Aula Prática 5

Objetivos

Utilização de Classes Abstratas e Interfaces.

Problema 5.1

Tome como base para este trabalho as classes desenvolvidas nas aulas anteriores relativamente às figuras geométricas (classes ColecaoFiguras, Ponto, Figura, Circulo, Quadrado e Rectângulo).

a) Aplicando os conceitos de Classes Abstractas e de Interfaces construa o código necessário para executar o seguinte programa:

```
public class Test {

   public static void main(String[] args)  {
        Circulo c1 = new Circulo(2);
        Circulo c2 = new Circulo(1, 3, 2);
        Quadrado q1 = new Quadrado(2);
        Quadrado q2 = new Quadrado(3, 4, 2);
        Rectangulo r1 = new Rectangulo(2, 3);
        Rectangulo r2 = new Rectangulo(3, 4, 2, 3);

        ColecaoFiguras col = new ColecaoFiguras(42.0);                  // MaxArea
        System.out.println(col.addFigura(c2));                          // true
        System.out.println(col.addFigura(r1));                          // true
        System.out.println(col.addFigura(r1));                          // false
        System.out.println(col.addFigura(r2));                          // true
        System.out.println(col.addFigura(c1));                          // true
        System.out.println(col.addFigura(q2));                          // true
        System.out.println(col.addFigura(q1));                          // false
        System.out.println(col.delFigura(r1));                          // true
        System.out.println(col.addFigura(q1));                          // true

        System.out.println("\nÁrea Total da Lista de Figuras: " + col.areaTotal());

        Figura[] listaFig = col.getFiguras();
        System.out.println("\nLista de Figuras:");
        for (Figura f: listaFig)
            System.out.println(f);

        System.out.println("\nComparacao da area do primeiro elemento com todos");
        for (int i = 0; i < listaFig.length; i++) {
            System.out.printf("%2d %12s de area %6.2f compareTo(listaFig[0]) = %2d\n", i,
                                    listaFig[i].getClass().getSimpleName(),
                                    listaFig[i].area(),
                                    listaFig[i].compareTo(listaFig[0]));
        }

        System.out.println("\nFigura com maior Area: " + UtilCompare.findMax( listaFig ) );

        // Ordena (crescente) o array de Figuras por areas
        UtilCompare.sortArray(listaFig);

        System.out.println ("\nLista de Figuras Ordenadas por Area:");
        for (Figura f: listaFig)
            System.out.println (f + " -> area: " + String.format("%2.2f", f.area()) +
                                  " e perimetro: " + String.format("%2.2f", f.perimetro()));
```

```
    }
}
```

Notas:

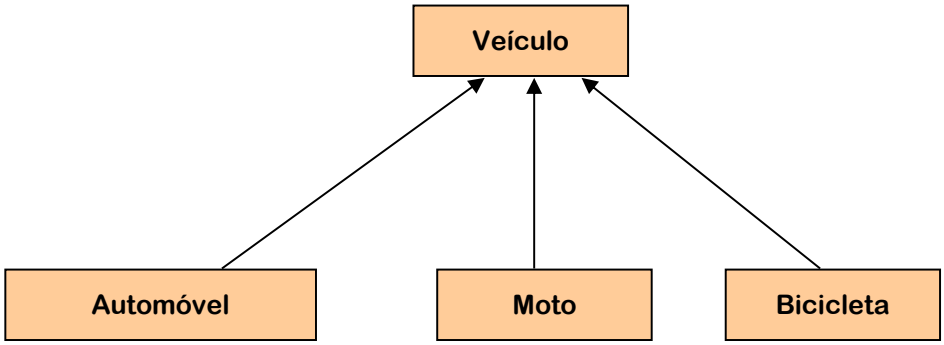
- Tenha em atenção que o método "int compareTo(T obj)" estaá definido numa Interface Comparable<T> de Java.
- Os utilitários "findMax" e "sortArray" devem ser colocados numa classe "UtilCompare.java"
- Garanta que estes métodos podem ser aplicados a qualquer conjunto de "Comparable"s.

