Estruturas de Dados

Teste Laboratorial 2 – *10 de Dezembro de 2018*

Nome:_____

Número:_____

Indicações gerais.

- **Responda na folha da prova. Utilize os espaços disponibilizados e o verso da folha se necessitar de mais espaço.**
- **Leia as perguntas com atenção. As soluções pedidas têm frequentemente que obedecer a determinadas restrições. Caso essas restrições não sejam cumpridas, a resposta não é valorizada.**
- **Sempre que declarar referências para tipos genéricos, deve indicar explicitamente o tipo dos respetivos parametros formais.**

1- Pretende-se construir uma estrutura de dados que armazena informação sobre cidades. Esta estrutura deve suportar, entre outras, as seguintes operações, com a complexidade indicada:

a) **`defineCidade(String nome1, int população)`**: indica que existe uma cidade com o nome e populaçã indicados (complexidade O(1))
b) **`migração(String origem, String destino, int quantos):`** indica que migram "quantos" pessoas da cidade "origem" para a cidade "destino" (complexidade O(1)).
c) **`getPopulacao(String cidade):`** devolve o número de habitantes da cidade com o nome indicado (complexidade O(1)).

2- Pretende-se usar um priority queue para construir uma estrutura de dados que é usada para decidir qual das encomendas em espera será atendida primeiro. A política a seguir é remeter primeiro as encomendas urgentes, desempatando por data de pedido - os pedidos mais antigos devem ser atendidos primeiro *(a classe Date é comparável, isto é, podem ser comparadas duas datas d1 e d2 através de d1.compareTo(d2))*.

```
class Encomenda{
boolean urgente;
Date dataDePedido;
};
```

Estruturas de Dados

Teste Laboratorial 2 – *10 de Dezembro de 2018*

Nome:_____

Número:_____

3- Considere a seguinte implementação de uma árvore binária de pesquisa

```
public class BinaryTree<T extends Comparable<? super T>>{

class Node{

      T value;

      Node left,right;

      Node(T t){value=t;left=right=null;}

}

// ... etc

}
```

a) Construa um método que imprime todas as folhas da àrvore.

b) Construa um método recursivo (e correspondente método público) que remove da árvore todos os valores menores do que um valor X