RELATÓRIO DE ESTRUTURAS DE INFORMAÇÃO

Projeto 1 – *Covid-19*

Raúl Coelho 1190986

José Miguel Silva 1190778

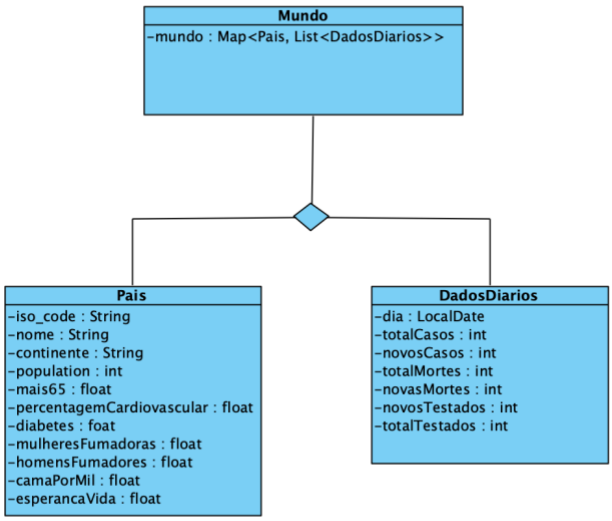
Introdução

Na disciplina de Estruturas de Informação foi-nos proposto, desenvolver uma aplicação que permita gerir a informação de vários países relacionada com a pandemia COVID-19, assim como aplicar os conhecimentos adquiridos nas aulas de Java Collections Framework para responder aos requisitos que nos foram colocados no projeto. s

Desenvolvimento/Explicação dos conteúdos e requisitos propostos

1º-Carregar e guardar informação do ficheiro Excel fornecido.

Para começar pensamos qual seria a melhor maneira de armazenar os dados fornecidos no ficheiro Excel e acabamos por armazená-los seguindo o diagrama de classes seguinte:



Para além destas, temos ainda a classe LerDados que lê os dados do ficheiro e os armazena na devida posição do Map. A estrutura deste programa foi concebida de forma a que haja uma fácil manipulação dos dados visto ser bastante simples chegar a qualquer dado pretendido.

2º-Apresentar uma lista de países ordenados por ordem crescente do número de dias para atingir os 50 000 casos positivos de COVID-19.

A maneira que arranjamos para apresentar os dados pretendidos foi previamente criar um Map<Pais, List<Integer>> para armazenar na primeira posição do List o valor de dias que demorou a atingir os 50000 casos e na segunda posição armazenar a posição do ArrayList<DadosDiarios> em que tal acontecia. Para isso verificamos na lista de dados qual o primeiro dia em que o total de casos era superior a 50000, quando isso acontecesse íamos

buscar o dia do dado em questão e subtraímos ao dia 1/1/2020 pois é o dia que queremos comparar e ainda adicionámos o valor obtido ao Map juntamente com a posição do Array que estávamos. Após este valor ser obtido para um país, avançavamos para o país seguinte do Map original. Finalmente, após todos os países serem verificados, criámos um pequeno método para onde passávamos o Map obtido, método este que o irá ordenar por ordem crescente de dias que demorou a atingir os 50000 casos.

3º-Apresentar o total de novos casos assim como novas mortes por continente/mês

Para esse requisito a maneira mais eficiente que encontrámos para resolver o problema proposto foi criar um List<Pais> e ordená-lo por ordem alfabética do continente para que pudéssemos então apresentar os dados por continente. Após isso, inicializamos uma variável mês para sabermos em qual mês do continente estamos a trabalhar. De seguida, para todos os dados do país verificamos se o mês é igual à variável definida e se for adicionamos os novos casos e as novas mortes desse dado. Seguidamente, passamos ao próximo país e verificamos se esse ainda faz parte do continente atual, se sim voltamos a fazer o procedimento anterior, caso contrário, adicionamos os dados recolhidos a um List<String> para podermos posteriormente apresentar na consola. Passamos de seguida ao próximo mês e se este não for superior a nove (último mês do ficheiro) ele passa ao mês seguinte e volta ao primeiro país para voltar a verificar dado a dado. Após passarem então os nove meses, passamos para o próximo continente e repetimos todo o processo anterior. No final apresentamos os dados no formato pedido.

4º-Devolver para cada dia de um determinado mês e para um dado continente, os países por ordem decrescente de casos positivos.

Neste caso, pedimos ao utilizador que selecione o mês e de seguida o continente. Posteriormente, verificamos quantos dias tem o mês escolhido e para cada país do continente selecionado vamos para cada dia guardar o pais e o número de novos casos que apresentam nesse dia num Map<Pais,Integer>. Após verificar o primeiro dia, vamos ordenar o Map por ordem decrescente de casos recorrendo a um Comparator, fazendo uso da classe Collections do Java. De seguida, apresentamos na consola os dados do Map para o primeiro dia e passamos para o dia seguinte repetindo todo o processo anterior.

5º-Devolver numa estrutura adequada, todos os países com mais de 70% de população de fumadores, ordenados por ordem decrescente do número de novas mortes.

Para este caso, criamos um List<Pais> onde verificamos para cada Pais se a soma das percentagens de homens fumadores e mulheres fumadoras era superior a 70, uma vez que os valores já se encontravam em percentagens. De seguida, ordenamos o conteúdo da List recorrendo mais uma vez à classe Collections do Java. Ordenação essa que foi feita por ordem decrescente do número de novas mortes dos países.

Conclusão

Com este trabalho ficamos a conhecer de forma mais aprofundada as ferramentas da Java Collections Framework, aplicando esses conhecimentos através do uso de Maps e indexação Key, Value para tornar as pesquisas mais eficientes, assim como o uso de TreeMaps, HashMaps, List e o método Collections.sort para garantir a rápida ordenação de dados.
Podemos concluir que as ferramentas da Java Collections Framework são realmente eficientes para pôr em prática porque conseguem proporcionar uma melhor organização de toda a informação.