Estruturas de Informação

Relatório

1º Trabalho Prático

Turma 2DL
Ricardo Mesquita 1190995
Gonçalo Jordão 1190633

isep Instituto Superior de Engenharia do Porto

Requisitos

1. **Armazenamento de informação do ficheiro**

Foi utilizado o seguinte *Map* para armazenar a informação que foi fornecida, tendo sido esta disponibilizada por um ficheiro de Excel:

```
public Map<ISO, TreeMap<LocalDate, Data>> map;
```

Foi realizado desta maneira para que fosse possível conseguir separar os diferentes países (Classe *ISO*) da restante informação, e para que cada um deles pudesse armazenar várias datas (*LocalDate*), ordenadamente (daí a utilização do *TreeMap*), e a respetiva informação (Classe *Data*) para cada uma delas.

Classe ISO:

```
public ISO(String code, Continent continent, String location, String population, String aged65Older, String cardiovascDeathRate,
          String diabetesPrevalence, String femaleSmokers, String maleSmokers, String hospitalBedsPerThousand, String lifeExpectancy) {
    this.code = this.checkStringData(code);
    this.continent = continent;
    this.location = this.checkStringData(location);
    this.population = this.checkPopulationData(population);
    this.aged65Older = this.checkNumericData(aged65Older);
    this.cardiovascDeathRate = this.checkNumericData(cardiovascDeathRate);
    this.diabetesPrevalence = this.checkNumericData(diabetesPrevalence);
    this.femaleSmokers = this.checkNumericData(femaleSmokers);
    this.maleSmokers = this.checkNumericData(maleSmokers);
    this.hospitalBedsPerThousand = this.checkNumericData(hospitalBedsPerThousand);
    this.lifeExpectancy = this.checkNumericData(lifeExpectancy);
}
```

Chegou-se ao consenso de guardar os atributos: *code*, *continent* e *location* para cada país, visto que os valores destes são únicos. Foram também guardados os atributos *femaleSomekers*, *maleSmokers, population, aged65Older, cardiovascDeathRate, diabetesPrevalence, hospitalBedsPerThousands,* and *lifeExpectancy* nesta classe, dado que se tratam de constantes para cada país ao longo do ficheiro.

Classe Data:

```
public Data(String totalCases, String newCases, String totalDeaths, String newDeaths, String totalTests, String newTests) {
    this.totalCases = this.checkNumericData(totalCases);
    this.newCases = this.checkNumericData(newCases);
    this.totalDeaths = this.checkNumericData(totalDeaths);
    this.newDeaths = this.checkNumericData(newDeaths);
    this.totalTests = this.checkNumericData(totalTests);
    this.newTests = this.checkNumericData(newTests);
}
```

Nesta classe foram guardados os valores que variam de data para data em cada país.

2. **Armazenamento de informação de países com casos positivos superiores a 50 000.**

   Neste procedimento, foi utilizada a seguinte estrutura de informação:

```
Map<String, Integer> mapAux = new HashMap<>();
```

   Inicialmente foi pensado em fazer-se um *Map* mais extenso tendo como *chave* a classe ISO e como *value* um outro *Map* com *chave LocalDate* e *value Integer* (numero mínimo de dias necessários para atingir os 50 000 casos).
   No entanto, mais tarde, foi acordado fazer a compactação deste *Map*, guardando o ISO e a *LocalDate* na *String*, isto porque não havia benefícios de o fazer tão extenso pois não era necessário aceder às duas informações individualmente, visto que a ordenação era feita através do número mínimo de dias (*Integer*).

3. **Armazenamento de novas mortes e novos casos para cada mês de cada continente.**

   Foi utilizado o seguinte *TreeMap* para armazenar a informação:

```
TreeMap<String, TreeMap<Integer, List<Integer>>> mapAux = new TreeMap<>();
```

   Neste caso utilizamos o *TreeMap* para que fosse feita a ordenação automática dos dados à medida que estes eram armazenados. Estas ordenações foram feitas para os continentes (*String*) e para os meses (*Integer*).
   Já o *List* (*Integer*), foi utilizado para guardar numa estrutura mais simples tanto o número de novos casos (posição 0), como o de novas mortes (posição 1).

4. **Armazenamento de casos diários de um mês e continente escolhidos.**

   Neste procedimento, foi utilizada a seguinte estrutura de informação:

```
TreeMap<Integer, HashMap<String, Integer>> mapAux = new TreeMap<>();
```

   Nesta situação a utilização do Treemap foi feita para ordenar os dias do mês (*Integer*) e o *HashMap* para guardar o país (*String*) e os respetivos novos casos (*Integer*).
   Este *HashMap* foi posteriormente ordenado por ordem decrescente através dos *values*.

5. **Armazenamento dos países com percentagem de fumadores superior a 70% e respetivo total de mortes.**

   Foi utilizado o seguinte *Map* para guardar a informação:

```
Map<ISO, Integer> mapAux = new HashMap<>();
```

   A razão da escolha deste *Map*, foi o facto de serem precisos apenas os dados do país (classe *ISO*), devido à localização bem como a percentagem de fumadores e a necessidade de guardar o número de mortes respetivo (*Integer*), que depois foi utilizado para que fosse possível a respetiva ordenação da informação, como era solicitado.

6. Outras Classes utilizadas

Classe Constants:

Esta classe foi utilizada meramente para guardar variáveis que foram consideradas constantes ao longo de todo o projeto.

Assim, caso fosse necessário alterar o valor de alguma destas variáveis, bastava apenas aceder a esta classe.

```java
public class Constants {

    /**
     * The constant FILE_PATH.
     */
    public static final String FILE_PATH = "Files\\owid-covid-data.csv";
    /**
     * The constant NA_STRING.
     */
    public static final String NA_STRING = "NA";
    /**
     * The constant CASES.
     */
    public static final int CASES = 50000;
    /**
     * The constant INITIAL_DATE.
     */
    public static final LocalDate INITIAL_DATE = LocalDate.of( year: 2020, Month.JANUARY,  dayOfMonth: 1);
    /**
     * The constant SMOKERS_PERCENTAGE.
     */
    public static final double SMOKERS_PERCENTAGE = 70.0;
    /**
     * The constant FILE_SPLIT.
     */
    public static final String FILE_SPLIT = ",";
    /**
     * The constant DATE_SPLIT.
     */
    public static final String DATE_SPLIT = "-";

}
```

Classe Continent:

Esta classe foi feita no âmbito de facilitar a validação dos continentes, no que serve de exemplo ao exercício 4, onde o continente inserido pelo utilizador é verificado pelo método *setContinent*.

```java
public enum Continent {

    /**
     * Asia.
     */
    AS() {
        @Override
        public String toString() { return "\"Asia\""; }
    },
    /**
     * Europe.
     */
    EU() {
        @Override
        public String toString() { return "\"Europe\""; }
    },
    /**
     * Africa.
     */
    AF() {
        public String toString() { return "\"Africa\""; }
    },
    /**
     * North America.
     */
    NA() {
        @Override
        public String toString() { return "\"North America\""; }
    },
```

```java
public static Continent setContinent(String continent) {
    for (Continent c : Continent.values()) {
        if (c.toString().equalsIgnoreCase(continent) || c.toString().equalsIgnoreCase( anotherString: "\"" + continent + "\"")) {
            return c;
        }
    }
    return null;
}
```