

Prática 9

Tópicos

- Java Collections

Exercício 9.1

A classe `java.util.ArrayList<E>` é uma implementação de um vetor dinâmico fornecida pelo JavaSE. Consulte a documentação desta classe e execute o seguinte programa.

(<https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html>)

```
import java.util.ArrayList;
import java.util.Collections;

public class ALDemo {

    public static void main(String[] args) {
        ArrayList<Integer> c1 = new ArrayList<>();
        for (int i = 10; i <= 100; i+=10)
            c1.add(i);
        System.out.println("Size: " + c1.size());
        for (int i = 0; i < c1.size(); i++)
            System.out.println("Elemento: " + c1.get(i));

        ArrayList<String> c2 = new ArrayList<>();
        c2.add("Vento");
        c2.add("Calor");
        c2.add("Frio");
        c2.add("Chuva");
        System.out.println(c2);
        Collections.sort(c2);
        System.out.println(c2);
        c2.remove("Frio");
        c2.remove(0);
        System.out.println(c2);
    }
}
```

- Modifique-o livremente para testar algumas das funções que estão disponíveis nesta classe (*add*, *contains*, *indexOf*, *lastIndexOf*, *set*, *subList*, ...).
- Com base neste exemplo, crie uma nova coleção (c3) que use um `HashSet`, em vez de `ArrayList`, e que contenha elementos do tipo `Pessoa`.

```
Set<Pessoa> c3 = new HashSet<>();
...
```

Insira 5 elementos distintos e use um iterador para listar todos os elementos no écran. Verifique a ordem da listagem relativamente à ordem de inserção.

Teste a inserção de elementos repetidos (que não podem existir num `Set`).

- Crie uma nova coleção (c4) que use um `TreeSet` de datas (problema 6.1).

```
Set<Data> c4 = new TreeSet<>();
...
```

Insira 5 elementos distintos e verifique a ordem da listagem relativamente à ordem de inserção.

Exercício 9.2

Usando como base o código seguinte (*CollectionTester.java*) compare o desempenho de algumas das coleções em Java, tais como *ArrayList*, *LinkedList*, *HashSet* e *TreeSet*.

```
public class CollectionTester {

    public static void main(String[] args) {
        int DIM = 5000;

        Collection<Integer> col = new ArrayList<>();
        checkPerformance(col, DIM);
    }

    private static void checkPerformance(Collection<Integer> col, int DIM) {
        double start, stop, delta;
        // Add
        start = System.nanoTime(); // clock snapshot before
        for(int i=0; i<DIM; i++ )
            col.add( i );
        stop = System.nanoTime(); // clock snapshot after
        delta = (stop-start)/1e6; // convert to milliseconds
        System.out.println(col.size()+ ": Add to " +
            col.getClass().getSimpleName() + " took " + delta + "ms");
        // Search
        start = System.nanoTime(); // clock snapshot before
        for(int i=0; i<DIM; i++ ) {
            int n = (int) (Math.random()*DIM);
            if (!col.contains(n))
                System.out.println("Not found???" + n);
        }
        stop = System.nanoTime(); // clock snapshot after
        delta = (stop-start)/1e6; // convert nanoseconds to milliseconds
        System.out.println(col.size()+ ": Search to " +
            col.getClass().getSimpleName() + " took " + delta + "ms");
        // Remove
        start = System.nanoTime(); // clock snapshot before
        Iterator<Integer> iterator = col.iterator();
        while (iterator.hasNext()) {
            iterator.next();
            iterator.remove();
        }
        stop = System.nanoTime(); // clock snapshot after
        delta = (stop-start)/1e6; // convert nanoseconds to milliseconds
        System.out.println(col.size() + ": Remove from " +
            col.getClass().getSimpleName() + " took " + delta + "ms");
    }
}
```

- d) Adapte o programa de modo a medir os resultados para várias dimensões da coleção (por exemplo, criando uma tabela semelhante à seguinte). Analise os resultados comparando estruturas e operações.

| Collection | 1000 | 5000 | 10000 | 20000 | 40000 | 100000 |
|------------|------|------|-------|-------|-------|--------|
| ArrayList | | | | | | |
| add | 0,5 | ... | | | | |
| search | 11,5 | | | | | |
| remove | 1,2 | | | | | |
| LinkedList | | | | | | |
| ... | | | | | | |

Sugestões: modifique o método `checkPerformance` de modo a devolver as 3 medições (`add`, `search` e `remove`) e retire todas as instruções `println` dentro deste método.

```
private static double[] checkPerformance(Collection<Integer> col, int DIM) {  
    ...  
}
```