Trabalho Final – Programação Orientada a Objetos

Carolina Michel Ferreira

Mateus Campos Caçabuena

Nícholas Spolti de Souza

Introdução

Neste trabalho nos foi apresentado um desafio de criar um sistema de transportes espaciais. Assim surgiu a ACMESpace que é nossa plataforma designada a esses transportes. Nesse sistema aplicamos a maioria dos conteúdos vistos em aula, para sua criação.

A plataforma conta com diversos métodos e recursos, tanto para criar objetos, quanto para salvar dados gerais.

Métodos e TADs

Nesse trabalho foram utilizadas as coleções de ArrayList, Queue e LinkedList. Todas foram inseridas nos métodos de ações do sistema.

O ArrayList foi usado para guardar objetos das classes e realizar métodos que possuem pesquisas, assim eles percorrem o array para alterar/consultar o objeto requerido. Além disso, foi também utilizado para a criação de novos objetos, pelo usuário ou por arquivos.

Métodos de cadastro – classe Frota:

```
public boolean cadastraEspPorto (EspacoPorto espacoPorto) {
    for (EspacoPorto espPortoCad : cadastroEspPorto) {
        if(espPortoCad.getNumero() == espacoPorto.getNumero()) {
            return false;
        }
    }
    cadastroEspPorto.add(espacoPorto);
    return true;
}

public boolean cadastraEspNave (Espaconave espaconave) {
    for (Espaconave espaconaveCad : cadastroEspNave) {
        if (espaconaveCad.getNome().equals(espaconave.getNome())) {
            return false;
        }
    }
    cadastroEspNave.add(espaconave);
    return true;
}

public boolean cadastraTransp(Transporte transporte) {
    for (Transporte transporteCad : cadastroTransp) {
        if(transporteCad.getIdentificador() == transporte.getIdentificador()) {
            return false;
        }
    }
    cadastroTransp.add(transporte);
    return true;
```

Métodos de pesquisa – classe Frota:

```
public EspacoPorto pesquEspacoPorto (int numero) {
    for (EspacoPorto espPortoCad : cadastroEspPorto) {
        if (espPortoCad.getNumero() == numero) {
            return espPortoCad;
        }
    }
    return null;
}

public Espaconave pesquEspacoNave(String nome) {
    for (Espaconave espNaveCad : cadastroEspNave) {
        if (espNaveCad.getNome().equals(nome)) {
            return espNaveCad;
        }
    }
    return null;
 }

public Transporte pesquTransporte(int identificador) {
    for(Transporte transpCad : cadastroTransp) {
        if(transpCad.getIdentificador() == identificador) {
            return transpCad;
        }
    }
    return null;
}
```

Métodos de alteração/consulta de dados - classe Frota:

```
public ArrayList<EspacoPorto> mostraPortos() {
    return cadastroEspPorto;
}

public ArrayList<Espaconave> mostraEspaconaves() {
    if (cadastroEspNave == null) {
        return null;
    }
    return cadastroEspNave;
}
```

```java
public ArrayList<Espaconave> espacoDisponiveis() {
    ArrayList<Espaconave> espaconaves = mostraEspaconaves();
    ArrayList<Espaconave> navesDisponiveis = new ArrayList<>();
    for (Espaconave espacoCad : espaconaves) {
        if (espacoCad.getTransporte() == null) {
            navesDisponiveis.add(espacoCad);
        }
    }
    return navesDisponiveis;
}
```

A Queue (fila) foi utilizada principalmente para os transportes, já que é necessária uma ordem em que eles serão designados. Então, a Queue foi implementada no método de criação dos transportes, na lista de transportes e na designação dos transportes.

Método de criação – classe Frota:

```java
public boolean cadastraTransp(Transporte transporte) {
    for (Transporte transporteCad : cadastroTransp) {
        if(transporteCad.getIdentificador() == transporte.getIdentificador()) {
            return false;
        }
    }
    cadastroTransp.add(transporte);
    return true;
```

Método de listagem dos transportes – classe Frota:

```java
public Queue<Transporte> mostraTransportes(){
 if (cadastroTransp == null) {
     return null;
     }
     return cadastroTransp;
 }
```

```java
public Queue<Transporte> mostraTransportesNP() {
    if (transpNaoPendentes == null) {
        return null;
    }
    return transpNaoPendentes;
}
```

A LinkedList foi utilizada para a criação das Queues de transportes.

