derivadas de classes abstratas são conhecidas como classes concretas.

Classes Abstratas

- Classes abstratas somente podem ser estendidas
- Não se cria um objeto a partir da mesma
- Caso um ou mais métodos abstratos estejam presentes em uma classe abstrata, a classe filha deverá definir tais métodos
- Se os métodos da classe mãe não forem definidos, a classe filha também se tornará abstrata
- **Métodos Abstratos** estão presentes somente em classes abstratas, e são aqueles que **não possuem implementação**

Classes Abstratas

```
abstract class Pessoa
{
 public string Nome;
 public int Idade;
 public string Email;

   public abstract void Gravar();
}
```

```
class Funcionario : Pessoa
{
   public override void Gravar()
   {
      Nome = "Wellington";
      Idade = 21;
      Email = "wellingtonbalbo@gmail.com";
   }
}
```

Por meio da palavra-chave **override** (apenas utilizada em métodos e atributos virtuais) sobrescreve-se o método **Gravar**, criado na classe **Pessoa**

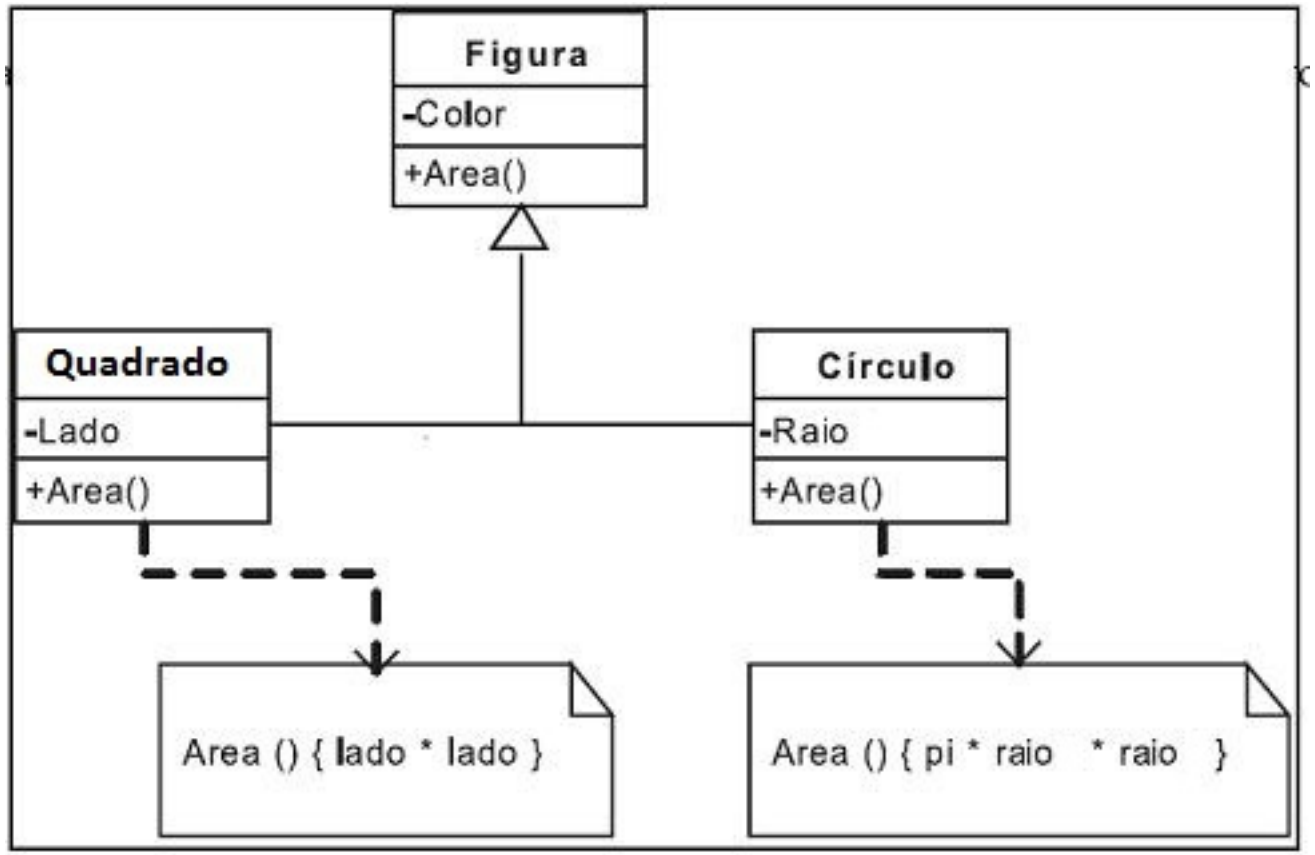
Métodos abstratos são implicitamente **virtuais**

Polimorfismo

- Permite que classes abstratas recebam comportamentos através de classes concretas
- Exemplo:
  - Dispositivo USB seria uma classe abstrata
  - Pen Driver, iPad, Câmeras, etc. seriam classes concretas

  - USB é uma especificação que permite várias implementações com características diferentes

Polimorfismo



Métodos Get e Set

- Os **métodos GET e SET** são técnicas padronizadas para gerenciamento sobre o acesso dos atributos
- Determina o acesso e quando será alterado um atributo
- Torna o controle e modificações mais práticas e limpas, sem contudo precisar alterar assinatura do método usado para acesso ao atributo
- Características essencial de um módulo, é a capacidade de ocultar todos os seus detalhes de implementação.
- Dá aos módulos uma comunicação somente através da sua API sem que um módulo não precise conhecer o funcionamento interno dos outros módulos.

Métodos Get e Set

```
public class Ponto {

    private double x;
    private double y;

    public double getX() {
        return x;
    }
    public void setX(double x) {
        this.x = x;
    }
    public double getY() {
        return y;
    }
    public void setY(double y) {
        this.y = y;
    }
}
```