

Exercícios práticos – aula 11

Java Collections Framework

1. Utilizando como base o programa abaixo, compare o tempo de execução relativo à inserção de um elemento no início, no final e no meio de uma `ArrayList<String>` e de uma `LinkedList<String>`. A que se deve a diferença de desempenho?

```
public class Main {  
  
    public static void main(String[] args) {  
        List<Integer> arrayList = new ArrayList<Integer>();  
        List<Integer> linkedList = new LinkedList<Integer>();  
  
        System.out.println("Test time Array List: " +  
testTime(arrayList)); System.out.println("Test time Linked List: "  
+ testTime(linkedList)); }  
  
    private static long testTime(List<Integer> list) {  
        long startTime = System.currentTimeMillis();  
  
        // Inserção de elementos no início da lista  
list.add(0);  
for (int i = 0; i != 100000; i++)  
    list.add(0,i);  
return System.currentTimeMillis() - startTime;  
    }  
}
```

2. Pretende-se organizar os electrodomésticos em cada divisão de uma casa segundo um mapa cuja chave de acesso é o nome de uma divisão e o valor é uma lista dos electrodomésticos que estão nessa divisão.
 - a. Crie as classes Casa e Electrodoméstico - a Casa contém o mapa dos electrodomésticos; cada electrodoméstico tem como atributos o seu tipo, a marca e a potência elétrica que consome.
 - b. Crie, na classe Casa, um método para inserir electrodomésticos no mapa,

assim como o método `toString()`, a fim de se conseguir visualizar todos os electrodomésticos, organizados segundo as divisões onde estão ligados.

- c. Faça um programa teste onde se criem alguns electrodomésticos e insira-os no mapa (use divisões diferentes). No final visualize os resultados.

3. Acrescente na classe `Casa` do exercício anterior um método que permita remover de uma divisão todos os eletrodomésticos que sejam de uma mesma marca (e.g., "Bosch").

De forma a evitar problemas relacionados com a remoção de elementos ao mesmo tempo que itera sobre uma colecção de dados, experimente três abordagens diferentes:

- a. Colocar numa lista todos os electrodomésticos da marca a remover e usar o método `removeAll()` após terminar a procura;
- b. Usar um iterador (classe `Iterator`) para ir removendo cada electrodoméstico ao mesmo tempo que se itera sobre a lista;
- c. Criar um filtro que implemente a interface `Predicate` e usar o método `removeIf()` para remover todos os electrodomésticos da marca a remover (só no Java 8).

4. Corra o programa abaixo e verifique que não é possível inserir o mesmo elemento repetido num `Set`:

```
public class Main {  
    public static void main(String[] args) {  
        Set<Integer> set = new TreeSet<Integer>();  
        Integer i = 1;  
        set.add(i);  
        set.add(i);  
        set.add(new Integer(1));  
        System.out.println(set);  
        set.add(new Integer(2));  
        System.out.println(set);  
    }  
}
```

5. Crie uma pilha de números inteiros usando a classe `Stack` e utilize os métodos `push` e `pop` para verificar que os números inseridos na pilha saem pela ordem inversa à sua inserção.
6. Use uma fila de dois topos (`Deque`) e verifique que é possível inserir e retirar elementos de ambos os topos.
7. Crie duas filas prioritárias (`PriorityQueue`): uma com números racionais (comparação intrínseca, implementação de `Comparable`) e outra com `Alunos` (comparação extrínseca, implementação de `Comparator`). Note que a fila prioritária apenas é ordenada ao esvaziar. Esvazie ambas e verifique que os elementos saem por ordem crescente.