Criação das Queue – classe Frota:

```java
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.Queue;

public class Frota {
    private ArrayList<EspacoPorto> cadastroEspPorto;

    private ArrayList<Espaconave> cadastroEspNave;

    private Queue<Transporte> cadastroTransp;

    private Queue<Transporte> transpNaoPendentes;

    public Frota() {
        cadastroEspPorto = new ArrayList<>();
        cadastroEspNave = new ArrayList<>();
        cadastroTransp = new LinkedList<>();
        transpNaoPendentes = new LinkedList<>();
    }
```

Salvamento e cadastro de arquivos:

Para salvar e cadastrar arquivos foram utilizados os recursos de FileReader e FileWriter. Aplicados nos métodos de instância.

```java
public void cadastraArquivo() {
    boolean ok;

    System.out.println(x: "O que voce deseja adicionar (1-Espaconave, 2-Espaco-Porto, 3-Transporte)?");
    int escolha = 0;
    do{
        ok=true;
        try{
            escolha = in.nextInt();
        } catch (InputMismatchException e3) {
            in.nextLine();
            ok = false;
            System.out.println(x: "Tipo incorreto. Redigite");
        } catch (Exception e4) {
            ok = false;
            e4.printStackTrace();
            System.out.println(x: "Redigite.");
        }
    } while(!ok);
    in.nextLine();

    if (escolha == 1) {
        System.out.println(x: "Insira o caminho relativo do arquivo: ");
        String nomeArquivo = in.nextLine();
        leArquivoEspacoNave(nomeArquivo);
        return;
    }
    if (escolha == 2) {
        System.out.println(x: "Informe o caminho relativo do arquivo:");
        String nomeArquivo = in.nextLine();
        leArquivoEspacoPorto(nomeArquivo);
        return;
    }
    if (escolha == 3) {
        System.out.println(x: "Insira o caminho relativo do arquivo:");
        String nomeArquivo = in.nextLine();
        leArquivoTransporte(nomeArquivo);
        return;
    }
```

```
    if (escolha != 1 && escolha != 2 && escolha != 3) {
        System.out.println(x: "Opção Inválida.");
        return;
    }
}

public void leArquivoEspacoNave(String nomeArquivo) {
    Path path = Paths.get(nomeArquivo + ".dat");
    try {
        BufferedReader br = Files.newBufferedReader(path, Charset.defaultCharset());
        String line = null;
        int tipo;
        String nome;
        int espacoPorto;
        double velMaxWarp;
        double velMaxImpulso;
        int qtdMaXPessoasOuCarga;
        String comb;
        while ((line = br.readLine()) != null) {
            String[] lines = line.split(regex: ";");
            tipo = Integer.parseInt(lines[0]);
            nome = lines[1];
            espacoPorto = Integer.parseInt(lines[2]);
            if (tipo == 1) {
                velMaxImpulso = Double.parseDouble(lines[3]);
                comb = lines[4];
                Espaconave espaconave = new Subluz(tipo, nome, espacoPorto, velMaxImpulso, comb, transporte: null);
                frota.cadastraEspNave(espaconave);
                System.out.println(espaconave.toString());
                System.out.println(x: "Espaconave cadastrada!");
            }
            else if (tipo == 2) {
                velMaxWarp = Double.parseDouble(lines[3]);
                qtdMaXPessoasOuCarga = Integer.parseInt(lines[4]);
                Espaconave espaconave = new FTL(tipo, nome, espacoPorto, velMaxWarp, qtdMaXPessoasOuCarga, transporte: null);
                frota.cadastraEspNave(espaconave);
                System.out.println(espaconave.toString());
