RELATÓRIO

Trabalho Final

DIAGRAMA DE CLASSES

Link do arquivo Astah: diagrama

O arquivo também está em anexo na pasta, com o nome “DiagramaClassesTF”.

COLEÇÕES

- **LinkedList**
  - Classe Estoque: classe catálogo para armazenamento de instâncias de Carga implementada com LinkedList.
  - Classe CadastroCarga: classe visual que extends JPanel, onde é realizado a inserção de novas Cargas na coleção da classe catálogo dela.
  - Classe AlterarSituacao: classe visual que extends JPanel, onde é consultada a coleção da classe catálogo de Carga para alterar a situação dos objetos.
  - Classe ConsultarCarga: classe visual que extends JPanel, onde é consultada a coleção da classe catálogo de Carga para impressão dos objetos e seus respectivos atributos.
  - Classe FretarCarga: classe visual que extends JPanel, onde é consultada a coleção da classe catálogo de Carga para alteração dos objetos, nos atributos: situação e prioridade. E para alocar em um Navio, uma instância de Carga.
  - Classe DadosIniciais: classe de deserialização de arquivo CSV para objetos em Java. Coleção é utilizada para inserção de novas instâncias de Carga na classe catálogo dela, após a deserialização dos arquivos.

Autores
Bianca Alves, Nicolas Miszczuk e Sofia Sartori

- Classe CarregarDados: classe de deserialização de arquivo JSON para objetos em Java. Coleção é utilizada para inserção de novas instâncias de Carga na classe catálogo dela, após a deserialização dos arquivos.

- Classe SalvarDados: classe de serialização de objetos Java para arquivo JSON. Coleção é utilizada para consulta de instâncias de Carga, para realizar a serialização.

- A classe Clientela representa uma coleção de objetos do tipo Cliente. Essa coleção é implementada usando uma LinkedList, que permite armazenar e manipular os clientes de forma dinâmica. A classe inclui uma classe interna OrderClientes, que implementa a interface Comparator para permitir a ordenação da lista de clientes com base em seus códigos.

- **ArrayList**
  - A classe Portos é uma coleção de objetos da classe Porto. Ela possui uma lista dinâmica (ArrayList) de objetos Porto, onde é possível adicionar, consultar e ordenar os portos. A classe também inclui uma classe interna OrderPortos, que implementa a interface Comparator para permitir a ordenação da lista de portos com base em seus identificadores..
  - A classe Tipos: representa uma coleção de objetos do tipo TipoCarga, onde TipoCarga pode ser uma subclasse de Perecivel ou Duravel. A coleção é implementada usando um ArrayList, que permite armazenar e manipular dinamicamente os tipos de carga.
- **Queue**
  - A classe FilaEstoque representa uma fila de objetos do tipo Carga. Essa fila é implementada utilizando a classe Queue, mais especificamente a implementação LinkedList, o que permite que os elementos sejam adicionados ao final da fila e removidos do início. A classe possui métodos para acessar a fila, obter e remover a primeira carga da fila, adicionar uma nova carga à fila e consultar se uma carga com um determinado identificador já está presente na fila.

- **Map**
    - Classe Porto: classe do objeto porto, o Map é usado para armazenar as distâncias entre cada porto, salvando a id do porto como chave e a distancia entre até esse porto como valor.

- **SortedMap**
    - Classe CadastroNavios: classe visual que extends JPanel, onde é salvo e armazenado os navios para armazenamento dos objetos e seus respectivos atributos.

LEITURA/ESCRITA DE ARQUIVOS TEXTO

O meio de escrita e leitura dos objetos, se dá meio da utilização da biblioteca Gson desenvolvida pela Google. O arquivo .jar da biblioteca foi adicionado na estrutura do projeto, de modo que seja possível o reconhecimento de seus import's.

No processo de escrita, utilizando a biblioteca conseguimos converter objetos Java para sua representação em JSON. Já no processo de leitura, convertemos as String JSON, resultantes da serialização, em objetos Java novamente.

```
Gson gson = new GsonBuilder().setPrettyPrinting().create(); // Classe de lib externa

// Converte a lista de objetos em String JSON e escreve no arquivo
FileWriter writerDuravel = new FileWriter( fileName: fileName + "-TIPOSDURAVEL"+".json");
gson.toJson(duraveis, writerDuravel);
writerDuravel.close();
```

Figura 1: Print de tela em que ocorre a escrita de uma coleção de objetos Duravel. Após a escrita, os objetos são gravados em um arquivo de extensão .json

```
Gson gson = new Gson(); // Lib externa

FileReader reader1 = new FileReader( fileName: fileName + "-CLIENTES.json");
// Configura as propriedades da coleção a ser lida, como de tipo Cliente
Type listaClientes = new TypeToken<ArrayList<Cliente>>(){}.getType();
// Converte o arquivo de String JSON em objetos Java do tipo Cliente, e grava na coleção
ArrayList<Cliente> clienteArrayList = gson.fromJson(reader1, listaClientes);
for(Cliente c : clienteArrayList) {
    clientes.adicionarCliente(c); // Retorna os objetos recuperados para coleção original
}
reader1.close();
```

Autores
Bianca Alves, Nicolas Miszczuk e Sofia Sartori

Figura 2: Print de tela em que ocorre leitura de um arquivo de String's JSON em objetos Java do tipo Cliente. Como no arquivo foi gravado uma coleção de Clientes, ele nos retorna de volta uma coleção de Clientes

Fonte utilizada para uso da biblioteca: Info Gson

Autores
Bianca Alves, Nicolas Miszczuk e Sofia Sartori