Lista de Exercícios sobre Coleções

1º As ordenações disponíveis em Collections do Java são estáveis, isso significa que elementos iguais não serão reordenados como resultado da classificação. Resolva o problema URI 1244 usando uma implementação de List disponível no Collections Framework. Para resolver este exercício é **obrigatório** o uso de **iterator** para mostrar as Strings ordenadas.
Dicas:

1. Para ler da entrada padrão (teclado), usa-se o System.in. A classe Scanner pode ser usada para simplificar a leitura do System.in.
   ```
   Scanner tecl = new Scanner(System.in);
   ```
2. Para ler um inteiro do teclado:
   ```
   int n = tecl.nextInt(); // lê o próximo inteiro
   tecl.nextLine(); // lê a linha até o final e pula para a próxima
   ```
3. Para ler uma linha:
   ```
   String linha = tecl.nextLine();
   ```
4. Para separar a linha em um vetor de palavras:
   ```
   String pals[] = linha.split(" ");
   ```
5. Para transformar um vetor de palavras em uma implementação de List:
   ```
   List<String> palavras = Arrays.asList(pals);
   ```
6. Para ordenar use:
   ```
   palavras.sort(new Comp());
   ```
   Note que a ordenação acima não leva em consideração a ordem alfabética das Strings, mas sim uma ordenação definida pela classe Comp.
   A classe Comp deve implementar Comparator, sugestão:
   ```
   class Comp implements Comparator<String> {}
   ```
7. Use iterator para mostrar as palavras ordenadas.
8. Atenção, não permita que um caractere espaço seja impresso ao final de cada linha, se isso acontecer o seu programa não passará no juiz do URI.

2º Resolva o problema URI 2174. É **obrigatório** o uso de **TreeSet<String>** para definir o conjunto de pomekons.
Como o conjunto criado com TreeSet não aceita repetições, basta inserir todos os pokemons na lista, mesmo se repetidos, pois eles serão inseridos apenas uma vez.

3º Resolva o problema URI 2727. É **obrigatório** o uso de **TreeMap<String, Character>** para decodificar a String lida.
A iniciação do Map pode ser feito de forma programática ou manualmente, adicionando os 26 códigos um a um para cada caractere.

Definindo:
```
Map<String, Character> decodifica = new TreeMap<String, Character>();
```
Manualmente, por exemplo, para a letra “i”, seria:
```
decodifica.put("... ... ...", 'i');
```
Para decodificar, basta ler o código de cada linha em uma String e usá-la como chave na função get do decodifica.

Dica: para repetir até que os dados acabem, use:
```
while (tecl.hasNextLine()) {
```

Obs.: os tipos Generics, que são definido entre o “<” e o “>”, não aceitam tipos primitivos, portanto o tipo *char* precisa ser substituído por Character que é uma classe. A conversão do tipo primitivo char para uma instância de Character acontece automaticamente.