                System.out.println(x: "Espaconave cadastrada!");

                }
            }
        } catch (FileNotFoundException e) {
            System.out.println(e);
        } catch (IOException e) {
            System.out.println(e);
        }
    }

    public void leArquivoEspacoPorto(String nomeArquivo) {
        Path path = Paths.get(nomeArquivo + ".dat");
        try {
            BufferedReader br = Files.newBufferedReader(path, Charset.defaultCharset());
            String line = null;
            int numero;
            String nome;
            double coordX;
            double coordY;
            double coordZ;
            while ((line = br.readLine()) != null) {
                String[] lines = line.split(regex: ";");
                numero = Integer.parseInt(lines[0]);
                nome = lines[1];
                coordX = Double.parseDouble(lines[2]);
                coordY = Double.parseDouble(lines[3]);
                coordZ = Double.parseDouble(lines[4]);

                EspacoPorto espacoPorto = new EspacoPorto(numero, nome, coordX, coordY, coordZ);
                frota.cadastraEspPorto(espacoPorto);
                System.out.println(espacoPorto.toString());
                System.out.println(x: "Espaco-porto cadastrado!");
            }
        } catch (FileNotFoundException e) {
            System.out.println(e);
        } catch (IOException e) {
            System.out.println(e);
        }
    }
```

```java
public void leArquivoTransporte(String nomeArquivo) {
    Path path = Paths.get(nomeArquivo + ".dat");
    try {
        BufferedReader br = Files.newBufferedReader(path, Charset.defaultCharset());
        String line = null;
        int tipo;
        int identificador;
        int origem;
        int destino;
        double carga;
        int qtdMaxPessoasOuCarga;
        String descricao;

        EstadoTransporte estado;
        String espaconave;
        while ((line = br.readLine()) != null) {
            String[] lines = line.split(regex: ";");
            tipo = Integer.parseInt(lines[0]);
            identificador = Integer.parseInt(lines[1]);
            origem = Integer.parseInt(lines[2]);
            destino = Integer.parseInt(lines[3]);
            if (tipo == 1) {
                qtdMaxPessoasOuCarga = Integer.parseInt(lines[4]);

                Transporte transporte = new TransportePessoa(tipo, identificador, frota.pesquEspacoPorto(origem),
                        frota.pesquEspacoPorto(destino), EstadoTransporte.PENDENTE, espaconave: null, qtdMaxPessoasOuCarga);
                frota.cadastraTransp(transporte);
                System.out.println(transporte.toString());
                System.out.println(x: "Transporte cadastrado!");
            }
            if (tipo == 2) {
                carga = Double.parseDouble(lines[4]);
                descricao = lines[5];

                Transporte transporte = new TransporteMaterial(tipo, identificador, frota.pesquEspacoPorto(origem),
                        frota.pesquEspacoPorto(destino), EstadoTransporte.PENDENTE, espaconave: null, carga, descricao);
                frota.cadastraTransp(transporte);
                System.out.println(transporte.toString());
                System.out.println(x: "Transporte cadastrado!");
            }
```

```java
            }
        } catch (FileNotFoundException e) {
            System.out.println(e);
        } catch (IOException e) {
            System.out.println(e);
        }
    }
```

