

## ESQUEMA DA AULA PRÁTICA 8

### □ Herança

1 – Implemente a classe abaixo:

```
public class Base {
    private int a;
    private int b;

    public Base(int a, int b) {
        this.a=a;  this.b=b;
    }
    public int soma(){
        return a+b;
    }
    public int soma(int x){
        return a+b+x;
    }
    public int getA(){...}
    public int getB(){...}
    public void setA(int a){...}
    public void setB(int b){...}
    public String toString {...}
    public boolean equals (Object o ) { ...}
    public Object clone () {...}
}
```

- a) Para a classe Base construa os métodos toString, equals e clone.
- b) Construa uma classe teste para testar a classe Base.
- c) Defina uma classe, Derivada, como subclasse da classe Base. Na subclasse Derivada defina um atributo *a* do tipo *int*, um atributo *c* do tipo *int*, e um construtor com a assinatura Derivada (Base b, int a, int c).
- d) Na subclasse Derivada construa ainda os seguintes métodos:
  - Método toString
  - Método que calcule a soma de todas as variáveis de instância de um objeto da classe Derivada
  - Método equals
  - Método clone

e) Construa uma classe teste para testar a classe Derivada

**2** - Implemente uma classe *Disciplina* com os atributos *codigo* (valor do tipo *int*) *designacao* (valor do tipo *String*) e *nota* (*int*) e com os seguintes métodos:

- Construtor com os atributos *codigo* e *designacao* como parâmetros;
- Getters e setters para cada atributo;
- Método *equals*;
- Método *toString*;
- Método *clone*.

- Implemente ainda uma classe *Aluno* tal que cada objecto do tipo *Aluno* tenha um número e um nome de aluno. Defina um construtor com o número e nome de aluno como parâmetros assim como os getters e setters para cada atributo.

a) – A partir da classe *Aluno* pretendem definir-se duas subclasses, *AlunoLicenciatura* e *AlunoPosGraduacao*. A classe *AlunoLicenciatura* terá como atributos o curso frequentado (*String*) e um vector disciplinas com objectos do tipo *Disciplina*. Este vector (objecto do tipo *Vector*) irá conter as disciplinas feitas pelo aluno.

Defina a classe *AlunoLicenciatura* com um construtor que tem como parâmetros um objecto do tipo *Aluno* e o curso e com os seguintes métodos:

- *setDisciplina* que dado um objecto do tipo *Disciplina* deverá adicioná-lo ao vector *disciplinas* caso este não exista ainda no vector.

- *getNotaDisciplina* que dado um código de disciplina deve devolver a nota da disciplina. Se essa disciplina não constar do vector de disciplinas do aluno de licenciatura o método deve devolver o valor 0.

- método que calcule a média das classificações obtidas pelo aluno de licenciatura.

Numa classe de teste:

b) Construa um método público e estático que dado um array de objectos do tipo *Aluno* conte quantos desses alunos são alunos de licenciatura.

c) Teste as operações:

- `getNotaDisciplina` da classe `AlunoLicenciatura` ;
- o método que calcula a média das classificações do aluno de licenciatura;
- o método que, dado um array de objetos do tipo aluno, conta quantos desses alunos são alunos de licenciatura.

d) Indique qual o output do programa anterior.

### 3 - Desenvolva um sistema de Gestão de Contas Bancárias<sup>1</sup>.

O mesmo deverá ser capaz de realizar diversas operações bancárias sobre contas, tendo em atenção que existem 3 tipos de conta diferentes: normal, vencimento e a prazo.

Todas as contas têm um número, um ou mais titulares, uma data de criação e um saldo.

As contas normais não permitem que o saldo seja inferior a zero.

As contas vencimento permitem que o saldo seja negativo (até determinado limite que pode ser alterado) e mantêm o número de identificação bancária (NIB) da entidade empregadora.

A conta a prazo rende juros, dependendo do tempo que cada depósito permaneça na conta.

Sobre todas as contas pretende-se ter disponíveis as seguintes operações:

- a. Criação de conta
- b. Levantamento de uma quantia
- c. Depósito de uma quantia
- d. Consulta do saldo atual
- e. Consulta da data, montante e descrição dos últimos 5 movimentos (depósitos e levantamentos)

Identifique os objetos relevantes para o seu sistema e implemente as classes necessárias servindo-se do mecanismo de herança sempre que apropriado.

4 - Teste as classes anteriores.

---

<sup>1</sup> Exercício adaptado de "Programação Orientada aos Objectos em Java 2", F. Mário Martins, FCA, 2000.