Aula prática nº 9

Tópicos

- Usar Java Collections.
- Resolução de problemas gerais de programação usando princípios de POO.

Exercícios

1. Usando como base o código seguinte compare o desempenho de algumas das coleções em Java, tais como *ArrayList*, *LinkedList*, *Stack* e *TreeSet*.

```
public class CollectionTester {

    public static void main(String[] args) {
        final int DIM = 5000;
        Collection<Integer> col;

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

    private static void checkPerformance(Collection<Integer> col, int DIM) {
        Iterator<Integer> iterator;
        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 = 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");
    }
}
```

a) 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						
...						

2. Para cada uma das tarefas seguintes, defina a(s) estrutura(s) de dados mais adequada(s), e teste-a usando 3 ou mais elementos, com as operações adicionar, listar e remover.

a) A empresa Brinca&Beira (BB) precisa de um registo com os nomes de todos os seus empregados.

b) Em cada mês é selecionado aleatoriamente um funcionário para receber um brinquedo grátis. Deve ser possível guardar todos os pares funcionário-brinquedo.

c) A empresa decidiu atribuir o primeiro nome de um empregado a cada produto. Prepare uma lista destes nomes sabendo que um nome só poderá ser usado uma vez.

d) A BB decide entretanto que só quer usar os nomes mais populares para os seus brinquedos. Precisamos de uma estrutura com o número de funcionários que têm cada primeiro nome.

e) A empresa adquire ingressos para a próxima temporada da equipa local de futebol, para serem distribuídos rotativamente pelos funcionários. Crie a estrutura mais adequada – pode usar uma ordem qualquer.