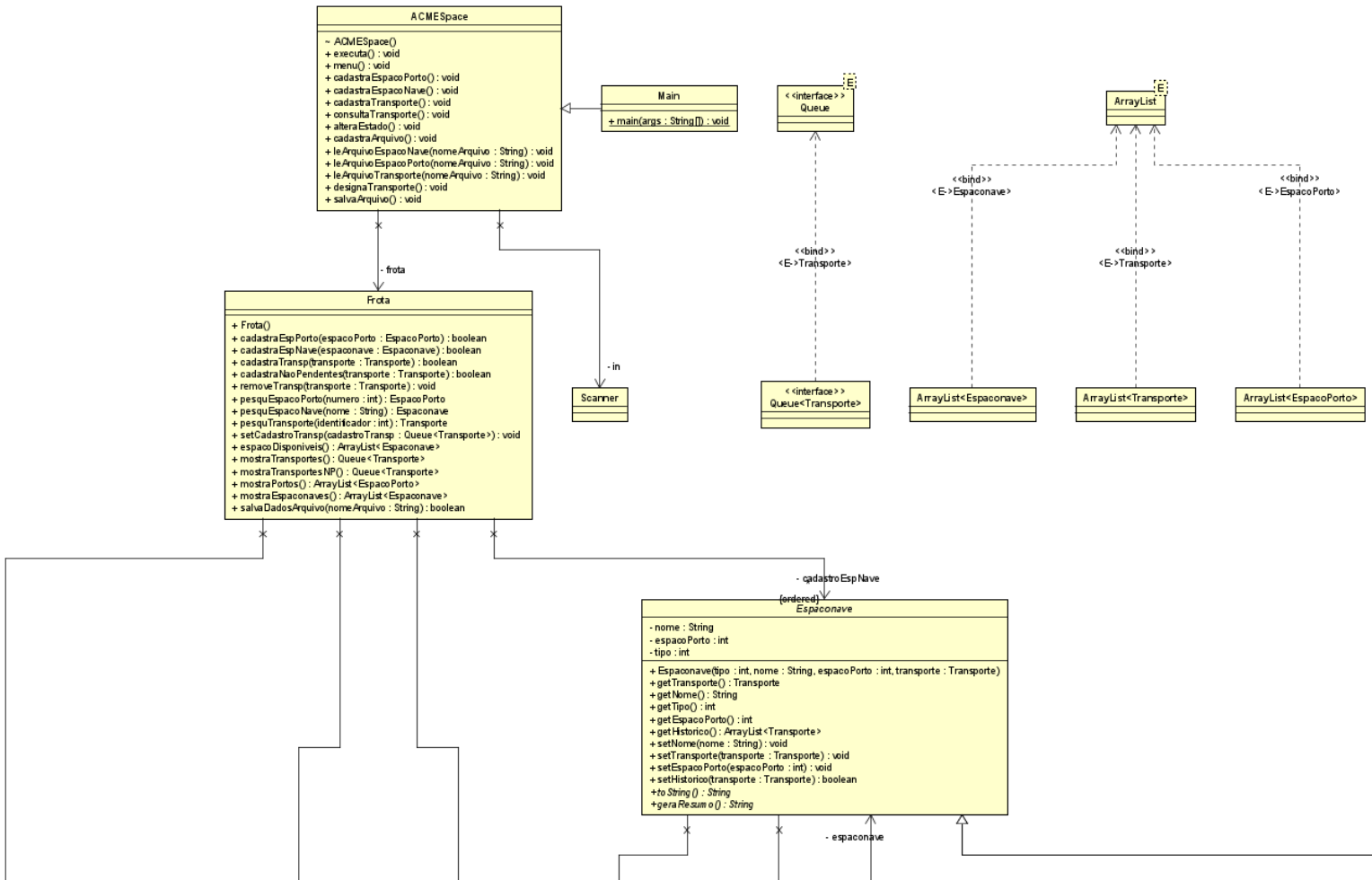
```java
    public void salvaArquivo() {
        try {
            System.out.println(x: "Informe o nome do arquivo em que os dados serao salvos:");
            String nomeArquivo = in.nextLine();
            frota.salvaDadosArquivo(nomeArquivo);
        } catch (InputMismatchException e) {
            in.nextLine();
            System.err.println(x: "Erro: entrada de dados incorreta.");
        }
    }
```

