

Relatório ED

Trabalho Prático Época Especial

Grupo 6

Célio Macedo - 8170358

João Leite - 8170556

Índice

[Índice de Figuras II](#_Toc81564335)

[Lista de Abreviaturas 1](#_Toc81564336)

[Introdução 2](#_Toc81564337)

[Classes usadas no desenvolvimento do Projeto 3](#_Toc81564338)

[Conclusão 5](#_Toc81564339)

# Índice de Figuras

[Figura 1 - Menu principal do nosso programa. 4](https://ipppt-my.sharepoint.com/personal/8170556_estg_ipp_pt/Documents/Relatorio_Grupo_6.docx#_Toc81564331)

# Lista de Abreviaturas

JSON : JavaScript Object Notation

# Introdução

Este projeto foi no proposto pela unidade curricular de Estrutura de Dados em que consiste implementar uma solução para um problema pedido. A solução necessária para resolver o problema consiste em desenvolver um programa utilizando todos os conhecimentos adquiridos durante o leccionamento da unidade curricular, usando a estrutura de dados que mais se adeque a nossas necessidades conforme o que necessário. Em resumo, implementamos um sistema de monitorização de movimentos de pessoas (funcionários e hospedes) num hotel, com o nome de “Toca”, sendo que temos que possibilitar o movimento de pessoas dentro do hotel e guardar em ficheiros esses movimentos.

# Classes usadas no desenvolvimento do Projeto

Este projeto vai ser constituído pelas seguintes classes:

* Divisão;
* Pessoa;
* Movimentos;
* Hotel;
* Gestão Hotel;
* JSONMovimentos;
* Menu;
* Export;
* Import;
* Grafo Hotel.

A **Divisão** corresponde a classe que vai possibilitar gerar um objeto Divisão que vai conter como “características” as seguintes instâncias:

nome : que vai corresponder ao nome da Divisão e é do tipo String;

tipoSala: corresponde a uma enumeração que pode ter os valores Quarentena ou Reservada, sendo que a Divisão do tipo Quarentena pode ser acedida por todas as pessoas do Hotel, as divisões Reservadas só podem ser acedidas pelos funcionários;

capacidadeMaxima : corresponde a capacidade máxima que a divisão pode ter de pessoas no seu interior;

listaDePessoas: corresponde a uma DoubleLinked List desordenada que vai guardar todos as pessoas que entrarem na Divisão.

As classes **Pessoa** contêm as seguintes instâncias:

Id: que corresponde ao id de cada pessoa, sempre que for instanciada uma pessoa, é incrementado uma unidade a ele;

tipo: corresponde a uma enumeração que contêm dois valores, Funcionário e Hospede, que ajuda a identificar as permissões que essas pessoas vão poder ter dentro do Hotel;

A classe **Movimentos** contêm como instâncias:

nomeDivisao : vai identificar o nome da Divisão, para onde a pessoa se deslocou;

idPessoa: corresponde ao ID da pessoa;

dataHoraAtual: vai retornar a hora atual do sistema, que vai representar a cada horário ele entrou na divisão.

A Classe **Hotel** onde vai ser estruturado maior parte do problema proposto. É nesta classe que vamos preencher o grafo, já agora, o grafo que optamos por utilizar, vai ser um grafo pesado não direcionado, vamos ler os mapas que nos foram fornecidos pelos docentes, lendo um ficheiro com a estrutura json, através da biblioteca json-simple1.1.1, e preenchendo, como já citado anteriormente, o grafo com as divisões presentes no ficheiro json, depois de adicionar-mos as divisões ao grafo, vamos ler as ligações que estão descritas no ficheiro json, colocando o peso dessas ligações a 1, de assinalar que o peso das arrestas das divisões vai ser a soma do número de pessoas atualmente dentro da divisão, ou seja, por exemplo, existe uma ligação entre o Hall de Entrada e o Elevador, supondo que o número de pessoas no Elevador atualmente é 2 e na Hall de Entrada são 5, logo o peso da deslocação entre esse duas divisões a soma das pessoas entre as divisões. Na classe Hotel vamos uma coleção que vão guardar todos os Movimentos realizados no Hotel, sendo que essa mesma coleção também vai ser exportada para um ficheiro json.

A classe **Gestão Hotel** vai ser onde vai ser feito os métodos para fazer o deslocamento das pessoas dentro do Hotel, tendo dois métodos principais, um que será o modo manual que irá pedir ao utilizador mover uma pessoa a sua escolha para uma divisão, temos também um método que irá permitir saber qual o caminho mais curto da atual divisão de uma pessoa até a sala de quarentena, de alertar que, o caminho mais curto significa que a pessoa tem de passar por o menor número de pessoas dentro das divisões por onde ele passar. Nesta classe vamos conter métodos para adicionar pessoas dentro do hotel, nós como entrada e saída do hotel, definimos que o Hall de Entrada vai ser entrada/saída.

