Programação orientada a objetos

Classes abstratas
Interfaces

Classe abstrata

- Classe abstrata em Java é uma classe que não pode ser instanciada, ou seja, não é possível criar objetos a partir dela.
- Em vez disso, ela é projetada para ser estendida por outras classes que podem ser instanciadas.
- Para criar uma classe abstrata em Java, você deve usar a palavra-chave "abstract" antes da palavra "class" na declaração da classe.
- Além disso, você pode definir métodos abstratos dentro da classe abstrata. Um método abstrato é um método sem implementação que deve ser implementado por qualquer classe que estenda a classe abstrata.

Classe abstrata – Exemplo 1

- A classe "Animal" não contém nenhuma implementação do método emiteSom(), apenas sua assinatura. Isso significa que qualquer classe que estenda "Animal" deve implementar o método emiteSom().

```
3 public abstract class Animal {
4     public String nome;
5     public int idade;
6     public abstract void emiteSom();
7 }
```

Classe abstrata – Exemplo 1

- A classe “Cachorro" que estende a classe "Animal" e implementa o método emiteSom().

```
3 public class Cachorro extends Animal{
4     @Override
5     public void emiteSom() {
6         System.out.println("au au");
7         System.out.println("Nome: "+nome);
8         System.out.println("Idade: "+idade);
9     }
10 }
```

Classe abstrata – Exemplo 1

```
3 public class Principal {
4     public static void main(String[] args) {
5         Animal c = new Cachorro();
6         c.emiteSom();
7     }
8 }
```

- As classes abstratas são úteis quando você deseja definir um comportamento padrão que deve ser implementado por outras classes.
- Elas permitem definir uma interface comum para as classes que as estendem e garantir que essas classes implementem determinados métodos.

Classe abstrata – Exemplo 1

- Jogo rápido:
  - Implemente que todos os animais contenham atributos como espécie (String) e idade (int)

Classe abstrata – Exemplo 1

- Jogo rápido:
  - Implemente que todos os animais contenham atributos como espécie (String) e idade (int)

```
3  public abstract class Animal {
4      public String especie;
5      public int idade;
6      public abstract void emiteSom();
7  }
```

```
3  public class Cachorro extends Animal {
4      public void emiteSom() {
5          System.out.println("Au! Au!");
6      }
7  }
```

```
3  public class Principal {
4      public static void main(String[] args) {
5          Animal c = new Cachorro();
6          c.especie = "Dog";
7          c.idade = 10;
8          c.emiteSom();
9      }
10 }
```

Classe abstrata – Exemplo 1

- Jogo rápido [2]:
  - Implemente um método que apresente os dados do animal:

Classe abstrata – Exemplo 1

- Jogo rápido [2]:
  - Implemente um método que apresente os dados do animal:

```java
public abstract class Animal {
    public String especie;
    public int idade;
    public abstract void emiteSom();

    public void exibeDados() {
        System.out.println("Espécie: "+especie);
        System.out.println("Idade: "+idade);
    }
}
```

```java
public class Cachorro extends Animal {
    public void emiteSom() {
        System.out.println("Au! Au!");
    }
}
```

```java
public class Principal {
    public static void main(String[] args) {
        Animal c = new Cachorro();
        c.especie = "Dog";
        c.idade = 10;
        c.exibeDados();
        c.emiteSom();
    }
}
```

Classe abstrata – Exemplo 1

- Jogo rápido [3]:
  - Implemente um método para que o cachorro cuide do pátio (cuidarPatio()):

Classe abstrata – Exemplo 1

- Jogo rápido [3]:
  - Implemente um método para que o cachorro cuide do pátio (cuidarPatio()):

```java
public abstract class Animal {
    public String especie;
    public int idade;
    public abstract void emiteSom();

    public void exibeDados() {
        System.out.println("Espécie: "+especie);
        System.out.println("Idade: "+idade);
    }
}
```

```java
public class Cachorro extends Animal {
    public void emiteSom() {
        System.out.println("Au! Au!");
    }
    public void cuidarPatio() {
        System.out.println("Estou cuidando do pátio");
    }
}
```

```java
public class Principal {
    public static void main(String[] args) {
        Cachorro c = new Cachorro();
        c.especie = "Dog";
        c.idade = 10;
        c.exibeDados();
        c.emiteSom();
        c.cuidarPatio();
    }
}
```

Classe abstrata – Exemplo 1

- Jogo rápido [4]:
  - Adicione um atributo raça para o Cachorro e reimplemente um método exibeDados para que exiba os dados do cachorro:

