

Aluno: Júlio Henrique Busarello

a) Descreveu justificativas para o desenvolvimento do algoritmo (crítico).

R: Ajudar a prefeitura juntamente a empresa de transporte a tomar uma decisão adequada quanto ao transporte público, sabendo se necessário fazer alguma alteração em uma rota, adicionando mais ônibus ou aumentando sua capacidade.

b) Incluiu o fluxograma do algoritmo no arquivo LeiaMe (crítico).

R: Link para acessar o fluxograma

<https://drive.google.com/drive/folders/168vndYBNvXJbqagMWFGZMDV7088QCK?usp=sharing>

Caso o primeiro link não funcione:

https://miro.com/welcomeonboard/d2JUY2QwaTFoOVhGUEZHV05IVHNBaU5ENIZIT2ZHSDZwYk5SWFBWR2RQQWxRYVdQODRWYUJGcUFwVVJuRHg5dnwzNDU4NzY0NTc3MjE0NzgzNTU3fDI=?share_link_id=3245195867

c) Incluiu o algoritmo no arquivo LeiaMe (crítico).

R: **ARQUIVO MAIN**

```
import java.io.*;
import java.util.*;

/**
 * @author julio_busarello
 */
public class Main {

    public static Scanner ler = new Scanner(System.in);
    public static ArrayList<Onibus> listaOnibus = new ArrayList();
    public static ArrayList<Linha> listaLinha = new ArrayList();
    public static ArrayList<Viagem> listaViagem = new ArrayList();

    public static void main(String[] args) throws IOException {
        try {
            // Recuperar todos os dados registrados anteriormente
            recuperarOnibus();
```

```
recuperarLinha();  
recuperarViagem();  
  
} catch (IOException e) {  
    // Caso não seja possível recuperar algum arquivo, informa o arquivo e o erro  
    System.err.println("Erro ao recuperar arquivos: " + e.getMessage());  
}
```

```
// Menu de opções  
  
int option;  
  
do {  
    System.out.println("===== MENU =====");  
    System.out.println("= 1- Cadastrar um Onibus   =");  
    System.out.println("= 2- Cadastrar uma Linha   =");  
    System.out.println("= 3- Cadastrar uma Viagem  =");  
    System.out.println("= 0- Sair                  =");  
    System.out.println("=====");  
    option = ler.nextInt();  
    switch (option) {  
        // Chama a função conforme a opção escolhida  
        case 1:  
            cadastrarOnibus();  
            break;  
        case 2:  
            cadastrarLinha();  
            break;  
        case 3:  
            cadastrarViagem();  
            break;  
        case 0:  
            break;
```

```

        // Caso nenhuma opção acima for utilizada
        default:
            System.err.println("Opcao invalida!");
        }
    } while (true);
}

// ----- Funções de cadastro

// Função para cadastrar um Onibus
public static void cadastrarOnibus() throws IOException {
    // Variáveis do Onibus
    String placa;
    int capMaxima;

    // Solicitando dados do Onibus
    System.out.println("Informe a placa do onibus (XXX-0000): ");
    placa = ler.next();
    System.out.println("Informe a capacidade do onibus: ");
    capMaxima = ler.nextInt();

    // Criando objeto e adicionando na lista
    Onibus onibus = new Onibus(placa, capMaxima);
    listaOnibus.add(onibus);

    // Adicionando os dados no arquivo txt
    FileWriter arquivo = new FileWriter("registroOnibus.txt", true);
    try (PrintWriter gravador = new PrintWriter(arquivo)) {
        gravador.println(onibus);
    }

    System.out.println("Onibus cadastrado com sucesso!");
}

