Aula prática nº 6

Tópicos

- Polimorfismo
- Classes abstratas

Exercícios

1. Continue a trabalhar no ex. 2 da aula 5. Às classes existentes Pessoa e Aluno adicione tipos seguintes:
   - classe Professor derivada da classe Pessoa. A classe Professor deve ter um atributo adicional que é área principal de investigação. Redefina o método toString() para que este devolva para além do nome e apelido, a área de investigação do professor. Não reescreva o método toString() completamente, aproveite o método existente na classe base.
   - classe AlunoPosGrad derivada da classe Aluno. A classe AlunoPosGrad deve ter um atributo adicional – referência para o orientador. Redefina o método toString() para que este devolva para além do nome, apelido e número mecanográfico do aluno, o nome, apelido e a área de investigação do orientador.
   - classe Disciplina que contém informação sobre o nome da disciplina, o professor responsável e alunos inscritos. Imagine que numa disciplina podem inscrever-se os alunos de mestrado integrado e os de pós-graduação e que o número de alunos é ilimitado. Inicialmente não se sabe quantos alunos vão inscrever. Assegure que o vetor de referências para alunos cresça dinamicamente.

   Teste as classes desenvolvidas com uma função main que permita criar alunos e disciplinas, inscrever alunos em disciplinas e imprimir toda a informação sobre disciplinas, incluindo o seu nome, informação sobre o professor responsável, número de alunos inscritos e os dados de todos os alunos.

2. Utilizando os programas desenvolvido nas aulas anteriores relativamente às figuras geométricas (classes Ponto, Figura, Circulo, Quadrado e Rectângulo), crie uma nova classe `ColecaoFiguras` que suporta um conjunto (coleção) de Figuras não repetidas. Implemente os seguintes métodos:

```
public ColecaoFiguras(double maxArea)  // O construtor define a área
máxima da coleção de figuras
public boolean addFigura(Figura f)     // Adiciona uma figura à coleção
public boolean delFigura(Figura f)     // Remove uma figura da coleção
public double areaTotal()              // Retorna a área total das figs.
public boolean exists(Figura f)        // Verifica se uma figura existe
```

```
public String toString()   // Retorna as características da coleção
public Figura[] getFiguras() // Retorna uma lista com todas as figs.
public Ponto[] getCentros()  // Retorna uma lista com todos os centros
```

Teste as classes desenvolvidas com 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));
        System.out.println(col.addFigura(r1));
        System.out.println(col.addFigura(r1));
        System.out.println(col.addFigura(r2));
        System.out.println(col.addFigura(c1));
        System.out.println(col.addFigura(q2));
        System.out.println(col.addFigura(q1));
        System.out.println(col.delFigura(r1));
        System.out.println(col.addFigura(q1));
        System.out.println("\nÁrea Total: " + col.areaTotal());
        System.out.println("\nLista de Figuras:");
        for (Figura f: col.getFiguras())
           System.out.println(f);

        System.out.println("\n\nCirculos na Lista de Figuras:");
        for (Figura f: col.getFiguras())
            if (f instanceof Circulo)
               System.out.println(f);

        System.out.println("\n\nCentro das Figuras:");
        for (Ponto p: col.getCentros())
           System.out.println(p);
   }
}
```

Verifique se obteve o seguinte resultado:

```
true
true
false
true
true
true
false
true
true
Área Total: 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
Circulos na Lista de Figuras:
Circulo de Centro x: 1.0, y:3.0 e de raio 2.0
Circulo de Centro x: 0.0, y:0.0 e de raio 2.0
Centro das Figuras:
x: 1.0, y:3.0
x: 3.0, y:4.0
x: 0.0, y:0.0
x: 3.0, y:4.0
x: 0.0, y:0.0
```

3. Uma empresa transportadora aérea precisa de um programa de registo dos passageiros para cada um dos seus voos. Projete e construa esse programa, com as seguintes especificações:

   Cada passageiro regista nome e apelido, data de nascimento, sexo e o número de viagens realizadas nesse ano civil e permite o cálculo do número de milhas que ganhará com a viagem (essas milhas podem posteriormente ser convertidas em viagens grátis).

Implemente os seguintes tipos de passageiros:
- passageiros que viajam em classe turística (ganham o número de milhas da viagem).
- passageiros que viajam em classe executiva (ganham duas vezes o número de milhas do voo).
- passageiros que viajam em primeira classe. Este tipo de passageiro ganha o seguinte número de milhas por viagem (sendo *n* o número de viagens feitas nesse ano, e *m* a distância da viagem):

$$Milhas = \begin{cases} 2 \times m, & se \;\; n \leq 5 \\ (2 + 0.2 \times n) \times m, & se \;\; n > 5 \end{cases}$$

Desenvolva o código de modo a que seja facilmente extensível (por exemplo, para poder adicionar novos tipos de passageiros) e modificável (por exemplo, para alterar o modo de cálculo de milhas).

Exemplifique o uso das classes desenvolvidas para o caso dum programa simples que deve simular, de uma forma interativa com o utilizador, a gestão de um voo. Inicialmente, deve ler o número de milhas do voo. Após essa leitura deve
- permitir o registo de novos passageiros;
- apresentar a lista de todos os passageiros inscritos até ao momento com informação sobre cada um deles (nome, apelido, data de nascimento, sexo, classe de viagem, número de milhas que ganhará com este voo);
- apresentar o total de milhas oferecidas nesse voo.