

Cliente: Prefeitura

Aluno: Wilian Gabriel Cardoso - Técnico em Desenvolvimento de Sistemas – T DESI
2024/1 N1

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

Descrição:

O objetivo do projeto era resolver o problema de superlotação em determinadas linhas de ônibus durante os horários de pico. A empresa de transporte necessitava de um método preciso e confiável para medir o fluxo de passageiros e justificar a adição de mais ônibus nessas rotas movimentadas. Ao desenvolver um algoritmo que calcula a quantidade de passageiros nos ônibus com base em dados coletados por sensores e câmeras, a empresa poderá tomar decisões embasadas sobre onde investir para aprimorar a capacidade e o serviço de transporte público, assegurando eficiência e a satisfação dos usuários.

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

Link do Fluxograma:

https://drive.google.com/drive/folders/1iiQ4_1YQy7aylMLiWwwAw9T4HMzQKDxK?usp=sharing

Caso não consiga ver nitidamente, segue o link no Miro:

https://miro.com/welcomeonboard/enprQ3BBZVNJWEIZdG1icHlBNzdrcUJINkVTN2VnUDRKtNkNGRPdkkzRW9ORFVJRvpLT3BydWVEc0dnMzNPM3wzNDU4NzY0NTgyNzUxNjA2NDQ5fDI=?share_link_id=745738222880

c) Incluiu o algoritmo no arquivo LeiaMe (crítico)

Main.java

```
import java.io.FileWriter;  
  
import java.io.IOException;  
  
import java.io.PrintWriter;  
  
import java.util.ArrayList;  
  
import java.util.InputMismatchException;
```

```

import java.util.Scanner;

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

    private static ArrayList<Onibus> onibusList = new ArrayList<>();//onde serão gravados
os onibus

    private static ArrayList<Linha> linhas = new ArrayList<>();//onde serão gravados as
linhas

    private static ArrayList<Viagem> viagens = new ArrayList<>(); //onde serão gravadas
as viagens

    private static Scanner ler = new Scanner(System.in); //Scanner global

    public static void main(String[] args) throws IOException {

        menu();

    }

    //Menu de cadastro

    public static void menu() throws IOException {

        boolean sair = false;

        do {

            try {

                System.out.println(" _____ Cadastre sua Viagem _____ ");

                System.out.println("|                               |");

                System.out.println("|      1 - Cadastre o Onibus      |");

                System.out.println("|      2 - Cadastre a viagem      |");

            }

        }

    }

}

```

```
System.out.println("|    3 - Cadastre a linha    |");
System.out.println("|    4 - Exibir dados      |");
System.out.println("|    5 - Decorrer Viagem   |");
System.out.println("|    6 - sair              |");
System.out.println("|_____|");
System.out.print("Entrada: ");
int op = ler.nextInt();
ler.nextLine(); // Consumir nova linha

switch (op) {
    case 1:
        cadastroOnibus();
        break;
    case 2:
        cadastroViagem();
        break;
    case 3:
        cadastroLinha();
        break;
    case 4:
        exibirDados();
        break;
    case 5:
        decorrerViagem();
        break;
    case 6:
        sair = true;
        System.out.println("Saindo...");
```

```

        break;

    default:

        System.out.println("Opcao invalida!");

    }

} catch (InputMismatchException e) {

    System.out.println("Entrada invalida. Por favor, insira um numero.");

    ler.nextLine(); // Consumir a entrada inválida

} catch (IOException e) {

    System.out.println("Erro de E/S: " + e.getMessage());

}

} while (!sair);

}

```

//Faz o cadastro dos onibus

```

public static void cadastroOnibus() {

    try {

        System.out.println("Informe a placa do onibus:");

        System.out.print("ENTRADA: ");

        String codPlaca = ler.next();

        ler.nextLine(); // Consumir nova linha

        System.out.println("Informe a capacidade maxima do onibus:");

        System.out.print("ENTRADA: ");

        int capMax = ler.nextInt();

        if (capMax <= 0) {

            System.out.println("Capacidade maxima deve ser maior que zero.");

            return;

        }

    }
}

```

```

        Onibus onibus = new Onibus(codPlaca, 0, 0, capMax);
        onibusList.add(onibus);

        System.out.println("Onibus cadastrado com sucesso!");
    } catch (InputMismatchException e) {

        System.out.println("Entrada invalida. A capacidade máxima deve ser um numero
inteiro.");

        ler.nextLine(); // Consumir a entrada inválida
    }
}

```

```

//Faz o cadastro das Viagens

public static void cadastroViagem() {

    if (onibusList.isEmpty()) {

        System.out.println("Nenhuma onibus cadastrado. Cadastre um onibus antes de
cadastrar uma viagem.");

        return;
    }
}

```

```

try {

    System.out.print("Nome da viagem:");

    String nmViagem = ler.nextLine();

    System.out.print("Data da Viagem: ");

    String data = ler.next();

    System.out.print("Hora da Viagem: ");

    String hora = ler.next();

    System.out.println("Informe a placa do onibus da viagem:");
}