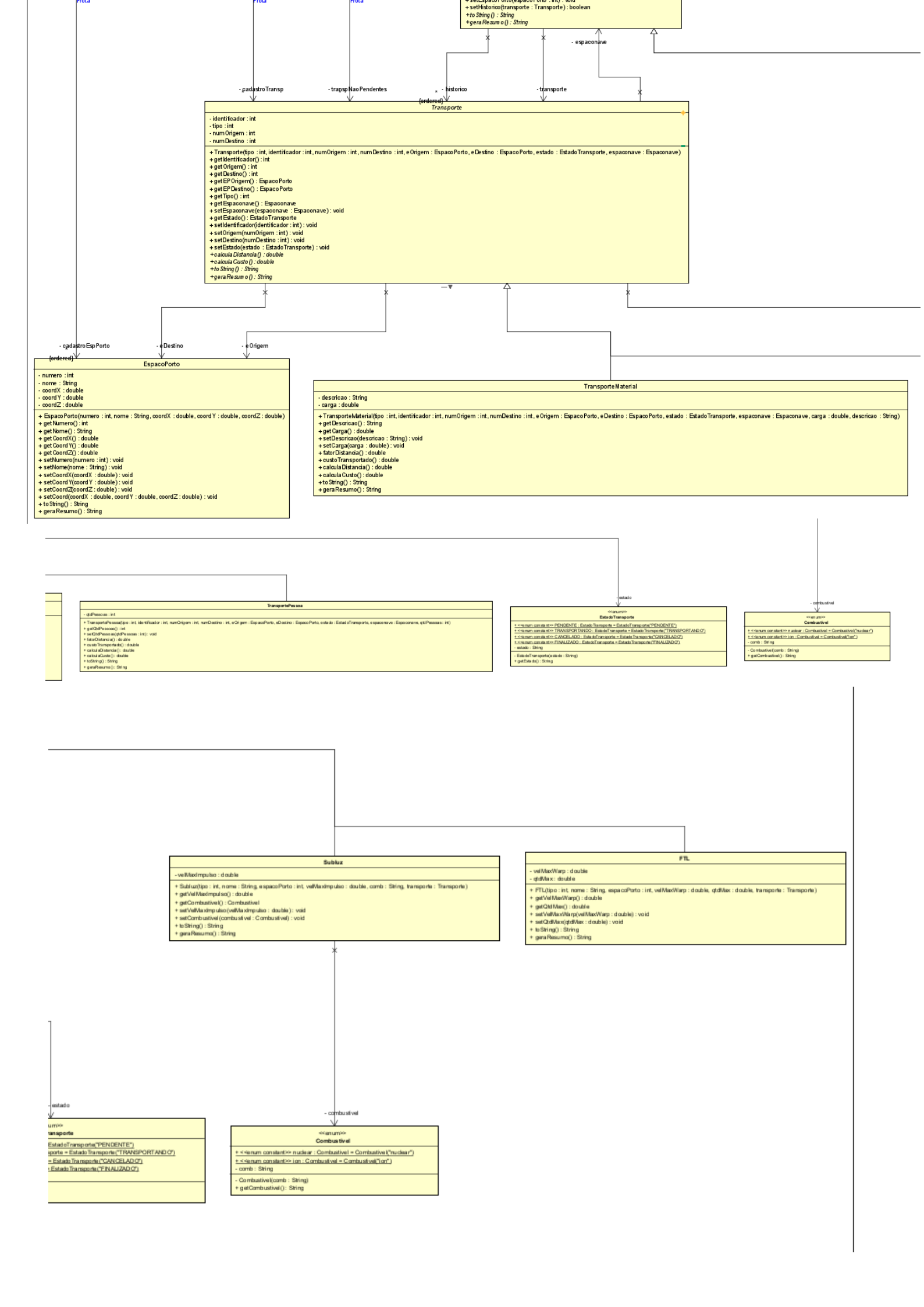
Para salvar arquivos em JSON foi utilizado a biblioteca Gson do Google, para facilitar o manuseio de dados:

```java
public void salvaJSON()  {
    try {
    System.out.println(x: "Informe o nome do arquivo:");
    String nomeArquivo = in.nextLine();
    Gson gson = new Gson();
    String jsonpor = gson.toJson(frota.mostraPortos());
    String jsonespn = gson.toJson(frota.mostraEspaconaves());
    String jsontrans = gson.toJson(frota.mostraTransportes());
    String jsonnp = gson.toJson(frota.mostraTransportesNP());

    FileWriter fw1 = new FileWriter(nomeArquivo + "(espacoportos)" + ".json");
    FileWriter fw2 = new FileWriter(nomeArquivo + "(espaconaves)" + ".json");
    FileWriter fw3 = new FileWriter(nomeArquivo + "(transportes)" + ".json");
    FileWriter fw4 = new FileWriter(nomeArquivo + "(transportesnaopendentes)" + ".json");

    //fw.write(jsonpor + jsonespn + jsontrans + jsonnp);
    fw1.write(jsonpor);
    fw1.flush();
    fw1.close();
    fw2.write(jsonespn);
    fw2.flush();
    fw2.close();
    fw3.write(jsontrans);
    fw3.flush();
    fw3.close();
    fw4.write(jsonnp);
    fw4.flush();
    fw4.close();
    } catch (IOException exception){
        System.err.println(exception);
    }
}
```




