Relatório Trabalho Final
Programação orientada a objetos

Nomes:

Renan Rubbo Silveira

Turma: 30

APIS UTILIZADAS

Foram utilizadas as seguintes APIs:

```
import java.io.BufferedReader;
```

```
import java.io.FileInputStream;
```

```
import java.io.InputStreamReader;
```

```
import java.text.Normalizer;
```

Para a leitura e salvamento de dados em arquivos texto.

```
import java.util.ArrayList;
```

Utilizado para a criação de uma lista de Espaçonaves e Espaço-Portos presente nos arquivos de dados.

```
import java.util.LinkedList;
```

```
import java.util.Queue;
```

Utilizado para a criação da uma fila de Transportes como foi pedido.

```
import java.io.IOException;
```

```
import java.util.InputMismatchException;
```

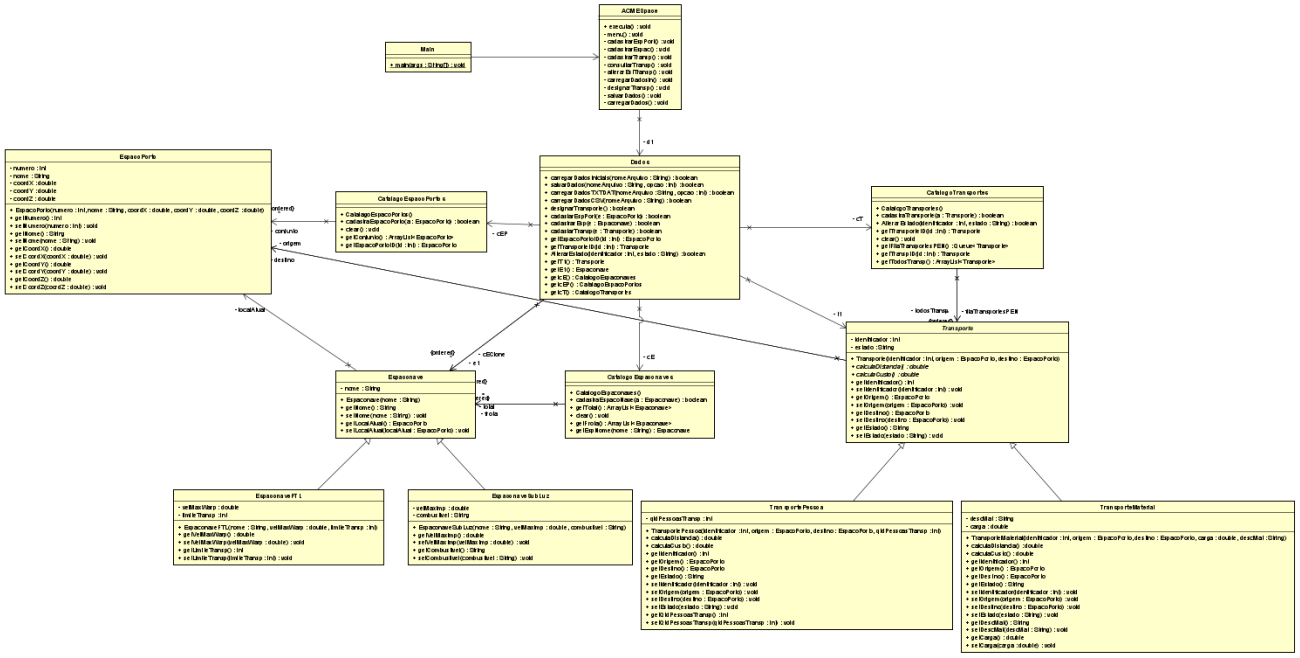
Usado para tratar algumas exceções do programa.

```
import java.util.Scanner;
```

Usado para receber a resposta do usuário.

Diagrama de Classes

Para o sistema foi gerado o seguinte diagrama de classes demonstrado abaixo.



Com o diagrama de classes possuindo muitas classes, acabou não sendo possível conter todas em uma só foto e sua visibilidade afetada, o diagrama de classes está sendo enviado para poder ser enxergado melhor todos os atributos e métodos de cada classe junto com o código e este relatório.

Coleção de dados

Na coleção de dados foram usadas as seguintes TADs:

Para criar um conjunto de Espaçonaves - ArrayList

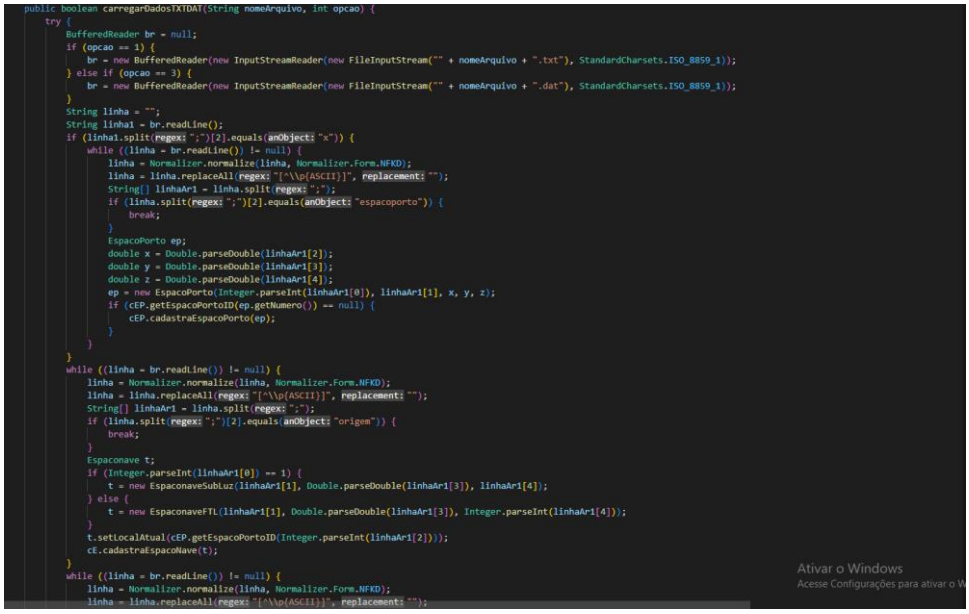
Para criar um conjunto de Espaco-Portos – ArrayList

Para criar um conjunto de Transportes - Fila, LinkedList, ArrayList

Optei por usar ArrayList na criação de conjuntos de Espaçonaves (classe CatalogoEspaconaves), na criação de uma lista contendo todos os Transportes e no conjunto de Espaços-Portos (classe CatalagoEspacoPorto), por ser uma maneira mais eficaz e fácil de adicionar objetos sem precisar limitar o tamanho. Já no conjunto de Transportes (Classe CatalogoTransportes), além de um ArrayList, utilizei uma fila para armazenar os Transportes PENDENTES, porque foi solicitado e especialmente o LinkedList por suas características de poder adicionar e remover elementos no início ou final da lista, acessar os elementos no início ou final e percorrer a lista elemento por elemento, onde são uteis nos métodos.

Armazenamento de dados

Optei por um armazenamento de dados diferente, como o programa devera ler mais de um tipo de dado (ou separado por “,” ou por “;”), criei dois métodos de leitura desses dados, um onde se o usuário pede para ler arquivos em separados por “;” (.txt ou .dat) aplica-se um método, e se o usuário pede para ler arquivos separados por “,”(.csv), ocorre outro método.

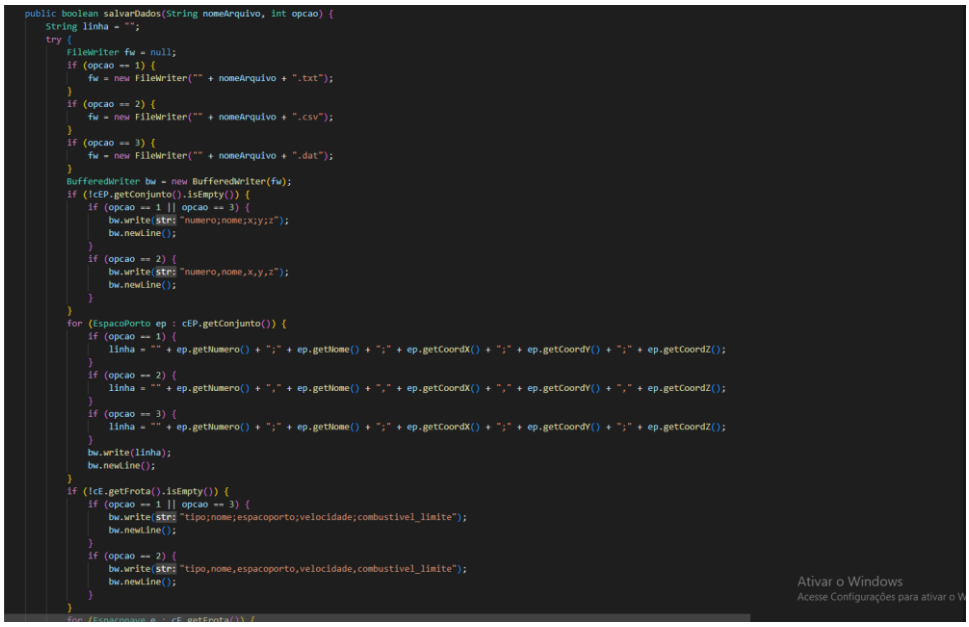


```
public boolean carregarDadosTXTDAT(String nomeArquivo, int opcao) {
    try {
        BufferedReader br = null;
        if (opcao == 1) {
            br = new BufferedReader(new InputStreamReader(new FileInputStream("" + nomeArquivo + ".txt"), StandardCharsets.ISO_8859_1));
        } else if (opcao == 3) {
            br = new BufferedReader(new InputStreamReader(new FileInputStream("" + nomeArquivo + ".dat"), StandardCharsets.ISO_8859_1));
        }
        String linha = "";
        String linha1 = br.readLine();
        if (linha1.split(regex: ";")[2].equals(anObject: "x")) {
            while ((linha = br.readLine()) != null) {
                linha = Normalizer.normalize(linha, Normalizer.Form.NFKD);
                linha = linha.replaceAll(regex: "[^\\p{ASCII}]", replacement: "");
                String[] linhaAr1 = linha.split(regex: ";");
                if (linha.split(regex: ";")[2].equals(anObject: "espacoporto")) {
                    break;
                }
                EspacoPorto ep;
                double x = Double.parseDouble(linhaAr1[2]);
                double y = Double.parseDouble(linhaAr1[3]);
                double z = Double.parseDouble(linhaAr1[4]);
                ep = new EspacoPorto(Integer.parseInt(linhaAr1[0]), linhaAr1[1], x, y, z);
                if (cEP.getEspacoPortoID(ep.getNumero()) == null) {
                    cEP.cadastraEspacoPorto(ep);
                }
            }
        }
        while ((linha = br.readLine()) != null) {
            linha = Normalizer.normalize(linha, Normalizer.Form.NFKD);
            linha = linha.replaceAll(regex: "[^\\p{ASCII}]", replacement: "");
            String[] linhaAr1 = linha.split(regex: ";");
            if (linha.split(regex: ";")[2].equals(anObject: "origem")) {
                break;
            }
            Espaconave t;
            if (Integer.parseInt(linhaAr1[0]) == 1) {
                t = new EspaconaveSubLuz(linhaAr1[1], Double.parseDouble(linhaAr1[3]), linhaAr1[4]);
            } else {
                t = new EspaconaveFTL(linhaAr1[1], Double.parseDouble(linhaAr1[3]), Integer.parseInt(linhaAr1[4]));
            }
            t.setLocalAtual(cEP.getEspacoPortoID(Integer.parseInt(linhaAr1[2])));
            cE.cadastraEspacoNave(t);
        }
        while ((linha = br.readLine()) != null) {
            linha = Normalizer.normalize(linha, Normalizer.Form.NFKD);
            linha = linha.replaceAll(regex: "[^\\p{ASCII}]", replacement: "");
```


