Aula Prática 8

Resumo:

- Listas e vectores ordenados.
- Recursão e iteração.

Exercício 8.1

A classe `SortedListInt`, desenvolvida na aula teórica, encontra-se em anexo. Usando esta classe, faça um programa que leia números inteiros de um ou mais ficheiros e, no final, imprima todos os números lidos por ordem. Os nomes de ficheiros serão passados como argumentos ao programa. O ficheiro poderá conter outras palavras: tem de extrair apenas as que representam inteiros!

Exercício 8.2

Faça as necessárias adaptações de forma a transformar a classe `SortedListInt` numa classe genérica `SortedList` que deverá colocar no pacote `p2utils`.[1]

Exercício 8.3

Acrescente à nova classe `SortedList` o método `contains(e)`, que verifica se um dado elemento `e` existe na lista. Este método já tinha sido implementado na aula anterior para a classe `LinkedList`. No entanto, no caso da `SortedList`, como a lista está ordenada, pode desenvolver uma implementação mais eficiente. Comece por desenvolver e testar uma solução iterativa. Em seguida, comente essa solução e desenvolva e teste uma solução recursiva.

Exercício 8.4

Acrescente ainda à classe `SortedList` os seguintes métodos:

- `toString()` - Devolve uma cadeia de caracteres que representa o conteúdo da lista, por exemplo `[1,12,7,9]`. Desenvolva uma solução iterativa para este método.

[1] Na declaração da classe, deve especificar que os elementos do tipo `E` são comparáveis: `public class SortedList<E extends Comparable<E>> {...}`. Na comparação de elementos terá que usar a função `compareTo()`

- `merge(lst)` - Devolve uma nova lista ordenada contendo os elementos da lista (em que o método é chamado) e os elementos da lista dada no argumento. Desenvolva uma solução recursiva para este método.

Exercício 8.5

Acrescente uma nova classe `SortedArray` ao pacote `p2utils`. Esta classe tem uma funcionalidade em grande parte semelhante à da `SortedList`, no entanto deverá ser implementada com base num vector de dimensão fixa. A dimensão será dada como argumento do construtor. Como o vector tem capacidade limitada, deverá ter um método `isFull()`, que devolve `true` se o vector estiver cheio, e `false` caso contrário.

Exercício 8.6

A classe `sortedList` genérica permite, por exemplo, manter uma lista ordenada dos aniversários dos seus familiares e amigos. Para isso:

- Desenvolva uma classe `Pessoa` com as seguintes funcionalidades: construtor tendo como parâmetros o dia de nascimento (uma `Data`) e o nome da pessoa (`String`); métodos de acesso aos atributos data de nascimento e nome; método toString(), que devolve uma representação da pessoa em cadeia de caracteres.

- Adicione a essa classe a implementação do método `compareTo(Pessoa p)` tendo em conta o seguinte: o método devolve um inteiro; a comparação deve apenas ser feita com base no dia e mês de nascimento; se a pessoa em que se está a invocar a função `compareTo()` fizer anos antes de `p`, o valor a devolver deve ser `-1`; se ambas tiverem a mesma data de aniversário, deve devolver `0`; nos outros casos deve devolver +1.

  Como este método (`compareTo()`) é pre-definido em Java, terá que explicitar que o seu método é uma implementação desse método pré-definido: `public class Pessoa implements Comparable<Pessoa> ...` .

- Crie um programa que crie uma lista ordenada (`SortedList`) de pessoas, adicione cinco pessoas a essa lista e em as apresente por ordem de data de nascimento.