**Instituto Superior de Engenharia de Lisboa**
**Programação III**
2º Teste – 24 de Janeiro de 2024 (Duração: 1:00)

ISEL ADEETC

## Grupo I

```
public interface Comparator<E> {
   int compare(E e1, E e2);
}
```

```
public interface Supplier<S> {
  // Retorna um S
  S get( );
}
```

**a)** [3] Realize o método com a seguinte assinatura:

```
public static <V, S extends Collection<V>>
S getSorted( Iterable<V> seq,
             Comparator<V> cmp,
             Supplier<S> ss)
```

Que com os elementos da sequência não vazia **`seq`** produz e retorna uma coleção ordenada de forma estritamente crescente (sem elementos repetidos) segundo o comparador **`cmp`**. Para tal, deve percorrer a sequência **`seq`** e adicionar na nova coleção os elementos que garantam que a nova coleção esteja sempre ordenada. A coleção a retornar é obtida com o **`Supplier ss`**.

Exemplos:

**`getSorted`**( List.of(**5**, **8**, 2, **9**, 3, 9, **70**, 70), Comparator.naturalOrder(), ArrayList::new) → [5, 8, 9, 70]

**`getSorted`**( List.of(30, 2, 1, 4, 10, 15, 20), Comparator.naturalOrder(), ArrayList::new) → [30]

**b)** [3] Utilizando o método da alínea anterior e implementando o **`Comparator`** e o **`Supplier`** realize o método com a seguinte assinatura:

```
public static LinkedList<String> getDecrescent( List<String> words )
```

que retorna uma lista ligada, ordenada de forma decrescente, com as *strings* contidas na lista **`words`**.

**c)** [3] Realize o método com a seguinte assinatura:

```
public static <K, V extends Comparable<V>>
List<K> greater(SortedMap<K, V> map)
```

```
public interface Comparable<E> {
   int compareTo(E e);
}
```

Que produza uma lista ordenada com as chaves do contentor associativo **`map`**, cujo valor associado é o maior segundo a ordem natural.

Nota: O contentor associativo **`map`**, não pode ser alterado.

Exemplo:

Seja: **`SortedMap<String, Integer> map;`**

Se **`map.toString()`** -> "{Ana= **20**, Carla = 18, David= **20**, Maria= 18, Pedro= 13, Rita= **20**, Rui= 14}"

**`greater(map).toString()`** -> "[Ana, David, Rita]"

**Instituto Superior de Engenharia de Lisboa**
**Programação III**
2º Teste – 24 de Janeiro de 2024 (Duração: 1:00)

ISEL ADEETC

## Grupo II

```
public interface BiConsumer<E1, E2> {
  // Executa a ação sobre e.
  void accept(E1 e1, E2 e2);
}
```

**a)** [3] Realize o método estático com a seguinte assinatura:

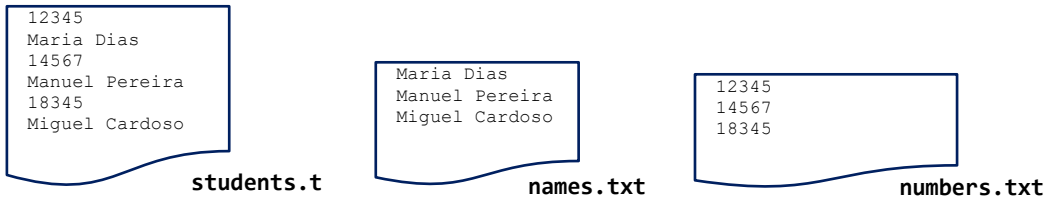
```
void getLines( String filename,
               BiConsumer<Integer, String> action ) throws IOException
```

Que recebendo por parâmetro o nome de um ficheiro de texto, por cada linha lida do ficheiro invoca o método **`accept`** de **`action`** passando-lhe por parâmetro o número da linha e a linha.

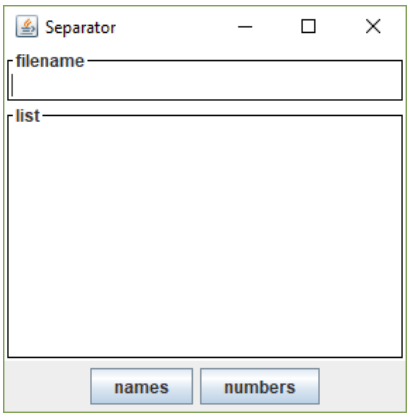
**b)** [3] Utilizando o método da alínea a) realize o método estático público com a seguinte assinatura:

```
void separate( String filename ) throws IOException
```

Que recebendo o nome de um ficheiro de texto **`filename`** em que alternadamente tem linhas com um número e um nome separe em dois ficheiros um com os números ("*numbers.txt*") e outro com os nomes ("*names.txt*"). Exemplo caso **`filename`** seja "*students.txt*" o ficheiro "*numbers.txt*" contém os números, e o ficheiro "*names.txt*" contém os nomes:




**c)** [5] Faça uma aplicação com o aspeto da figura. Quando for premido o botão "*names*" ou o botão "*numbers*" deve ser chamado o método **`getLines`** da alínea a), para escrever na área de texto "*list*" as linhas do ficheiro que contêm os nomes ou as que contêm os números respetivamente. O nome do ficheiro de texto é o que está presente na caixa de texto "*filename*" quando os botões são premidos. Assuma que o ficheiro de texto tem alternadamente linhas com um número e um nome. Em caso de erro no acesso aos ficheiros deve ser escrito numa janela de diálogo a mensagem associada ao erro.




Para a instânciação das caixas de texto e da área de texto use os métodos:

```
public static JTextField newJTextField( String title ) {
  JTextField tf = new JTextField( 15 );
  tf.setBorder( new TitledBorder( title ) );
  return tf;
}

public static JTextArea newJTextArea( String title ) {
  JTextArea ta = new JTextArea( 15, 30 ); ta.setBorder(new TitledBorder( title ));
  return ta;
}
```

Método para a escrita da mensagem na janela de diálogo:

```
public static void showMessage( String msg ) { JOptionPane.showMessageDialog( null, msg );}
```

Exemplos quando é premido o botão "*names*" e quando é premido o botão "*numbers*" :