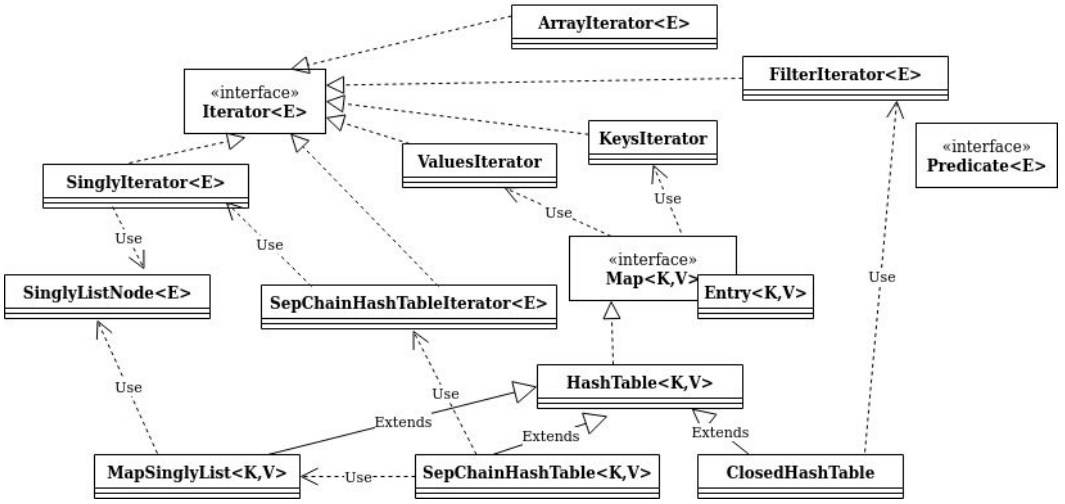
Exercício #1 – TAD Map

A.

- **Implemente** as classes fornecidas no pacote dataStructures:
  - Map
  - SepChainHashTable
  - HashTable
  - SepChainHashTableIterator
  - KeysIterator
  - ValuesIterator

- **Implemente** também a classe:
  - MapSinglyList



Exercício #1 – Programa TextProcessor

B. Para testar as implementações realizadas no exercício #1-A, implemente o programa TextProcessor.

O programa TextProcessor permite construir um índice de palavras sobre um texto.

O programa deverá suportar textos com várias linhas. Assuma que não existem caracteres especiais (por exemplo, @, %, &, (), etc.), nem acentos ou apostrofes. Não existem também números nem palavras separadas por hifens. Contudo, podem existir um ou mais sinais de pontuação (. ; , : ? !) juntos ao fim de cada palavra (mas nunca diretamente antes ou no meio de uma palavra). Por fim, assuma que palavras com a mesma sequência de letras mas capitalização diferentes são iguais. Por exemplo, AED, AeD e aed são a mesma palavra.

Exercício #1 – Programa TextProcessor

Pretende-se suportar os seguintes comandos:

- **New**: lê um texto do input e cria o índice de palavras sobre o mesmo.
- **Count**: conta quantas palavras distintas existem.
- **Where**: indica em que linhas uma dada palavra ocorre.
- **Remove**: remove uma palavra do índice.
- **Help**: lista os comandos suportados pela aplicação.
- **Exit**: termina a aplicação.

Os comandos são case insensitive. Ou seja, tanto New, new, NEW, NeW são reconhecidos como sendo o comando New.

Em todos os comandos assuma que a formatação do input está correta, e que os únicos erros que podem ocorrer são os descritos em cada comando.

Exercício #1 – Programa TextProcessor

```
public enum Command {
    NEW(”new - insert a text to analyze."),
    COUNT(”count - count the number of different words in the text"),
    WHERE(”where - list the lines in which a given word occurs in the text."),
    REMOVE(”remove - remove a word. After executing this command, the remaining
commands should behave as if this word is no longer present in the text."),
    HELP(”help - list all commands."),
    EXIT(”exit - exit the program."),
    UNKNOWN("");
    ...

    String helpMsg;

    Command(String s) {
        helpMsg = s;
    }

    public String getMsg() {
        return helpMsg;
    }
}
```

Exercício #1 – Comando New

Lê um texto do input e constrói um índice de palavras sobre o mesmo. O comando recebe dois argumentos (integers), por esta ordem: nº **linhas (l)** e nº estimado de **palavras (p)** diferentes (no texto completo). Note que **p** é uma estimativa e, por isso, pode não estar correto. De seguida, o comando salta para a linha seguinte de input, e lê **l** linhas de input contendo o texto a analisar. Assuma que **l** e **p** são inteiros positivos. Assuma também que o texto tem sempre pelo menos uma palavra (mas uma linha pode ser vazia). Se já existia um índice sobre outro texto antes, esse índice é apagado e criado um novo de raiz. O comando sucede sempre. No fim, o comando imprime a mensagem “Text added successfully.”.

