→ Streams de bytes

As subclasses de OutputStream e InputStream implementam Streams de bytes. Os métodos abstractos definidos na classe OutputStreams são idênticos aos de Writer com a diferença de que em vez de caracteres aceitam bytes.

→ As classes DataOutputStream e DataInputStream

Construtores: DataOutputStream(OutputStream out);
DataInputStream(InputStream in)

1 – Teste o código abaixo e seguidamente construa uma classe que leia o ficheiro teste6.dat

```
public static void main (String args[]){
    DataOutputStream os;

    try {
     os = new DataOutputStream ( new FileOutputStream("d:\\My_work\\teste6.dat"));
     os.writeDouble(2.335);
     os.writeInt(33);
     os.writeUTF("XPTO");  //Unicode Text Format
     os.flush();
     os.close();
    }
    catch (IOException e){
      System.out.println(e.getMessage());
    }
}
```

2 – Construa um programa que gere dois vectores de valores reais aleatórios e que guarde esses vectores em ficheiros. Guarde também no ficheiro a dimensão de cada vector.

3 – Construa um programa que leia os ficheiros anteriores.

4 – Construa uma só aplicação que permita:
1 – Gerar vectores
2 – Gravar vector num ficheiro
3 – Ler vector de um ficheiro
4, 5, 6 ... – Realizar operações com vectores (adição, produto por um escalar, produto interno, ...

→ A interface java.io.Serializable

A interface Serializable não define qualquer método sendo apenas a especificação da propriedade de que as instâncias de uma classe que implemente essa interface poderão ser gravadas em memória secundária sob a forma de objectos.

5 – Implemente e teste a classe abaixo

```
public class C7 implements Serializable{
  private int n;
  private String nome;
  public C7(int n, String nome){
    this.n=n;
    this.nome=nome;
  }
  public void setNumero (int i){
    n=i;
  }
  public void setNome (String n){
    nome=n;
  }
  public int getNumero(){
    return n;
  }
  public String getNome(){
    return nome;
  }
  public String toString (){
    return (n+" "+nome);
  }
}
```

→ As classes ObjectInputStream e ObjectOutputStream

Construtores: ObjectInputStream(InputStream in)
ObjectOutputStream(OutputStream out)

Uma ObjectOutputStream permite armazenar objectos através do método writeObject() que implementa um algoritmo de serialização que garante que todas as referências cruzadas existentes entre instâncias de diferentes classes serão repostas aquando do processo de leitura dessas mesmas instâncias.

Para que se possam gravar instâncias de uma determinada classe numa ObjectOutputStream é necessário que a classe implemente a interface *Serializable*. Além disso, todas as variáveis dessa classe terão que ser também serializáveis. Isto significa que todas as variáveis de instância da classe devem por sua vez pertencer a classes serializáveis. Os tipos simples são por definição serializáveis, assim como o são os arrays e as instâncias das classes String e Vector.

6 – Implemente e teste a classe abaixo.

```
public class teste {
  public static void main (String args[]){
     int i;
     C7[] v= new C7[100];
     for (i=0; i<100; i++){
       v[i] = new C7( i, "XPTO da Silva");
     }
     try {
       ObjectOutputStream os = new ObjectOutputStream
         ( new FileOutputStream ("d:\\testeoo.dat"));
       for (i=0; i<100; i++){
          os.writeObject( v[i]);

       }
       os.flush();
      }
      catch (IOException e){
       System.out.println(e.getMessage());
     }//catch  }//main  } //class teste
```

7 – Construa uma classe que lhe permita ler o ficheiro "testeoo.dat" criado anteriormente.

8 – Defina uma classe aluno, em que cada aluno tem um número, um nome e pertence a um dado curso (apenas a designação).

9 – Construa agora uma aplicação que crie uma turma de alunos. A cada novo aluno inserido na turma deverá ser atribuído um número sequencial que identifique a sua ordem de inscrição na turma. Quando o utilizador sair da aplicação, os alunos da turma deverão ser guardados num ficheiro, de tal forma que quando se iniciar novamente a aplicação os alunos do ficheiro sejam lidos para o programa. Defina ainda as operações que considerar úteis para o funcionamento da sua aplicação.