```

```
}
```

```
// Função para cadastrar uma Linha
```

```
public static void cadastrarLinha() throws IOException {
```

```
    // Variáveis da Linha
```

```
    int nParadas;
```

```
    String terminal;
```

```
    // Pedindo as informações da Linha
```

```
    System.out.println("Informe o numero de paradas da linha: ");
```

```
    nParadas = ler.nextInt();
```

```
    terminal = ler.nextLine(); // Desbugar
```

```
    System.out.println("Informe o terminal da linha: ");
```

```
    terminal = ler.nextLine();
```

```
    // Criando objeto e adicionando na lista
```

```
    Linha linha = new Linha(nParadas, terminal);
```

```
    listaLinha.add(linha);
```

```
    // Adicionando os dados no arquivo txt
```

```
    FileWriter arquivo = new FileWriter("registroLinha.txt", true);
```

```
    try (PrintWriter gravador = new PrintWriter(arquivo)) {
```

```
        gravador.println(linha);
```

```
    }
```

```
    System.out.println("Linha cadastrada com sucesso!");
```

```
}
```

```
// Função para cadastrar Viagem
```

```
public static void cadastrarViagem() throws IOException {
```

```
    // ---- Cadastrando a viagem
```

```
// Variáveis da Viagem

String data;

String hora;

int idOnibus;

int nLinha;


// Se não houver Onibus cadastrado
if (listaOnibus.isEmpty()) {

    System.err.println("Nenhum onibus cadastrado!");

} else {

    // Escolhendo o Onibus

    System.out.println("Selecione um onibus: ");

    for (int i = 0; i < listaOnibus.size(); i++) { // Listar todos Onibus cadastrados

        System.out.println((i + 1) + ". " + listaOnibus.get(i).getPlaca());

    }

    System.out.print("ID do onibus: ");

    idOnibus = ler.nextInt();

    // Criando objeto com o Onibus selecionado

    Onibus onibusViagem = listaOnibus.get(idOnibus - 1);


// Se não houver Linha cadastrada
if (listaLinha.isEmpty()) {

    System.err.println("Nenhuma linha cadastrada!");

} else {

    // Escolhendo a Linha

    System.out.println("Selecione uma linha: ");

    for (int i = 0; i < listaLinha.size(); i++) {

        System.out.println((i + 1) + ". " + listaLinha.get(i).getTerminal());

    }

    System.out.print("Informe o numero da linha: ");

    nLinha = ler.nextInt();

}
```

```

// Criando objeto com a Linha selecionada
Linha linhaViagem = listaLinha.get(nLinha - 1);

// Solicitando outras informações da viagem
System.out.println("Informe a data da viagem: ");
data = ler.next();
System.out.println("Informe a hora da viagem: ");
hora = ler.next();

// Criando objeto Viagem
Viagem viagem = new Viagem(onibusViagem, linhaViagem, data, hora);
listaViagem.add(viagem);

System.out.println("A viagem foi cadastrada com sucesso!");

// ---- Decorrer Viagem
// Variáveis para decorrer viagem
int embarque;
int totalEmbarque = 0;

System.out.println("Decorrendo viagem...");
for (int i = 0; i < viagem.getLinha().getnParadas(); i++) { // Repetir conforme o numero
de paradas
    if (i < 1) { // Se for a primeira parada
        System.out.println("Informe quantos passageiros embarcaram: ");
        embarque = ler.nextInt();
        viagem.getOnibus().embarcar(embarque);
    } else { // Se ja passou da primeira parada
        System.out.println("Informe quantos passageiros embarcaram: ");
        embarque = ler.nextInt();
        viagem.getOnibus().embarcar(embarque);
    }
}

```

```

        System.out.println("Informe quantos passageiros desembarcaram: ");
        viagem.getOnibus().desembarcar(ler.nextInt());
    }

    // Salva o total de passageiros que passou
    totalEmbarque += embarque;
}

// Informa a conclusão da viagem e quantos passageiros passaram
System.out.println("Viagem concluída, passaram " + totalEmbarque + " passageiros na
viagem.");

// Salva os dados no arquivo txt
FileWriter arquivo = new FileWriter("registroViagem.txt", true);
try (PrintWriter gravador = new PrintWriter(arquivo)) {
    gravador.println(viagem + "," + totalEmbarque);
}

}

}

}

// ---- Funções de Recuperação
// Função para recuperar um Onibus
private static void recuperarOnibus() throws IOException {
    // Variáveis para Recuperar Onibus
    String aarq = "registroOnibus.txt";
    String linha;

    // Cria o objeto file para consultar o txt
    File arq = new File(aarq);
    if (arq.exists()) {

