

## ESQUEMA AULA PRÁTICA 11

### □ Ficheiros

1 - Implemente o exemplo abaixo, classe Livro, classe GerirLivros e classe FuncLivros, e estude o código:

**Classe Livro. Os objetos do tipo livro irão ser escritos num ficheiro.**

```
import java.io.Serializable;
```

```
public class Livro implements Serializable{
    private int id;
    private String titulo;
    private String autor;

    public Livro(int id, String titulo, String autor) {
        this.id = id;
        this.titulo = titulo;
        this.autor = autor;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getTitulo() {
        return titulo;
    }
    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }
    public String getAutor() {
        return autor;
    }
    public void setAutor(String autor) {
        this.autor = autor;
    }
    @Override
    public String toString() {
        return "Livro{" + "id=" + id + ", titulo=" + titulo + ", autor=" + autor + '}';
    }
    public boolean equals (Object obj){
        if (obj!= null && this.getClass()== obj.getClass()){
            Livro li = (Livro)obj;
            return this.id == li.id && this.autor.equals(li.autor) &&
                this.titulo.equals(li.titulo);
        }
        return false;
    }
    public Object clone(){
        Livro copia = new Livro (this.id, this.titulo, this.autor);
        return copia;
    } }
}
```

**Classe com o main:**

```
import java.io.*;
import java.util.ArrayList;
import myinputs.Ler;

public class GerirLivros {

    public static int menu() { // função na classe do main
        int opcao;
        System.out.println("1 - Novo livro");
        System.out.println("2 - Listar livros");
        System.out.println("3 - Apagar livro");
        System.out.println("4 - Consultar livro dado título");
        System.out.println("5 - Alterar título livro");
        System.out.println("6 - Qual o autor com mais livros");
        System.out.println("7 - Sair");
        System.out.println("Qual a sua opção:");
        opcao = Ler.umInt();
        return opcao;
    }

    public static void main(String[] args) {
        int escolha;
        // Lista que vai conter todos os livros;
        ArrayList<Livro> meusLivros = new ArrayList<Livro>();
        // Ler ficheiro
        try {
            ObjectInputStream is = new ObjectInputStream( new FileInputStream("d:\\teste\\livros.dat"));
            meusLivros = (ArrayList<Livro>) is.readObject();
        }
        catch (IOException e){
            System.out.println(e.getMessage());
        }
        catch ( ClassNotFoundException e) {
            System.out.println(e.getMessage());
        }
        }

        do{
            escolha = menu();
            switch (escolha){
                case 1: FuncLivros.inserirLivro(meusLivros);
                    break;
                case 2: System.out.println(meusLivros);
                    break;
                case 3: // a implementar
                case 4: // a implementar
                case 5: // a implementar
                case 6: // a implementar
            }
        } while(escolha != 7);
    }
}
```

Classe com funções para gerir o conjunto dos livros:

```
import java.io.*;
import java.util.ArrayList;
import myinputs.Ler;

public class FuncLivros {

    public static void inserirLivro (ArrayList<Livro> livros){
        System.out.println("Qual o número do Livro? ");
        int num = Ler.umInt();

        // validar número de livro
        for (int i = 0; i < livros.size(); i++) {
            if (livros.get(i).getId()== num){
                System.out.println("Já existe um livro com esse número");
                return;
            }
        }

        //obter dados do livro; instanciar o objeto Livro;
        System.out.println("Qual o nome do Livro? " );
        String nome = Ler.umaString();
        System.out.println("Qual o autor do Livro? " );
        String autor = Ler.umaString();
        Livro l = new Livro (num, nome, autor);

        // adicionar o novo livro à lista
        livros.add(l);

        // atualizar ficheiro
        try {
            ObjectOutputStream os = new ObjectOutputStream(new FileOutputStream("d:\\teste\\livros.dat"));
            // escrever o objeto livros no ficheiro
            os.writeObject(livros);
            os.flush(); // os dados são copiados de memória para o disco
        } catch (IOException e) {
            System.out.println(e.getMessage());
        }
    }
}
```

2 - Implemente as opções 3, 4, 5, e 6.

