ESINF

Relatório

1º Trabalho

Prático

Turma 2DL
João Gomes, 1210818
André Gonçalves, 1210804
Manuel Silva, 1200585
Jorge Moreira, 1201458
Alexandre Vieira, 1211551

isep Instituto Superior de Engenharia do Porto

Requisitos

1. **Carregar e guardar a informação de um ficheiro (formato CSV) em objetos adequados à eficiente pesquisa de informação**

   Classe- FileReaderData

```
public ArrayList<Data> readData(String filename) throws IOException, ParseException{
```

   Nesta classe existe apenas um método que irá receber como argumento o nome do ficheiro csv que será guardado.

```
    FileReader readFile = new FileReader(filename);
    String line;
    BufferedReader bufferedReader = new BufferedReader(readFile);

    // discards the initial line
    bufferedReader.readLine();
    Scanner sc = new Scanner(System.in);

    ArrayList<Data>  dataList = new ArrayList<>();
    while ((line= bufferedReader.readLine()) != null) {
```

   Através de um buffer iremos percorrer o ficheiro até que este encontre uma linha null, que será a última linha.

```
if(line.contains("\"")) { //BIG
    line = line.replace( target: "\",\"", replacement: "|");
    line = line.replace( target: "\"", replacement: "");
}
```

   Caso o ficheiro selecionado seja o big.csv irá haver uma substituição de “,” por “|”

   E de \” por “ “ para que a leitura do ficheiro seja possível

```
else { //small
    line = line.replace( target: ",", replacement: "|");
}
```

Caso o ficheiro seja o small.csv a troca será apenas de, para "|"

```
if(Objects.equals(info[11], b: "")){

    info[11] = String.valueOf(0);
}

 Data d1 = new Data(info[0],info[1],info[2],new Area(info[3]),info[4],
         info[5],info[6], new Item(info[7]),info[8], new Year(Integer.parseInt(info[9])), info[10],
         new Value(Integer.parseInt(info[11])), info[12], info[13]);


dataList.add(d1);
```

O primeiro if aqui será caso não haja valor, pois há linhas no big que não têm alguns dados, este valor será assumido como 0.

À medida que cada linha é lida a Data vai ser guardada e adicionada numa lista

2. **Armazenamento e organização de forma crescente de decrescente de paises que tiveram produção de um fruto F maior que Q**

Para primeiro retirar ao ficheiro que é lido, as frutas que não são pedidas, é iterado o primeiro *ArrayList* e removidos os membros que não equivalem ao fruto que é dado.

```
while (it.hasNext()) {
    Data data = it.next();
    if (data.getItem().getItem().equals(I)) {
        fruitArray.add(data);
    }
}
```

Da mesma maneira são retirados os membros do *arrayList* que são inferiores ao valor de produção dado.

```
while (it1.hasNext()) {
    Data data = it1.next();
    if (data.getValue().compareTo(Q)==-1){
        it1.remove();
    }
}
```

Para que fosse possível descobrir o valor mínimo superior a Q (valor de produção dado pelo utilizador), é corrido o *array* e usando o método *compareTo* verifica-se o valor mínimo de cada país que se encontra no ficheiro.

```
while (it2.hasNext()){
    Data data=it2.next();
    min=data.getValue();
    String area= data.getArea().getArea();
    while (data.getArea().getArea().equals(area)&&it2.hasNext()){
        if (data.getValue().compareTo(min)==-1){
            min=data.getValue();
            maxArray[m]=min;
        }
        data=it2.next();
    }
    m++;
}
```

Para retirar em seguida os valores superiores a este mínimo de cada país usamos um *iterator* para percorrer e retirar esses valores.

```
while (it3.hasNext()){
    Data data=it3.next();
    String area=data.getArea().getArea();
    while (data.getArea().getArea().equals(area) &&it3.hasNext()){
        if (data.getValue().compareTo(maxArray[m])==1){
            it3.remove();
        }
        data=it3.next();
        if (!data.getArea().getArea().equals(area)){
            it3.previous();
            break;
        }
    }
    m++;
}
```

Infelizmente depois de retirar todos os elementos incorretos, o último elemento da lista que não é o mínimo, não é apagado, logo é necessário usando o *iterator* voltar atrás e removê-lo manualmente.

