ESCOLA POLITECNICA

CURSO BACHARELADO EM ENGENHARIA DE SOFTWARE

GUSTAVO PEREIRA DE MELO

**Entrega:** Mapa dos Fenícios

Porto Alegre

2023

**Mapa dos Fenícios**

O programa consiste em fazer uma menor rota entre pontos chamados de Portos no menor caminho possível, desviando de asteriscos onde são pontos que não pode ser percorrido. O programa vem a simular uma navegação de um porto inicial até diversos outros portos pelo menor caminho possível e desviando de trechos não navegáveis e portos inacessíveis e no final, voltar ao porto inicial e mostrar o quanto gastaria este caminho em cada ponto navegado.

O programa foi modelado usando a linguagem Java, fazendo muito uso da Orientação a Objeto, fazendo leitura de um arquivo de texto onde continha as informações de tamanho da matriz do mapa e o mapa em si. O programa foi separado em classes e cada uma com sua função.

* **App** é a classe principal do programa, é nela onde é chamada as outras classes e executa os métodos “navegar” e “mostrar mapa” que mostra ao usuário o caminho feito entre cada porto e o gasto de combustível.
* **CarregaMatriz** é a classe onde o usuário seleciona o mapa que deseja carregar e a classe faz a leitura do arquivo de texto através do método “leArquivo” e monta o grafo na memória interna do programa, fechando o arquivo.
* **Grafo** é a classe onde é montada o grafo, ele é armazenado em um vetor contiguo os vértices. Apesar disso, cada vértice tem guardada dentro de si, o valor do seu índice, o caractere e sua posição em linha e coluna, tornando assim o acesso aos vértices vizinhos de forma otimizada.
* **CarregaArestas** é a classe responsável por fazer a ligação de cada vértice através do seu método ligaArestas e verificar se ele é ou não navegável, verificando o caractere, caso seja asterisco, não conecta a aresta e através de operação matemática, verifica o vizinho de baixo para melhor otimização de tempo.
* **tiposVizinhos** é classe de enums que define se o vizinho do vértice a ser verificado é o da esquerda ou o de baixo.
* **FilaPrioridadeMinima** é a estrutura de dados necessária para utilizar a busca em largura usada para fazer o caminhamento.
* **Vertice** é a classe que armazena os dados de cada vértice como índice no vetor principal, posição em coluna, posição em linha e o caractere que representa este Vertice.
* **CaminhoMinimo** é a classe onde foi feito o caminhamento, armazenando através da lista de adjacência o caminho feito entre os vértices e utilizando a fila como suporte para ir desempilhando até chegar no caminho mínimo.
* **Temporizador** é a classe responsável por fazer o calculo de tempo de execução do programa, o método initClock é onde inicializa a contagem e o método getClockSec é onde retorna o tempo passado de execução.