Classe abstrata – Exemplo 1

- Jogo rápido [4]:
  - Adicione um atributo raça para o Cachorro e reimplemente um método exibeDados para que exiba os dados do cachorro:

```
3  public class Cachorro extends Animal {
4      String raca;
5
6      public void emiteSom() {
7          System.out.println("Au! Au!");
8      }
9      public void cuidarPatio() {
10         System.out.println("Estou cuidando do pátio");
11     }
12     @Override
13     public void exibeDados() {
14         System.out.println("Espécie: "+especie);
15         System.out.println("Idade: "+idade);
16         System.out.println("Raça: "+raca);
17     }
18 }
```

```
3  public class Principal {
4      public static void main(String[] args) {
5          Cachorro c = new Cachorro();
6          c.especie = "Dog";
7          c.idade = 10;
8          c.raca = "SRD";
9          c.exibeDados();
10         c.emiteSom();
11         c.cuidarPatio();
12     }
13 }
```

Classe abstrata – Exemplo 2

```
3 abstract class Forma {
4     public abstract double area();
5     public abstract double perimetro();
6 }
```

Classe abstrata – Exemplo 2

```
3  class Retangulo extends Forma {
4      private double largura;
5      private double altura;
6
7      public Retangulo(double largura, double altura) {
8          this.largura = largura;
9          this.altura = altura;
10     }
11
12     public double area() {
13         return largura * altura;
14     }
15
16     public double perimetro() {
17         return 2 * (largura + altura);
18     }
19 }
```

Classe abstrata – Exemplo 2

```
3  class Circulo extends Forma {
4      private double raio;
5
6      public Circulo(double raio) {
7          this.raio = raio;
8      }
9
10     public double area() {
11         return Math.PI * raio * raio;
12     }
13
14     public double perimetro() {
15         return 2 * Math.PI * raio;
16     }
17 }
```

Classe abstrata – Exemplo 2

```
3  public class Principal {
4
5      public static void main(String[] args) {
6          Retangulo r = new Retangulo(3,4);
7          System.out.println("Área do retangulo: "+r.area());
8          System.out.println("Perímetro do retangulo: "+r.perimetro());
9
10         Circulo c = new Circulo(5);
11         System.out.println("Área do círculo: "+c.area());
12         System.out.println("Perímetro do círculo: "+c.perimetro());
13     }
14 }
```

Interface

- Uma interface é uma espécie de contrato que define um conjunto de métodos que uma classe deve implementar.
- Em Java, uma interface é definida através da palavra-chave interface.
- A interface permite que diferentes classes possam implementar seus próprios comportamentos para um conjunto de métodos definidos em comum.
- Isso permite que as classes possam ser tratadas de forma polimórfica, ou seja, uma referência a uma interface pode referenciar objetos de diferentes classes que implementam essa interface.

Interface

- Os métodos definidos na interface são por padrão public e abstract, ou seja, são métodos abstratos e públicos.
- Não é necessário especificar esses modificadores de acesso, já que eles são implicitamente definidos.
- Para implementar uma interface em uma classe, utiliza-se a palavra-chave implements.
- A classe deve então implementar todos os métodos definidos na interface.

Interface

```java
public interface Animal {
    void emitirSom();
}
```

```java
public class Cachorro implements Animal {
    public void emitirSom() {
        System.out.println("Au au!");
    }
}
```

```java
public class Gato implements Animal {
    public void emitirSom() {
        System.out.println("Miau!");
    }
}
```

Interface

```
3  public class Prinicpal {
4      public static void main(String[] args) {
5          Animal cachorro = new Cachorro();
6          cachorro.emitirSom(); // Saída: Au au!
7
8          Animal gato = new Gato();
9          gato.emitirSom(); // Saída: Miau!
10     }
11 }
```

Interface

- Jogo rápido:
  - Consigo declarar atributos na interface Animal?