3 – Implemente a classe Aluno listada abaixo.

```
import java.io.*;

public class Aluno implements Serializable {
// Para poder ser escrita num ficheiro a classe tem que implementar a interface Serializable

private static int ultimo = 0;
private int numero;
private String nome;

public Aluno (String n){
    ultimo++;
    numero = ultimo;
    nome = n;
}
public static int getUltimo() { return ultimo; }
public static void setUltimo(int ultimo) { Aluno.ultimo = ultimo; }
public void setNome (String n ){ nome = n; }
public String getNome (){ return nome; }
public int getNumero (){ return numero; }

public String toString (){
    return "Aluno nº: " + numero + "\t Nome: " + nome ;
}
}
```

a) Construa uma pequena aplicação que permita as seguintes opções:

- 1 - Criar aluno
- 2 – Consultar aluno, dado o seu número
- 3 – Consultar aluno, dado o seu nome
- 4 – Listar todos os alunos
- 5 – Apagar um aluno
- 6 – Corrigir o nome de um aluno
- 7 – Terminar

- Os alunos criados deverão ser armazenados num objeto alunos, do tipo `java.util.ArrayList<Aluno>`. Sempre que a lista de alunos for modificada, deverá ser escrita para um ficheiro (`fichAlunos`). Quando o programa é executado deverá começar por ler o conteúdo desse ficheiro.

b) Analise a aplicação e tente perceber o que se passa com a variável estática quando escreve a lista de alunos para o ficheiro.

c) - No programa que escreve para o ficheiro, substitua a instrução,  
`os.writeObject( alunos);`

por:

```
os.writeInt(Aluno.getUltimo());  
os.writeObject(alunos);
```

No programa que lê o ficheiro, substitua a instrução  
`ArrayList lista = (ArrayList) is.readObject();`

por:

```
int ult= is.readInt();  
Aluno.setUltimo(ult);  
ArrayList lista = (ArrayList) is.readObject();
```

- O que mudou?

### Exercícios de revisão da matéria:

4 - O programa abaixo usa a classe `JOptionPane` para pedir e mostrar dados ao utilizador.

a) Implemente o programa e estude a classe `JOptionPane`.

```
import javax.swing.*;  
public class TesteOption{  
    public static void main (String[]args){  
  
        JOptionPane.showMessageDialog(null,"Ora viva");  
        String valor = JOptionPane.showInputDialog(null, "Introduza um inteiro:");  
        int i = Integer.parseInt(valor);  
        JOptionPane.showMessageDialog(null, " O valor dado foi "+i);  
        System.exit(0);  
    }  
}
```

b) O que acontece se, quando a caixa de diálogo lhe pede um valor, você pressionar o botão "Cancel"? Altere o programa de forma a não terminar anormalmente.

c) Modifique o programa para ler um valor do tipo `double`.

5 - Simplificando, podemos afirmar que, uma pessoa é alguém de quem sabemos o nome, o sexo e a nacionalidade. Programe a classe `Pessoa` com os construtores (entre eles implemente o construtor não parametrizado – qual a utilidade? – e o *construtor por cópia*), seletores e modificadores que considerar necessários bem como os métodos públicos `toString`, `clone` e `equals` (ainda que os considere pouco naturais para uma pessoa terão interesse do ponto de vista da programação!).

**6** - Programe a classe `Autor` de obra literária. Um autor é uma pessoa sobre a qual será interessante saber o ano de nascimento e falecimento (caso já tenha falecido), a sua nacionalidade bem como o prémio literário mais importante que conquistou, se detivermos esta informação.

**7** - Programe a classe `Musico`. Um músico pode ser um indivíduo ou uma banda de que interessa manter uma descrição e o seu estilo musical dominante.

**8** - Programe a classe `Amigo`. Um amigo é uma pessoa de quem sabemos a data de nascimento, o ano em que o conhecemos, um contacto, o nível de amizade que por ela nutrimos e ainda o “parceiro” com quem normalmente “anda”.

**9** - Pretende-se organizar a coleção de livros e CDs áudio das nossas biblioteca e discoteca particulares<sup>1</sup>.

Um livro é caracterizado pelo título, pelo autor, pela língua em que está escrito, pelo seu valor afetivo e se já o lemos pela indicação da data em que concluímos a sua leitura. Por vezes emprestamos livros a amigos e é necessário saber se o livro está emprestado e a quem.

A nossa coleção de CDs de música tem um tratamento semelhante à dos livros: podemos emprestá-los, têm determinado valor afetivo e poderemos querer saber quando os ouvimos pela última vez. Existem analogias entre as duas classes e também diferenças (por exemplo, o CD de música não tem autor mas sim músico/banda e não será necessário indicar a língua mas sim o estilo de música). Agrupe as semelhanças de comportamento e estrutura numa classe chamada `Bibliónimo`<sup>2</sup>. O objetivo desta classe é simplificar a subsequente programação das classes `Livro` e `CDAudio`.

**10** - Programe uma classe (`FilmeDVD`) para manter e manipular informação sobre os filmes de uma filmoteca em suporte DVD.

---

<sup>1</sup> Exercício adaptado de "Programação com classes em C++", P. Guerreiro, FCA, 2000.

<sup>2</sup> Bibliónimo significa nome de qualquer livro de reputação universal (do grego *bíblion* (livro) + *ónyma* (nome)). Parece ser por isso um nome apropriado (até que sugiram um melhor) para uma classe que procura disponibilizar o comportamento comum dos nossos objetos culturais de consumo.