Exemplo:

```
new 2 10↵
This is an example text.↵
Was the command description clear enough?↵
Text added successfully.↵
```

Exercício #1 – Comando Count

Imprime o nº de palavras diferentes no texto. O comando não recebe nenhum argumento. Note que duas palavras que tenham a mesma lista de letras mas com capitalizações diferentes são consideradas uma mesma palavra. Por exemplo, CoUnt e count são consideradas a mesma palavra. O comando falha se não tiver sido inserido um texto e imprime a mensagem “No text added yet!”. Caso contrário, é imprimido a mensagem “The text has ***x*** different words”, onde **x** é o número de palavras diferentes no texto.

Exemplo:

```
count↵
No text added yet!↵
new 2 13↵
I am an example text for count.↵
Notice how all the upcoming words count as one: Count, CounT, CoUnT, CoUNT, CouNt, COUNT, counT.↵
Text added successfully.↵
count↵
The text has 15 different words.↵
```

Exercício #1 – Comando Where

Indica as linhas onde a palavra ocorre no texto. O comando recebe como único argumento a **palavra** a procurar. A capitalização da **palavra** não é relevante – se o texto contiver a palavra “aqui”, tanto **where aqui** como **where AqUi** devem encontrar a palavra. O comando falha se a **palavra** não existir no texto (“Word **aVeRyComPliCaTEDWord** does not occur anywhere in the text.”), ou se não tiver sido inserido nenhum texto (“No text added yet!”). Em caso de sucesso, a mensagem “Word **AqUi** occurs in the following lines: ” é impressa, seguido do nº das linhas (de 1 a n) em que ocorre, separado por vírgula, sem repetições.

Exemplo:

new 3 10↵
Hello. A small text that does not have a very complicated word. aqui! Aqui?↵
Notice how where command can find words even if they appear with a different capitalization. AqUi!↵
aQUi. AQui! AQUI!↵
Text added successfully.↵
where **aVeRyComPliCaTEDWord**↵
Word aVeRyComPliCaTEDWord does not occur anywhere in the text.↵
where **aqui**↵
Word aqui occurs in the following lines: 1, 2, 3↵

Exercício #1 – Comando Remove

Remove a palavra recebida como argumento do índice. O comando recebe como único argumento a **palavra** a remover. A capitalização da **palavra** não é relevante – se o texto contiver a palavra “aqui”, tanto **remove aqui** como **remove AqUi** removem a palavra. O comando falha se a **palavra** não existir no texto (“Word **aVeRyComPliCaTEDWord** does not occur anywhere in the text.”), ou se não tiver sido inserido nenhum texto (“No text added yet!”). Em caso de sucesso, a mensagem “Word **AqUi** removed successfully. ” é impressa.

Exemplo:

new 3 10↵
Hello. A small text that does not have a very complicated word. aqui! Aqui?↵
Notice how where command can find words even if they appear with a different capitalization. AqUi!↵
aQUi. AQui! AQUI!↵
Text added successfully.↵
remove **aqui**↵
Word aqui removed successfully.↵
where **aqui**↵
Word aqui does not occur anywhere in the text.↵
remove **big**↵
Word big does not occur anywhere in the text.↵

Exercício #1 – Comando Help

Lista os comandos disponíveis. O comando não recebe nenhum argumento. O comando sucede sempre.

Exemplo:

```
HElp↵
new - insert a text to analyze.↵
count - count the number of different words in the text.↵
where - list the lines in which a given word occurs in the text.↵
remove - remove a word. After executing this command, other commands should behave as if this word is
no longer present in the text.↵
help - list all commands.↵
exit - exit the program.↵
```

Exercício #1 – Comando Exit

Termina a execução do programa. O comando não recebe nenhum argumento. O comando sucede sempre.

Exemplo:

```
EXIT↵
Bye!↵
```

Desafio #5 – TAD Map

C. Submeta o programa (programa TextProcessor + pacote dataStructures com a implementação do **TAD Map**) ao mooshak no concurso AED2025_Aulas problema D.

Este exercício será considerado como desafio na sua avaliação da componente prática. Caso seja considerado para avaliação, a nota deste desafio corresponde à nota aceite no mooshak (nota entre 0 e 20 valores).

Desafio #5 – TAD Map

C. Submeta o programa (programa TextProcessor + pacote dataStructures com a implementação do **TAD Map**) ao mooshak no concurso AED2025_Aulas problema D.

As regras que deve cumprir para que o desafio seja considerado são:

- Submeter o seu programa no concurso AED2025_Aulas utilizando como seu identificador (número de aluno);
- Submeter até às 18 horas do dia 12 de novembro de 2025 a sua solução.
  - O ficheiro que submete deve ter extensão . zip e conter o programa TextProcessor mais o pacote de dataStructures fornecido implementado por si.
  - O ficheiro Main deve estar na raiz do ficheiro .zip (à semelhança dos desafios anteriores). Deve criar um pacote para a funcionalidade do TextProcessor.

Desafio #5 – TAD Map

- O programa entregue deve:
  - Ser da autoria do aluno que o entrega;
  - A classe Main e qualquer classe relacionada com o TextProcessor deve ser criada por si.
  - Todas as interfaces e classes do pacote dataStructures só poderão ser alteradas quando pedido (//TODO: Left as an exercise).
  - Não podem acrescentar variáveis de instância às classes fornecidas.
  - Não podem usar o pacote java.util.