Interface

- Jogo rápido:
  - Consigo declarar atributos na interface Animal?
  - Não? Sim?
    - Podemos somente declarar atributos com a palavra reservada final, ou seja, que não aceitarão alterações (constantes)

```
3 public interface Animal {
4     public int constante = 10;
5     void emitirSom();
6 }
```

Interface

- Jogo rápido [2]:
  - Consigo implementar métodos concretos na interface Animal?
  - Somente se o método for estático (static)

Interface

- Jogo rápido [3]:
  - Implemente um método para que o cachorro cuide do pátio (cuidarPatio()):

Interface

- Jogo rápido [3]:
  - Implemente um método para que o cachorro cuide do pátio (cuidarPatio()):

```
3  public class Cachorro implements Animal {
4      public void emitirSom() {
5          System.out.println("Au au!");
6      }
7      public void cuidarPatio() {
8          System.out.println("Estou cuidando do pátio");
9      }
10 }
```

Interface

- Jogo rápido [4]:
  - Adicione atributos para as classes (nome, idade) e adicione um método exibeDados para que exiba os dados:

Interface

- Jogo rápido [4]:
  - Adicione atributos para as classes (nome, idade) e adicione um método exibeDados para que exiba os dados:

```java
public interface Animal {
    public int constante = 10;
    void emitirSom();
    void exibeDados();
}
```

```java
public class Cachorro implements Animal {
    public String nome;
    public int idade;
    public void emitirSom() {
        System.out.println("Au au!");
        System.out.println("Constante: "+constante);
    }
    public void cuidarPatio() {
        System.out.println("Estou cuidando do pátio");
    }
    @Override
    public void exibeDados() {
        System.out.println("Cachorro!");
        System.out.println("Nome: "+nome);
        System.out.println("Idade: "+idade);
    }
}
```

Interface

- Jogo rápido [4]:
  - Adicione atributos para as classes (nome, idade) e adicione um método exibeDados para que exiba os dados:

```
3  public class Gato implements Animal {
4      public String nome;
5      public int idade;
6      public void emitirSom() {
7          System.out.println("Miau!");
8          System.out.println("Constante: "+constante);
9      }
10
11     @Override
12     public void exibeDados() {
13         System.out.println("Gato!");
14         System.out.println("Nome: "+nome);
15         System.out.println("Idade: "+idade);
16     }
17 }
```

Interface

- Mas analisando este exemplo, qual é a diferença de uma classe abstrata para uma interface?

Interface

- Mas analisando este exemplo, qual é a diferença de uma classe abstrata para uma interface?
  - Uma classe abstrata pode conter métodos abstratos e não abstratos, enquanto uma interface só pode conter métodos abstratos.
  - Uma classe abstrata pode ter construtores, já que é possível instanciar uma classe abstrata por meio de uma classe concreta que a estenda. Já uma interface não pode ter construtores.
  - Uma classe pode implementar várias interfaces, mas só pode herdar de uma classe abstrata.
  - Uma classe abstrata pode conter atributos com diferentes níveis de visibilidade (public, protected, private), já em uma interface os atributos são sempre public e static, e podem ser final ou não.

Interface

- A classe abstrata é utilizada quando se deseja criar uma classe base com algumas implementações e comportamentos já definidos, que serão herdados por suas subclasses, mas que permite que elas implementem comportamentos diferentes.

- A interface é utilizada quando se deseja definir um conjunto de métodos que devem ser implementados por classes distintas, que podem ter diferentes comportamentos, mas que apresentam alguma característica em comum.

- A interface é mais restritiva, pois exige que todas as classes que a implementam implementem todos os seus métodos, enquanto a classe abstrata permite que alguns métodos sejam abstratos e outros não, sendo assim uma opção menos restritiva.

Interface

- Como o Java não permite herança múltipla é possível realizar a herança em cascata (a extends b, b extends c, logo a também extends c)
- Ou...
- Criar interface: que é um conjunto de declarações de métodos (nome, tipo de retorno, tipos dos argumentos) sem implementação.
- Fica a critério do programador que deseja implementar a interface em questão providenciar uma implementação desses métodos na classe que ele está desenvolvendo.

Interface

```java
interface PC {
    public void verificaEmail();
}
```

```java
interface Celular {
    public void realizarChamada();
}
```

```java
public class Smartphone implements Celular, PC{
    String tel;
    String email;

    public Smartphone(String tel, String email) {
        super();
        this.tel = tel;
        this.email = email;
    }

    @Override
    public void verificaEmail() {
        System.out.println("Verificando e-mails");

    }

    @Override
    public void realizarChamada() {
        System.out.println("Realizando chamada");

    }

}
```

Interface

- A utilização de interfaces em Java traz algumas vantagens para o desenvolvimento de software. Algumas dessas vantagens são:
  - Permite o polimorfismo de objetos de diferentes classes que implementam a mesma interface;
  - Favorece a modularização do código, pois as classes que implementam a mesma interface são agrupadas de forma lógica;
  - Promove a independência de implementação, pois uma classe pode implementar várias interfaces diferentes.