Frota
Frota
Frota
+ setHistorico(transporte : Transporte) : boolean
+to String () : String
+geraResumo () : String
- espaconave
- cadastroTransp
- trapspNaoPendentes
- historico
{ordered}
- transporte
Transporte
- identificador : int
- tipo : int
- numOrigem : int
- numDestino : int
+ Transporte(tipo : int, identificador : int, numOrigem : int, numDestino : int, eOrigem : EspacoPorto, eDestino : EspacoPorto, estado : EstadoTransporte, espaconave : Espaconave)
+ getIdentificador() : int
+ getOrigem() : int
+ getDestino() : int
+ getEPOrigem() : EspacoPorto
+ getEPDestino() : EspacoPorto
+ getTipo() : int
+ getEspaconave() : Espaconave
+ setEspaconave(espaconave : Espaconave) : void
+ getEstado() : EstadoTransporte
+ setIdentificador(identificador : int) : void
+ setOrigem(numOrigem : int) : void
+ setDestino(numDestino : int) : void
+ setEstado(estado : EstadoTransporte) : void
+calculaDistancia () : double
+calculaCusto () : double
+to String () : String
+geraResumo () : String
- cadastroEspPorto
{ordered}
- eDestino
- eOrigem
EspacoPorto
- numero : int
- nome : String
- coordX : double
- coordY : double
- coordZ : double
+ EspacoPorto(numero : int, nome : String, coordX : double, coordY : double, coordZ : double)
+ getNumero() : int
+ getNome() : String
+ getCoordX() : double
+ getCoordY() : double
+ getCoordZ() : double
+ setNumero(numero : int) : void
+ setNome(nome : String) : void
+ setCoordX(coordX : double) : void
+ setCoordY(coordY : double) : void
+ setCoordZ(coordZ : double) : void
+ setCoord(coordX : double, coordY : double, coordZ : double) : void
+ toString() : String
+ geraResumo() : String
TransporteMaterial
- descricao : String
- carga : double
+ TransporteMaterial(tipo : int, identificador : int, numOrigem : int, numDestino : int, eOrigem : EspacoPorto, eDestino : EspacoPorto, estado : EstadoTransporte, espaconave : Espaconave, carga : double, descricao : String)
+ getDescricao() : String
+ getCarga() : double
+ setDescricao(descricao : String) : void
+ setCarga(carga : double) : void
+ fatorDistancia() : double
+ custoTransportado() : double
+ calculaDistancia() : double
+ calculaCusto() : double
+ toString() : String
+ geraResumo() : String
TransportePessoa
- qtdPessoas : int
+ TransportePessoa(tipo : int, identificador : int, numOrigem : int, numDestino : int, eOrigem : EspacoPorto, eDestino : EspacoPorto, estado : EstadoTransporte, espaconave : Espaconave, qtdPessoas : int)
+ getQtdPessoas() : int
+ setQtdPessoas(qtdPessoas : int) : void
+ fatorDistancia() : double
+ custoTransportado() : double
+ calculaDistancia() : double
+ calculaCusto() : double
+ toString() : String
+ geraResumo() : String
- estado
«enum»
EstadoTransporte
+ «enum constant» PENDENTE : EstadoTransporte = EstadoTransporte("PENDENTE")
+ «enum constant» TRANSPORTANDO : EstadoTransporte = EstadoTransporte("TRANSPORTANDO")
+ «enum constant» CANCELADO : EstadoTransporte = EstadoTransporte("CANCELADO")
+ «enum constant» FINALIZADO : EstadoTransporte = EstadoTransporte("FINALIZADO")
- estado : String
- EstadoTransporte(estado : String)
+ getEstado() : String
- combustivel
«enum»
Combustivel
+ «enum constant» nuclear : Combustivel = Combustivel("nuclear")
+ «enum constant» ion : Combustivel = Combustivel("ion")
- comb : String
- Combustivel(comb : String)
+ getCombustivel() : String
Subluz
- velMaxImpulso : double
+ Subluz(tipo : int, nome : String, espacoPorto : int, velMaxImpulso : double, comb : String, transporte : Transporte)
+ getVelMaxImpulso() : double
+ getCombustivel() : Combustivel
+ setVelMaximpulso(velMaximpulso : double) : void
+ setCombustivel(combustivel : Combustivel) : void
+ toString() : String
+ geraResumo() : String
FTL
- velMaxWarp : double
- qtdMax : double
+ FTL(tipo : int, nome : String, espacoPorto : int, velMaxWarp : double, qtdMax : double, transporte : Transporte)
+ getVelMaxWarp() : double
+ getQtdMax() : double
+ setVelMaxWarp(velMaxWarp : double) : void
+ setQtdMax(qtdMax : double) : void
+ toString() : String
+ geraResumo() : String
- estado
- combustivel
«enum»
Combustivel
+ «enum constant» nuclear : Combustivel = Combustivel("nuclear")
+ «enum constant» ion : Combustivel = Combustivel("ion")
- comb : String
- Combustivel(comb : String)
+ getCombustivel() : String