```

```

try {
    FileReader abrindo = new FileReader(aarq);
    try (BufferedReader leitor = new BufferedReader(abrindo)) {
        while (true) {
            linha = leitor.readLine();
            if (linha == null) {
                break;
            }
            // Cria um vetor separando as informações do txt por ","
            String[] linhaAtualOnibusArquivo = linha.split(",");

            // Cria o novo objeto pegando a coordenada do vetor e readiciona na lista
            Onibus onibus = new Onibus(linhaAtualOnibusArquivo[0],
Integer.parseInt(linhaAtualOnibusArquivo[1]));
            listaOnibus.add(onibus);
        }
    }
} catch (IOException erro) {
    // Se houve algum erro no arquivo na hora de carregar
    System.err.println("Erro ao recuperar dados do arquivo registroOnibus.txt: " +
erro.getMessage());
}

}

}

// Função para recuperar uma Linha
private static void recuperarLinha() throws IOException {
    // Variaveis para Recuperar Onibus
    String aarq = "registroLinha.txt";
    String linhaCod;

```



```

// Cria o objeto file para consultar o txt
File arq = new File(aarq);
if (arq.exists()) {
    try {
        FileReader abrindo = new FileReader(aarq);
        try (BufferedReader leitor = new BufferedReader(abrindo)) {
            while (true) {
                linhaCod = leitor.readLine();
                if (linhaCod == null) {
                    break;
                }
                // Cria um vetor separando as informações do txt por ","
                String[] linhaAtualLinhaArquivo = linhaCod.split(",");

                // Cria o novo objeto pegando a coordenada do vetor e readiciona na lista
                Linha linha = new Linha(Integer.parseInt(linhaAtualLinhaArquivo[0]),
                linhaAtualLinhaArquivo[1]);
                listaLinha.add(linha);
            }
        }
    } catch (IOException erro) {
        // Se houve algum erro no arquivo na hora de carregar
        System.err.println("Erro ao recuperar dados do arquivo registroLinha.txt: " +
        erro.getMessage());
    }
}

}

}

// Função para recuperar uma Viagem
private static void recuperarViagem() throws IOException {
    // Variaveis para Recuperar Onibus

```

```

String aarq = "registroViagem.txt";

String linhaCod;

// Cria o objeto file para consultar o txt
File arq = new File(aarq);
if (arq.exists()) {

    try {

        FileReader abrindo = new FileReader(aarq);

        try (BufferedReader leitor = new BufferedReader(abrindo)) {

            while (true) {

                linhaCod = leitor.readLine();

                if (linhaCod == null) {

                    break;

                }

                // Cria um vetor separando as informações do txt por ","
                String[] linhaAtualViagemArquivo = linhaCod.split(",");

                // Recupera a Linha e o Onibus utilizados na Viagem

                Linha recLinha = new Linha(Integer.parseInt(linhaAtualViagemArquivo[4]),
                linhaAtualViagemArquivo[5]);

                Onibus recOnibus = new Onibus(linhaAtualViagemArquivo[2],
                Integer.parseInt(linhaAtualViagemArquivo[3]));

                // Cria o novo objeto pegando a coordenada do vetor e readiciona na lista

                Viagem viagem = new Viagem(recOnibus, recLinha, linhaAtualViagemArquivo[0],
                linhaAtualViagemArquivo[1]);

                listaViagem.add(viagem);

            }

        }

    } catch (IOException erro) {

        // Se houve algum erro no arquivo na hora de carregar

```

```
        System.err.println("Erro ao recuperar dados do arquivo registroViagem.txt: " +  
erro.getMessage());  
    }  
  
    }  
    }  
}
```