Foto acima represente código usado para carregamento de arquivos de .txt ou .dat

Já para salvamento de dados, optei por um método onde salvaria tudo em um mesmo arquivo, todos os dados (Espaçonaves, Espaço-Porto, Transportes), onde o usuário tem a opção de escolher em qual extensão deseja salvar.

Caso o arquivo que deseja ler esteja com atributos separados por dois pontos (“ : ”), terá que ser alterado o código tendo que substituir/ fazer um replace all de todos os pontos e virgula (“;”) para dois pontos. (Como foi feito para/na apresentação feita em aula).



```
public boolean salvarDados(String nomeArquivo, int opcao) {
    String linha = "";
    try {
        FileWriter fw = null;
        if (opcao == 1) {
            fw = new FileWriter("" + nomeArquivo + ".txt");
        }
        if (opcao == 2) {
            fw = new FileWriter("" + nomeArquivo + ".csv");
        }
        if (opcao == 3) {
            fw = new FileWriter("" + nomeArquivo + ".dat");
        }
        BufferedWriter bw = new BufferedWriter(fw);
        if (!cEP.getConjunto().isEmpty()) {
            if (opcao == 1 || opcao == 3) {
                bw.write(str: "numero;nome;x;y;z");
                bw.newLine();
            }
            if (opcao == 2) {
                bw.write(str: "numero,nome,x,y,z");
                bw.newLine();
            }
        }
        for (EspacoPorto ep : cEP.getConjunto()) {
            if (opcao == 1) {
                linha = "" + ep.getNumero() + ";" + ep.getNome() + ";" + ep.getCoordX() + ";" + ep.getCoordY() + ";" + ep.getCoordZ();
            }
            if (opcao == 2) {
                linha = "" + ep.getNumero() + "," + ep.getNome() + "," + ep.getCoordX() + "," + ep.getCoordY() + "," + ep.getCoordZ();
            }
            if (opcao == 3) {
                linha = "" + ep.getNumero() + ";" + ep.getNome() + ";" + ep.getCoordX() + ";" + ep.getCoordY() + ";" + ep.getCoordZ();
            }
            bw.write(linha);
            bw.newLine();
        }
        if (!cE.getFrota().isEmpty()) {
            if (opcao == 1 || opcao == 3) {
                bw.write(str: "tipo;nome;espacoporto;velocidade;combustivel_limite");
                bw.newLine();
            }
            if (opcao == 2) {
                bw.write(str: "tipo,nome,espacoporto,velocidade,combustivel_limite");
                bw.newLine();
            }
        }
        for (Espaconave e : cE.getFrota()) {
```