Então finalmente organizamos primeiro de forma crescente e em seguida de forma decrescente o *arrayList*.

```
Collections.sort(fruitArray); // Sorts the array in as
Collections.sort(reverse,Collections.reverseOrder());
```

Finalmente é enviado os dados para uma lista de países onde são guardados.

```
ArrayList<Area> areaList=new ArrayList<>();
```

Para melhorar:

O *loop* que descobre o mínimo valor de cada país e o *loop* que retira os membros que são maiores que o mínimo poderia ser apenas um *loop*, mas infelizmente enquanto

tentávamos criar um *loop* assim, ocorriam muitos erros, o que a criarmos dois *loops* separados. O método em geral poderia estar mais simples e eficiente, existindo algumas partes que poderiam ser simplificadas.

3. **Dada uma quantidade de produção Q, descobrir qual o número mínimo de países que, em conjunto, consegue ter uma quantidade de produção superior a Q.**

   Classe – *MinimalQuantityController*:

   - Dentro desta classe foi criado uma função (do tipo *int*) que recebe como argumento um objeto do tipo *Value*, objeto este que corresponde ao valor de produção Q;

```
public int minimalNumber(Value Q) throws IOException, ParseException
{
```

   - Seguidamente é criado um ciclo "for" de forma a percorrer a informação existente no ficheiro *CSV* pretendido.

     Dentro do ciclo criado é carregada a informação necessária, como o código do país (*areaCode – String*) e o valor de produção (*value – Value*) num *Map* (*Map<String,Long> minimoCountry = new HashMap<String,Long>();*).

     Se para cada país só existir um único valor de produção é automaticamente adicionado ao *Map* o código do país e o valor de produção correspondente ao mesmo, mas caso exista mais que um valor de produção num mesmo país é então realizada a soma de todos esses valores para cada país e adicionado ao *Map*

```
Map<String, Long> minimoCountry = new HashMap<~>();
ArrayList<Data> fileReaderData;
fileReaderData = new FileReaderData().readData( filename: "FAOSTAT_data_en_9-7-2022_SMALL.csv");

  for(Data data : fileReaderData)
  {
      if(minimoCountry.containsKey(data.getAreaCode()))
      {
          long sum = minimoCountry.get(data.getAreaCode()) + data.getValue().getValue().longValue();
          minimoCountry.put(data.getAreaCode(), sum); //put in the map the values
      }
      else
      {
          minimoCountry.put(data.getAreaCode(), data.getValue().getValue().longValue());
      }

  }
```

- Depois de adicionado todos os dados necessários é feita uma organização de todos os valores de produção carregados dentro do *Map*.

```
SortedSet<String> keys = new TreeSet<>(minimoCountry.keySet()); //Sort all the values
```

Estando os dados organizados é adicionado todos os valores de produção de cada país e caso este valor seja superior ao valor de produção Q (valor dado pelo utilizador) o programa retorna o número mínimo de países necessários para atingir o valor de produção Q, no caso de não ser superior é acrescentado o próximo valor de produção e incrementado o número de países necessários para atingir o valor de produção Q.

```
long somaCountries = 0;
int numberOfCountries = 0;
SortedSet<String> keys = new TreeSet<>(minimoCountry.keySet());
for (String key : keys)
{
    somaCountries += minimoCountry.get(key);
    numberOfCountries++;

    if (somaCountries > Q.getValue())
    {
        return numberOfCountries;
    }
}

return 0;
```

4. **Dado um fruto F, devolver, numa estrutura de dados adequada, os países agrupados pelo número máximo de anos consecutivos em que houve crescimento de quantidade de produção do fruto F.**

Classe- *SuperiorQualityController*:

Nesta classe funcionam dois métodos, o primeiro método:

```
public ArrayList<Data> getCountriesWithFruit(String fruto) {
```

Que irá receber como argumento uma *String* que irá corresponder a um fruto,

este método irá depois criar uma sublista do ficheiro *CSV* principal, apenas com "Data" que contenham o fruto inserido.

```
for (Data f : fileReaderData) {
    if (f.getItem().getItem().equals(fruto)) {


        countriesWithFruit.add(f);


    }
}
```

O ficheiro principal será percorrido e caso o fruto seja igual ao inserido, esta "Data" será adicionada à nova lista.