A classe **JSONMovimentos** é classe que vai permitir criar objetos que posteriormente, com o formato deles mesmo vão ser exportados para um ficheiro json, estes objetos vão conter três instâncias principais, que são: nomeHotel, versão, movimentos. Os movimentos do hotel vão ser guardados dentro de uma estrutura, a classe JSONMovimentos vai implementar a interface Comparable, que vai fornecer o método compareTo que irá permitir dar aos objetos informação em que posição vão ficar dentro da coleção em relação aos restantes. Por lapso, não foi informado no ponto onde é citada a informação do Movimento, mas está classe também implementa a interface Comparable.

A classe **Menu**, é a classe que irá permitir ao utilizador mostrar todas as opções para acederem as funcionalidades do programa, como podemos ver como está exemplificado na imagem abaixo.

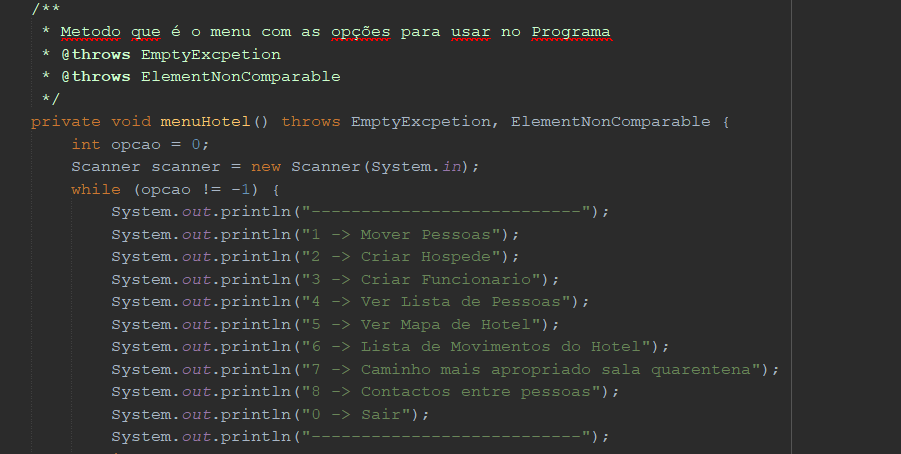


Figura - Menu principal do nosso programa.

A classe **Export** irá possibilitar fazer o export para ficheiro json da informação dos JSONMovimentos, ou seja, sempre que finalizarmos a execução do programa ele irá exportar a informação que está contida na coleção, onde guardamos os JSONMovimentos, para um ficheiro do tipo json.

A classe **Import** é classe que irá permitir fazer a importação da informação dos JSONMovimentos presentes no ficheiro json, sempre o que o programa iniciar ele irá fazer o importação da informação do ficheiro para dentro de uma coleção, se o ficheiro não existir, o programa irá criar um ficheiro novo, com o nome de movimentos.json.

E por fim, como classe do nosso programa vamos ter a classe **GrafoHotel**, é uma classe que faz “extends” a classe do Grafo Pesado, e como características de diferentes da superclasse, é que a matriz de adjacência em vez de ser preenchida com true/false vai ser com os pesos das arrestas, vamos ter um método que vai imprimir os vértices adjacentes de determinado vértice e um método que irá retornar todas os vértices inseridos dentro do grafo.

A nível de classe são estas que na nossa análise do problema são essenciais para a resolução do problema, que nos vão permitir fazer todas as operações para conseguirmos chegar aos objetivos que nos foram propostos com a realização deste projeto.

# Conclusão

Durante o desenvolvimento deste projeto tivemos algumas dificuldades, mais a nível de interpretação, mas com a ajuda e auxílio dos docentes essas mesmas dúvidas foram dissipadas e ultrapassadas. A nível de funcionalidades conseguimos cumprir com sucesso, no nosso entender, todas as tarefas que estavam descritas no enunciado. A nível de estruturas para armazenarmos a informação guardamos quase todas as informações em Double Linked Lists e Grafo, a primeira é usada porque utilizando arrays ia haver mais desperdício de memoria, usando as Linked List vamos ter uma melhor gestão da memoria porque a capacidade da lista vai ser conforme os elementos que lá forem inseridos ou removidos, mas por outro lado podíamos usar arrays porque os dados que vamos utilizar não iam acabar por usar muita memoria mas na eventualidade de cada vez mais usássemos mais informação mais memoria ia ser desperdiçada em vão. A fim de cada execução vai ser guardado num ficheiro json todos os movimentos realizados durante a execução do programa, o nome do Hotel e a sua versão passando para um objeto JSONMovimentos e depois exportado para o ficheiro supramencionado, vamos ter diferentes mapas, ou seja, vamos ter mais de que uma versão do Hotel, em que variam algumas divisões e a sua capacidade máxima e as divisões nas quais elas se conectam.