Instituto Federal de Educação, Ciência e Tecnologia do Ceará
Campus Maracanaú
Coordenadoria de Computação
Curso de Bacharelado em Ciência da Computação
Disciplina: Programação Orientada a Objetos
Professor: Igor Rafael Silva Valente

ATIVIDADE

Assunto:
Interfaces.

Orientações:
A atividade deve ser executada individualmente e entregue através do ambiente *Google Classroom*.

Regras de criação dos programas:
Crie um novo projeto Java denominado **AtividadeInterfaces**. As classes devem possuir os nomes informados no texto. Ao final, o projeto deve ser exportado para um arquivo em formato ZIP.

Nome completo:

Yan Pedro Façanha Brasileiro

1. Quais as diferenças entre classes abstratas e interfaces? Explique.
   Uma classe abstrata pode ter atributos, métodos abstratos ou concretos, e até construtor. Ela é usada quando há uma lógica comum que será compartilhada pelas subclasses. Só é possível estender uma classe abstrata por vez.

   Já as interfaces servem para definir contratos que uma classe deve cumprir, sem se preocupar tanto com a implementação. Até o Java 7, todas as interfaces só tinham métodos abstratos. A partir do Java 8, foi permitido criar métodos com implementação usando a palavra-chave `default`. Além disso, uma classe pode implementar várias interfaces, o que ajuda bastante quando queremos adicionar comportamentos diferentes a uma classe.

2. Interfaces podem ter métodos concretos? Explique.
   Sim, desde o Java 8, as interfaces passaram a aceitar métodos com corpo, chamados de *default methods*. Esses métodos são úteis porque permitem que a interface forneça uma implementação padrão, o que evita que todas as classes que a implementam precisem repetir o mesmo código. Também é possível ter métodos `static` com implementação dentro da interface. Isso fez as interfaces mais flexíveis, sem perder a ideia de que elas servem para definir um "contrato".

3. Demonstre como o uso de *default methods* pode evitar a repetição de código.

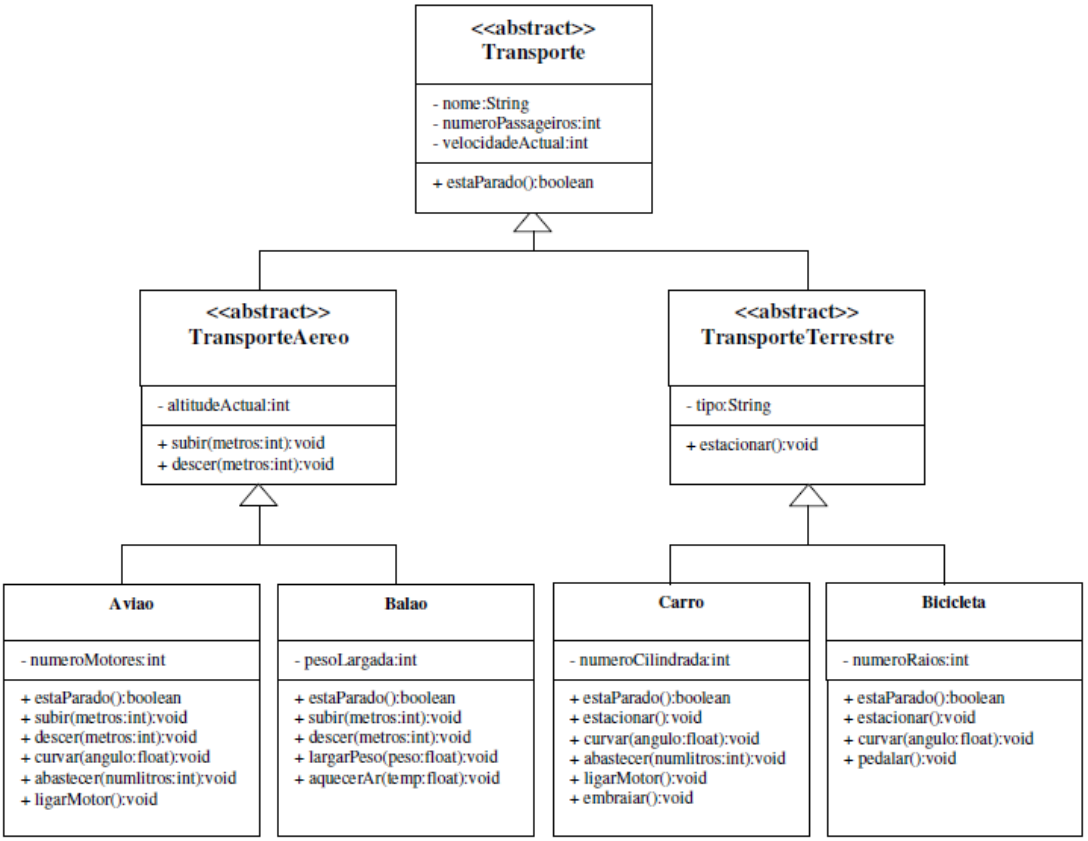
```
interface Logavel {
    default void log(String mensagem) {
        System.out.println("[LOG] " + mensagem);
    }
}

class Servico implements Logavel {
    public void executar() {
        log("Executando serviço...");
    }
}

class Repositorio implements Logavel {
    public void salvar() {
        log("Salvando dados...");
    }
```

```
}
```

Nesse caso, tanto a classe `Servico` quanto `Repositorio` conseguem usar o método `log()` da interface `Logavel` sem precisar reescrevê-lo. Isso economiza tempo, deixa o código mais limpo e evita erros de repetição.

4. Em uma situação em que classes abstratas e interfaces são opções viáveis, qual deve ser utilizada prioritariamente?
   Se for só para definir comportamentos, prefira a interface. Se for necessário compartilhar implementações e estrutura, aí é melhor usar uma classe abstrata.

5. Considere o diagrama UML a seguir e faça o que se pede:



O que se pede:

- Crie uma interface de nome Motorizado em que são declarados os métodos void ligarMotor() e void abastecer(int numLitros).
- Implemente a interface Motorizado nas classes Aviao e Carro.
- Escreva um programa de teste capaz de verificar a implementação anterior.
- Crie uma interface de nome Conduzivel onde é declarado o método void curvar(float angulo).
- Implemente a interface Conduzivel nas classes Aviao, Carro e Bicicleta.
- Complete o programa de teste criado anteriormente por forma a testar estas últimas implementações.

Boa sorte!

Prof. Igor.