## **Estruturas de Dados**

## Ficha Laboratorial N.º 5

Engenharia Informática – Instituto Superior de Engenharia de Coimbra

- 1 Collections.fill recebe uma lista (List) e um valor, e coloca esse valor em todas as posições da lista. Implemente um método equivalente.
- 2 Construa um método que imprime o conteúdo de uma lista (List) por ordem inversa usando um ListIterator de forma adequada.
- 3 Pretende-se criar uma classe que implementa uma pilha de valores genéricos, através dos métodos empty(), peek(), pop() e push(valor). O método empty() indica se a pilha está vazia (ou seja, devolve verdadeiro se estiver vazia), o método peek() devolve o valor no topo da pilha, o método pop() devolve e também remove da pilha o valor que está no seu topo, enquanto que push(valor) acrescenta um novo valor ao topo da pilha. Os objetos são internamente guardados através de uma lista (List) que é recebida como pârametro do construtor da pilha. Implemente esta classe e:
  - a) Analise o desempenho da sua implementação quando é utilizado um ArrayList para guardar os valores da pilha. Teste, por exemplo, adicionando alguns milhares de elementos com push() e removendo-os de seguida com pop().
  - b) Analise o desempenho da sua implementação quando é utilizado uma LinkedList para guardar os valores da pilha.
  - c) Explique qualquer discrepância ou resultado que considere inesperado, identificando a ordem de complexidade dos métodos que implementou, conforme seja usada uma LinkedList ou ArrayList.
  - d) Tendo em conta os resultados anteriores, identifique qual o tipo de lista que deve ser utilizada para implementar uma *Pilha*.
- 4 Repita a alínea anterior, implementando desta vez uma fila. Esta deve suportar os métodos add(e), remove() e element(). O método add(e) acrescenta ao fim da fila, enquanto que os dois últimos métodos dizem respeito ao elemento à cabeça da lista o método remove() remove e devolve esse elemento, enquanto que element() limita-se a devolvê-lo.
- 5 Collections.reverse recebe uma lista (List) e troca a ordem dos seus elementos. Implemente um método equivalente (obviamente, sem recorrer ao método Collections.reverse), garantindo bom desempenho (em termos de complexidade) independentemente do tipo de lista que é recebida.

(Há várias formas de resolver este problema corretamente — Note que ao contrário do exercício 2, aqui é necessário que os elementos da lista seja efetivamente trocados, e que o método não devolve uma nova lista, mas faz alterações à lista recebida. Pode resolver percorrendo a lista do início e do fim ao mesmo tempo, realizando trocas através de uma variável auxiliar.)

- 6 Construa um método que usa o API de colecções para imprimir o conteúdo de uma colecção (*Collection*) por ordem inversa (daquela que é percorrida pelo seu iterador). Não deve usar o Collections.reverse(). (Ao contrário do exercício anterior, neste caso não é possível assumir que o objeto de entrada é uma List, e é necessário apenas imprimir a informação pela ordem indicada. Pode resolver recorrendo por exemplo a uma estrutura de dados auxiliar.)
- 7 Construa um método que remove todos os elementos em posições pares de uma lista de entrada (List). Sabendo que essa lista tanto pode ser do tipo ArrayList ou do tipo LinkedList e assumindo que a alocação de memória é feita em tempo constante:
  - a) Pretende-se que o tempo de execução seja O(N), podendo ser usadas estruturas de dados auxiliares para o efeito. Desenvolva o método correspondente.
  - b) Pretende-se que o espaço adicional utilizado pelo método seja de complexidade espacial O(1). Desenvolva o método correspondente.
- 8 Pretende-se criar uma estrutura que suporte as operações push(), pop() e findMin() em tempo constante (O(1)). Implemente esta estrutura. (Sugestão: Utilize duas pilhas.)