Este primeiro trabalho de programação em Java consiste na implementação parcial de dois tipos de dados que estão descritos no livro *Estruturas de Dados e Algoritmos em Java*, de António Adrego da Rocha publicado pela editora FCA em 2011. E de programas de testes unitários para validar a correção dos métodos já implementados ou por implementar.

Pretende-se implementar o tipo de dados ***Time*** utilizando programação defensiva com lançamento de excepções no caso de situações de dados anormais que violam as pré-condições e/ou as pós-condições dos métodos. Para esse efeito é disponibilizada uma classe de excepção designada por *InvalidValueException* para indicar quando uma qualquer informação horária excede por defeito ou por excesso o valor aceitável para as horas, ou minutos ou segundos ou um tempo total de um dia em segundos. A outra classe de excepção utilizada *NullPointerException* já existe na linguagem Java.

1. Começe por ler a descrição da classe ***Time*** (documento **Time.html**). Depois implemente os métodos em falta *addTimes* e *subTimes*. Em caso algum deve alterar os restantes métodos senão o programa de testes unitários pode deixar de funcionar.

2. Complete o programa de testes unitários **TestTime.java**. Acrescente testes utilizando a instrução *assert* para validar a correção dos métodos em falta: *getHours*; *getMinutes*; *getSeconds*; *setHours*; *setMinutes*; *setSeconds*; *compareTo*; *addTimes* e *subTimes*.

Pretende-se implementar o tipo de dados ***Memory*** que representa uma memória RAM. Uma memória deste tipo serve para fazer pesquisa, seleção e ordenação de valores. É uma implementação paramétrica que aceita apenas elementos que implementam a interface *Comparable*. Utiliza programação defensiva com lançamento de excepções no caso de situações de dados anormais que violam as pré-condições e/ou as pós-condições dos métodos. Para esse efeito são disponibilizadas as classe de excepção: *IndexOutOfMemoryException*; *MemoryEmptyException* e *MemoryFullException*. A outra classe de excepção utilizada *NegativeArraySizeException* já existe na linguagem Java.

1. Começe por ler a descrição da classe ***Memory*** (documento **Memory.html**). Depois implemente os métodos em falta: *insertPos*; *deletePos*; *biggerElement* e *smallerElement*. Em caso algum deve alterar os restantes métodos.

2. O programa de simulação **SimMemoryTime.java** permite simular uma memória RAM para armazenar informações horárias da classe ***Time***. Faça as simulações que achar necessárias para testar os métodos já fornecidos e os que tem que implementar. Também são disponibilizados dois ficheiros que representam duas memórias com tamanho 10: o *times.csv* armazena 8 informações horárias válidas, enquanto que o *empty.csv* não armazena qualquer informação horária. Este último serve para testar os métodos na situação de memória vazia.

3. Complete o programa de testes unitários **TestMemoryTime.java** que apenas testa os métodos *biggerElement* e *smallerElement*. Acrescente testes utilizando a instrução *assert* para validar a correção dos métodos em falta: *getElement*; *setElement*; *insert*; *insertPos*; *delete*; *deletePos*; *search*; *copy* e *clear*).

O trabalho deve ser entregue até 30 de Abril de 2022. Apenas um dos alunos do grupo deve enviar por email para adrego@ua.pt um arquivo zip contendo apenas os ficheiros finais das classes e dos programas de testes unitários: **Time.java**, **TestTime.java**, **Memory.java** e **TestMemoryTime.java**.

Para quaisquer questões de Java pode consultar o livro *Estruturas de Dados e Algoritmos em Java*. Também pode e deve contactar-me por *email* com dúvidas acerca do trabalho.