Verifique se obteve o seguinte resultado:

```
true
true
false
true
true
true
false
true
true

Área Total da Lista de Figuras: 39.132741228718345

Lista de Figuras:
Circulo de Centro x: 1.0, y:3.0 e de raio 2.0
Rectangulo de Centro x: 3.0, y:4.0, altura 3.0, comprimento 2.0
Circulo de Centro x: 0.0, y:0.0 e de raio 2.0
Quadrado de Centro x: 3.0, y:4.0 e de lado 2.0
Quadrado de Centro x: 0.0, y:0.0 e de lado 2.0

Comparacao da area do primeiro elemento com todos
 0      Circulo de area  12.57 compareTo(listaFig[0]) =  0
 1   Rectangulo de area   6.00 compareTo(listaFig[0]) = -1
 2      Circulo de area  12.57 compareTo(listaFig[0]) =  0
 3     Quadrado de area   4.00 compareTo(listaFig[0]) = -1
 4     Quadrado de area   4.00 compareTo(listaFig[0]) = -1

Figura com maior Area: Circulo de Centro x: 1.0, y:3.0 e de raio 2.0

Lista de Figuras Ordenadas por Area:
Quadrado de Centro x: 3.0, y:4.0 e de lado 2.0 -> area: 4.00 e perimetro: 8.00
Quadrado de Centro x: 0.0, y:0.0 e de lado 2.0 -> area: 4.00 e perimetro: 8.00
Rectangulo de Centro x: 3.0, y:4.0, altura 3.0, comprimento 2.0 -> area: 6.00 e
perimetro: 10.00
Circulo de Centro x: 1.0, y:3.0 e de raio 2.0 -> area: 12.57 e perimetro: 12.57
Circulo de Centro x: 0.0, y:0.0 e de raio 2.0 -> area: 12.57 e perimetro: 12.57
```

Problema 5.2

Considere o seguinte cenário de herança:



a) Implemente a hierarquia de classe acima descrita de forma a utilizar o conceito de generalização abordado nas aulas TP. Tenha em atenção as características e comportamento dos objectos apresentados como, por exemplo, o ano, a matrícula, a cor base, o nº de rodas, a cilindrada, a velocidade máxima, etc.

b) Para além da hierarquia acima descrita, pretendemos ter alguns veículos (por exemplo BicicletaPolicia, MotoPolicia, CarroPolicia) que possam ter os seguintes comportamentos:
    a. Policia
        i. getTipo()
           valores possíveis de retorno: INEM, Bombeiros, GNR, PSP, PJ
        ii. String getID() // identificador interno da viatura

    b. Motorizado
        i. getPotencia();
        ii. getConsumo();
        iii. getCombustivel();

   Note que para obter uma solução poderá ter de modificar a hierarquia de classes ou acrescentar novas classes e/ou interfaces.

c) Desenvolva um programa *main* para demonstrar todas as funcionalidades implementadas em a) e b). Crie também uma lista de Veículos e ordene-os usando a função *UtilCompare.sortArray* desenvolvida no problema anterior (considere o ano como atributo de comparação).

Problema 5.3

Utilizando o programa desenvolvido nas aulas anteriores relativamente à Agenda de Contactos (Pessoas), acrescente-lhe a capacidade de guardar e carregar uma agenda de um ficheiro de texto.

Por uma questão de compatibilidade com outras soluções, a aplicação deverá suportar 3 tipos diferentes de formatos (Nokia, vCard e CSV). A primeira linha do ficheiro identifica o respetivo formato. De acordo com o formato, os registos estão armazenados de forma diferenciada:

- Nokia: Cada linha contêm um dos elementos de informação (i.e. um campo). Existe uma linha em branco a separar cada registo;
- vCard: Cada linha contém um registo e os campos estão delimitados pelo caracter #;
- CSV: Cada linha contém um registo e os campos estão delimitados pelo caracter TAB (\t);

Sugestão: Defina uma interface normalizada com as operações (abstactas) necessárias à leitura e escrita de contactos. Implemente classes específicas que permitam operar com diferentes formatos de ficheiros de contactos.

Exemplos de ficheiros de contactos com diferentes formatos:

File1.txt

```
Nokia
Maria João Sousa
9123454
12/05/1989

Rui Manuel Costa
985423
20/09/19

Joaquim Ferreira
9312452
18/04/1964
...
```

File2.txt

```
vCard
#Maria João Sousa#9123454#12/05/1989
#Rui Manuel Costa#985423#20/09/19
#Joaquim Ferreira#9312452#18/04/1964
...
```

File3.txt

```
CSV
Maria João Sousa      9123454      12/05/1989
Rui Manuel Costa      9854233      20/09/19
Joaquim Ferreira      9312452      18/04/1964
...
```