Lab X.

Objetivos

Os objetivos deste trabalho são:

- Utilizar padrões estruturais (i.e., *Chain of Responsibility, Command, Interpreter, Iterator*) para resolver casos práticos.
- Aplicar boas práticas de programação por padrões

Nota: Para além do código no github, inclua também um ficheiro PDF ou PNG com o diagrama de classes da solução final.

X.1 Implementação de iteradores sobre um conjunto

Considere o código seguinte:

```
public class VectorGeneric<T> {
   private T[] vec;
   private int nElem;
   private final static int ALLOC = 50;
   private int dimVec = ALLOC;
   @SuppressWarnings("unchecked")
   public VectorGeneric() {
      vec = (T[]) new Object[dimVec];
      nElem = 0;
   }
   public boolean addElem(T elem) {
      if (elem == null)
         return false;
      ensureSpace();
      vec[nElem++] = elem;
      return true;
   }
   private void ensureSpace() {
      if (nElem>=dimVec) {
         dimVec += ALLOC;
         @SuppressWarnings("unchecked")
         T[] newArray = (T[]) new Object[dimVec];
         System.arraycopy(vec, 0, newArray, 0, nElem );
         vec = newArray;
      }
   }
   public boolean removeElem(T elem) {
      for (int i = 0; i < nElem; i++) {
         if (vec[i].equals(elem)) {
            if (nElem-i-1 > 0) // not last element
               System.arraycopy(vec, i+1, vec, i, nElem-i-1 );
            vec[--nElem] = null; // libertar último objecto para o GC
            return true;
         }
      }
      return false;
   }
   public int totalElem() {
      return nElem;
   }
   public T getElem(int i) {
      return (T) vec[i];
   }
}
```

a) Construa o código necessário para que a classe passe a incluir os seguintes métodos:

```
public java.util.Iterator<E> Iterator()
public java.util.ListIterator<E> listIterator()
public java.util.ListIterator<E> listIterator(index) // start at index
```

Não implemente os métodos opcionais e respeite rigorosamente os contratos especificados na documentação java8.

b) Desenvolva uma classe de teste para verificar todas as operações criadas. Inclua a situação de usar vários iteradores em simultâneo sobre o mesmo conjunto.

X.2 Chain of Responsibility

Um restaurante tem vários chefes de cozinha, cada um com a sua especialidade. Quando um pedido chega, o primeiro chefe confeciona o pedido se se tratar da sua especialidade; caso contrário, passa o pedido ao chefe seguinte e assim consecutivamente. Implemente e teste a solução, tentando replicar o seguinte resultado.

```
Can I please get a veggie burger?
SushiChef: I can't cook that.
PastaChef: I can't cook that.
BurgerChef: Starting to cook veggie burger. Out in 19 minutes!

Can I please get a Pasta Carbonara?
SushiChef: I can't cook that.
PastaChef: Starting to cook Pasta Carbonara. Out in 14 minutes!

Can I please get a PLAIN pizza, no toppings!?
SushiChef: I can't cook that.
PastaChef: I can't cook that.
BurgerChef: I can't cook that.
PizzaChef: Starting to cook PLAIN pizza, no toppings!. Out in 7 minutes!

Can I please get a sushi nigiri and sashimi?
SushiChef: Starting to cook sushi nigiri and sashimi. Out in 14 minutes!

Can I please get a salad with tuna?
SushiChef: I can't cook that.
PastaChef: I can't cook that.
BurgerChef: I can't cook that.
PizzaChef: I can't cook that.
DessertChef: I can't cook that.
We're sorry but that request can't be satisfied by our service!

Can I please get a strawberry ice cream and waffles dessert?
SushiChef: I can't cook that.
PastaChef: I can't cook that.
BurgerChef: I can't cook that.
PizzaChef: I can't cook that.
DessertChef: Starting to cook strawberry ice cream and waffles dessert. Out in 17 minutes!
```

X.3 Command

Usando o padrão *Command*, construa uma classe para adicionar um elemento a uma coleção (*java.util.Collection*<*E*>) permitindo realizar a operação *undo*. Repita a metodologia para uma classe que remova um elemento de uma coleção (com possibilidade de *undo*).