Paradigmas da Programação
Teórico-Prática 3
Interfaces Java

Objetivos Específicos: Criação e implementação de interfaces Java. Especificação de métodos e/ou constantes numa interface Java. Declaração de variável do tipo de uma interface Java. Utilização das interfaces Java nativas, *Comparable* e *Comparator*. Ordenação de instâncias armazenadas em *Arrays*.

Exercício

1. Crie um novo projeto ***Netbeans***, chamado ***Tributaveis***.

2. Considere a seguinte classe ***Veiculo*** para representar veículos caracterizados pela matrícula, cilindrada e cor.

```
public class Veiculo {

    private String matricula;
    private int cilindrada;
    private String cor;

    public Veiculo(String matricula, int cilindrada, String cor) {
        this.matricula = matricula;
        this.cilindrada = cilindrada;
        this.cor = cor;
    }

    public String getMatricula() { return this.matricula; }
    public int getCilindrada() { return this.cilindrada; }
    public String getCor() { return this.cor; }

    public void setMatricula(String matricula){ this.matricula = matricula; }
    public void setCilindrada(int cilindrada){ this.cilindrada = cilindrada; }
    public void setCor(String cor){ this.cor = cor; }

    @Override
    public String toString() {
        return String.format("Veículo com matrícula %s e cilindrada %d "
                  + "tem cor %s", this.matricula, this.cilindrada, this.cor);}
}
```

3. Crie uma classe principal, chamada ***TesteTributaveis***, para permitir testar a nova classe.

4. Nesta classe de testes **crie e mostre** no ecrã uma instância da classe ***Veiculo*** com matrícula **22-33-CC**, **encarnado** e com cilindrada de **1000 cc**.

5. Crie a interface Java ***Cores***, para representar o seguinte conjunto de cores:

   { azul, cinzento, encarnado, verde }

6. Crie e mostre no ecrã uma nova instância da classe ***Veiculo*** com matrícula **44-55-DD**, **azul** e com cilindrada de **2500 cc**. Use a interface ***Cores*** para definir a cor do novo veículo.

7. Implemente a interface ***Cores*** na classe ***Veiculo***.

8. Crie uma nova instância da classe ***Veiculo*** com matrícula **11-22-BB**, **verde** e com cilindrada **1400 cc**. Especifique a cor do novo veículo através da classe ***Veiculo***.

9. Considere a seguinte classe ***Moradia*** para representar moradias caracterizadas pela morada, área e cor.

```
public class Moradia {

    private String morada;
    private float area;
    private String cor;

    public Moradia(String morada, float area, String cor) {
        this.morada = morada;
        this.area = area;
        this.cor = cor;
    }

    public String getMorada() { return this.morada; }
    public float getArea() { return this.area; }
    public String getCor() { return this.cor; }

    public void setMorada(String morada) { this.morada = morada; }
    public void setArea(float area) { this.area = area; }
    public void setCor(String cor) { this.cor = cor; }

    @Override
    public String toString() {
        return String.format("Moradia situada na %s com área de %.1f "
                + "tem cor %s", this.morada, this.area, this.cor);
    }
}
```

10. Implemente a interface ***Cores*** na classe ***Moradia***.

11. Crie e mostre no ecrã uma nova instância da classe ***Moradia*** com a morada **Rua do Bocage**, com cor **cinzenta** e com área de **90** metros quadrados.

12. Crie e teste uma **variável** capaz de guardar **qualquer referência** das instâncias existentes.

13. Altere as classes, ***Veiculo*** e ***Moradia***, de forma a facilitar, através do **polimorfismo**, o cálculo do **valor do imposto** aplicado às instâncias dessas classes. Considere as seguintes formas de calcular o valor do imposto de um:

    - **Veículo:** 15 € para cilindrada inferior a 1500 cc e 40€, caso contrário. Assumir que estes valores poderão ser alterados no futuro;
    - **Moradia:** igual ao dobro da área em metros quadrados.

14. Crie um contentor de objetos do tipo ***Array***, chamado tributaveis, para armazenar todas as instâncias existentes.

15. Guarde no contentor todas as instâncias criadas.

16. Mostre no ecrã o **total do imposto** aplicado às instâncias do contentor.

17. Programe uma listagem das instâncias armazenadas no contentor e por **ordem crescente** do valor do **imposto**, usando as seguintes interfaces nativas:

    a) *Comparable*

    b) *Comparator*

18. Programe uma nova listagem mas por **ordem decrescente** do valor do imposto, usando as interfaces nativas indicadas anteriormente.