O segundo método desta classe:

```
public Map<String,Integer> getSuperiorQuantity(ArrayList<Data> countryList){
```

Irá receber como argumento uma lista de Data e irá retornar um mapa com uma *String*, que será o País e um *Integer* que será o número máximo de anos consecutivos que o fruto inserido no método anterior apresenta um crescimento

```
String paisAnterior = "";
int valormaior = 0;
int count = 0;
int countMaior = 0;
Map<String, Integer> CountriesWithFruit = new HashMap<>();
```

Primeiramente serão inicializadas várias variáveis auxiliares e o mapa que depois será retornado

```
for (Data g : countryList) {

    if(!g.getArea().getArea().equals(paisAnterior)){
        countMaior = 0;}
    if (g.getArea().getArea().equals(paisAnterior) )  {
        if (g.getValue().getValue() <= valormaior) {
            count = 0;
            valormaior = 0;

        } else
            valormaior = g.getValue().getValue();
        count++;
        if (count > countMaior) {
            countMaior = count;
            if (CountriesWithFruit.containsKey(g.getArea().getArea())) {
                CountriesWithFruit.replace(g.getArea().getArea(), countMaior);
            } else if (!CountriesWithFruit.containsKey(g.getArea().getArea())) {
                CountriesWithFruit.put(g.getArea().getArea(), countMaior);
            }
        }

    }
    paisAnterior = g.getArea().getArea();
```

Inicialmente a lista que foi passada como argumento será percorrida,

Na primeira iteração a variável “paisAnterior” será nula, o programa saltará

para a última linha e irá igual o primeiro país ao “paísAnterior”, a partir daí já será possivel entrar nos “if’s”, se o país for diferente o “countmaior” será 0, para que não haja contagens entre países diferentes, caso o país seja igual e o valor de produção do fruto for menor que o valor maior o count e o "valormaior” serão novamente inicializados a 0, caso isto não aconteça, o count é incrementado e o “valormaior” é substituido por o novo “valormaior” e colocar/trocar no mapa a contagem e o respetivo país.

Isto irá acontecer sucessivamente até que será atingido o maior número de anos em que há crescimento do fruto em questão, e este será colocado no mapa ao lado do respetivo país.

5. **Dado um determinado país P, devolver, numa estrutura adequada, o par de anos, o fruto e o maior valor absoluto da diferença de produção.**

   Classe – *CountryController*:

   Nesta classe apenas existe um método:

```
public LinkedHashMap<Data, Data> CountryMap(String country) {
```

Que irá receber como argumento uma *String* que irá corresponder ao país que o utilizador pretende receber os resultados. Este método irá depois criar um *Set* do ficheiro *CSV* principal, apenas com o País que o utilizador escolheu.

```
dataArr = new FileReaderData().readData( filename:
for (Data d : dataArr) {
    if (d.getArea().getArea().equals(country)) {
        dataChosenCountry.add(d);
    }
}
```

O CSV vai ser percorrido e se se igualar ao ao País inserido, irá percorrer o resto do código, senão vai ser devolvido o valor null.

```
if (dataChosenCountry.size() <= 0) {
    return null;
} else {
    ArrayList<Data> dataChosenCountryArr = new ArrayList<>(dataChosenCountry);
    dataChosenCountryArr.sort(new SortByYear());
    ListIterator<Data> iterator = dataChosenCountryArr.listIterator();
    Data dataIt1 = iterator.next();
    Data dataIt2 = iterator.next();
    while (iterator.hasNext()) {
        if (dataIt1.getItem().getItem().equals(dataIt2.getItem().getItem())) {
            if (dataIt2.getYear().getYear() - 1 == dataIt1.getYear().getYear()) {
                results.put(dataIt1, dataIt2);
                iterator.previous();
                iterator.previous();
                iterator.remove();
                dataIt1 = iterator.next();
                if (iterator.hasNext())
                    dataIt2 = iterator.next();
            }
        } else {
            dataIt1 = iterator.next();
            dataIt2 = iterator.next();
        }
    }

}
```

Esta última parte do método, transforma o Set em ArrayList para organizá-lo por Fruto e Ano e volta a transformar num ListIterator. São criados dois iterator's para percorrer os anos e adicionar ao Map results a Data para poder ser comparada.

```java
for( Map.Entry<Data, Data> entry : results.entrySet() ){
    //System.out.println( entry);
    int valorAbsAux = Math.abs( entry.getValue().getValue().getValue() - entry.getKey().getValue().getValue());
    if (valorAbs < valorAbsAux){
        resultsFinal.put(entry.getKey(), entry.getValue());
        valorAbs = valorAbsAux;
    }
}
```

No final, é feito um Map resultsFinal para comparar o maior valor absoluto e adicioná-lo ao Map. No Menu UI é impresso o resultado final do fruto com o maior valor absoluto.

Para melhorar:

- Como a variável "valormaior" é inicializada a 0, será sempre feita a primeira iteração, pelo que haverá sempre um "count++" o que significa que mesmo que não haja crescimento o valor do count entre esses dois anos será 1 em vez de 0. Isto acaba por não prejudicar o programa, mas é um aspeto que podia ser melhorado

Diagrama UML