ARQUIVO ONIBUS

```
/**  
 * @author julio_busarello  
 */  
public class Onibus {  
  
    private String placa;  
    private int capMaxima;  
    private int passageirosAtual = 0;  
  
    // ---- Construtor  
    public Onibus() {  
  
    }  
  
    public Onibus(String placa, int capMaxima) {  
        this.placa = placa;  
        this.capMaxima = capMaxima;  
    }  
  
    // ---- Getters and Setters  
    public String getPlaca() {  
        return placa;  
    }  
}
```

```
}
```

```
public int getCapMaxima() {  
    return capMaxima;  
}
```

```
public int getPassageirosAtual() {  
    return passageirosAtual;  
}
```

```
// ---- Métodos
```

```
public void embarcar(int passageiros) {  
    int sobra = 0;  
    if (passageiros > this.capMaxima - this.passageirosAtual) {  
        sobra = passageiros - (this.capMaxima - this.passageirosAtual);  
        this.passageirosAtual = this.capMaxima;  
        System.out.println("O onibus esta com a lotacao maxima e ficaram de fora " + sobra + "  
passageiros");  
    } else {  
        this.passageirosAtual += passageiros;  
        System.out.println("Subiram " + passageiros + " passageiros");  
        System.out.println("Agora o onibus possui " + this.passageirosAtual + " passageiros");  
    }  
}
```

```
public void desembarcar(int passageiros) {  
    if (passageiros > this.passageirosAtual) {  
        System.out.println("Desceram " + this.passageirosAtual + " passageiros");  
        this.passageirosAtual = 0;  
    } else if (this.passageirosAtual == 0) {  
        System.out.println("O onibus esta vazio");  
    }  
}
```

```

    } else {
        this.passageirosAtual -= passageiros;
        System.out.println("Desceram " + passageiros + " passageiros");
        System.out.println("Agora o onibus possui " + this.passageirosAtual + " passageiros");
    }
}

public String toString() {
    return this.placa + "," + this.capMaxima;
}

}

```

ARQUIVO LINHA

```

/**
 * @author julio_busarello
 */
public class Linha {

    private int nParadas;
    private String terminal;

    // ----- Construtor
    public Linha() {

    }

    public Linha(int nParadas, String terminal) {
        this.nParadas = nParadas;
        this.terminal = terminal;
    }
}

```

```

// ----- Getter and Setter

public int getnParadas() {

    return nParadas;

}

public String getTerminal() {

    return terminal;

}

// ----- Métodos

public String toString() {

    return this.nParadas + "," + this.terminal;

}

}

```

ARQUIVO VIAGEM

```

/**
 * @author julio_busarello
 */

public class Viagem {

    private String data;

    private String hora;

    private int passageirosAtual = 0;

    private Onibus onibus;

    private Linha linha;

    // ----- Construtor

    public Viagem() {

```

```
}
```

```
public Viagem(Onibus onibus, Linha linha, String data, String hora) {
```

```
    this.data = data;
```

```
    this.hora = hora;
```

```
    this.onibus = onibus;
```

```
    this.linha = linha;
```

```
}
```

```
// ---- Getters and Setters
```

```
public String getData() {
```

```
    return data;
```

```
}
```

```
public void setData(String data) {
```

```
    this.data = data;
```

```
}
```

```
public String getHora() {
```

```
    return hora;
```

```
}
```

```
public void setHora(String hora) {
```

```
    this.hora = hora;
```

```
}
```

```
public int getpassageirosAtual() {
```

```
    return passageirosAtual;
```

```
}
```

```
public Onibus getOnibus() {
```

```

        return onibus;
    }

    public Linha getLinha() {
        return linha;
    }

    // ---- Métodos

    public void embarcar(int passageiros) {
        this.passageirosAtual += passageiros;
    }

    public String toString() {
        return this.data + "," + this.hora + "," + this.onibus + "," + this.linha;
    }
}

FIM

```

d) Utilizou software próprio de fluxogramas para desenvolvimento do gráfico (desejável - 1º,2º)

R: Miro.

f) Descreveu no arquivo LeiaMe qual a linguagem foi utilizada no desenvolvimento do algoritmo. (desejável - 1,2º).

R: Java.

g) Descreveu no arquivo LeiaMe, qual IDE foi utilizada no desenvolvimento do algoritmo. (crítico).

R: Apache Netbeans IDE 19.

h) Descreveu no arquivo LeiaMe, infraestrutura de arquivos é necessário para funcionar o algoritmo. (crítico)

R: <https://github.com/JulioBusarello/Contador-de-Passageiros>

i) Instruiu no arquivo LeiaMe como se configura os arquivos de execução do algoritmo (crítico).

R: Instalar o repositório listado no tópico acima e abrir a pasta "Passageiros".