```

```

        System.out.print("ENTRADA: ");

        String codPlaca = ler.next();

        Onibus onibus = buscarOnibus(codPlaca);

        if (onibus == null) {

            System.out.println("Onibus nao encontrado!");

            return;

        }

        Viagem viagem = new Viagem(data, hora, nmViagem, codPlaca);

        viagens.add(viagem);

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

    } catch (InputMismatchException e) {

        System.out.println("Entrada inválida. Tente novamente.");

        ler.nextLine(); // Consumir a entrada inválida

    }

}

//Busca os onibus que foram cadastrados
private static Onibus buscarOnibus(String codPlaca) {

    for (Onibus onibus : onibusList) {

        if (onibus.getCodPlaca().equalsIgnoreCase(codPlaca)) {

            return onibus;

        }

    }

    return null;

}

```

```
//Busca as linhas cadastradas
```

```
private static Linha buscarLinha(String nmLinha) {  
    for (Linha linha : linhas) {  
        if (linha.getNmLinha().equalsIgnoreCase(nmLinha)) {  
            return linha;  
        }  
    }  
    return null;  
}
```

```
//Faz o cadastro das linhas
```

```
public static void cadastroLinha() {  
    if (viagens.isEmpty()) {  
        System.out.println("Nenhuma viagem cadastrada. Cadastre uma viagem antes  
de cadastrar uma linha.");  
        return;  
    }  
}
```

```
try {  
    System.out.print("Informe o nome da viagem: ");  
    String nmViagem = ler.nextLine();  
  
    Viagem viagem = buscarViagem(nmViagem);  
  
    if (viagem == null) {  
        System.out.println("Viagem nao encontrada!");  
        return;  
    }  
}
```

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

String nmLinha = ler.nextLine();


System.out.print("Informe a quantidade de paradas:");

int qtdParadas = ler.nextInt();


if (qtdParadas <= 0) {

    System.out.println("A quantidade de paradas deve ser maior que zero.");

    return;

}


Linha linha = new Linha(qtdParadas, nmLinha);

linhas.add(linha);

System.out.println("Sua linha foi cadastrada com sucesso!");

} catch (InputMismatchException e) {

    System.out.println("Entrada invalida. A quantidade de paradas deve ser um
numero inteiro.");

    ler.nextLine(); // Consumir a entrada inválida

}

}


//Busca as viagens cadastradas

private static Viagem buscarViagem(String nmViagem) {

    for (Viagem viagem : viagens) {

        if (viagem.getNmViagem().equalsIgnoreCase(nmViagem)) {

            return viagem;

        }

    }

}
```



```
}  
  
return null;  
  
}
```

```
//Faz a simulação da viagem
```

```
public static void decorrerViagem() throws IOException {  
  
    if (onibusList.isEmpty()) {  
  
        System.out.println("Nao ha nenhum onibus cadastrado. Cadastre um onibus  
antes de decorrer a viagem.");  
  
        return;  
  
    }  
  
    if (viagens.isEmpty()) {  
  
        System.out.println("Nenhuma viagem cadastrada. Cadastre uma viagem antes  
de decorrer a viagem.");  
  
        return;  
  
    }  
  
    if (linhas.isEmpty()) {  
  
        System.out.println("Nenhuma linha cadastrada. Cadastre uma linha antes de  
decorrer a viagem.");  
  
        return;  
  
    }  
  
}
```

```
System.out.print("Informe a placa do onibus da viagem a decorrer: ");
```

```
String codPlaca = ler.next();
```

```
Onibus onibus = buscarOnibus(codPlaca);
```

```
if (onibus == null) {
```

```
    System.out.println("Onibus nao encontrado!");
```

```
    return;
```

```
}
```

```
System.out.print("Informe o nome da linha da viagem: ");
```

```
ler.nextLine(); // Consumir nova linha
```

```
String nmLinha = ler.nextLine();
```

```
Linha linha = buscarLinha(nmLinha);
```

```
if (linha == null) {
```

```
    System.out.println("Linha nao encontrada!");
```

```
    return;
```

```
}
```

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

```
PrintWriter gravador = new PrintWriter(arquivo);
```

```
int totalEmbarcados = 0; // Variavel para acumular o total de passageiros  
embarcados
```

```
// Laco para cada parada
```

```
for (int parada = 1; parada <= linha.getQtdParadas(); parada++) {
```

```
    System.out.println("Parada " + parada + ":");
```

```
int qtdEmbarque = 0;
```

```
int qtdDesembarque = 0;
```

```
if (parada == 1) {
```

```
    // Solicita o embarque na primeira parada
```

```
    System.out.print("Informe o numero de passageiros embarcando: ");
```

```
qtdEmbarque = ler.nextInt();

if (qtdEmbarque > onibus.getCapMax()) {
    System.out.println("Numero de passageiros excede a capacidade maxima do
onibus!");
    gravador.close();
    return;
}

onibus.setQtdPassag(qtdEmbarque);
totalEmbarcados += qtdEmbarque; // Acumula a quantidade de embarques
} else {
    // Solicita o desembarque
    System.out.print("Informe o numero de passageiros desembarcando: ");
    qtdDesembarque = ler.nextInt();

    if (qtdDesembarque > onibus.getQtdPassag()) {
        System.out.println("Numero de passageiros para desembarque excede o
numero de passageiros a bordo!");
        gravador.close();
        return;
    }

    onibus.setQtdPassag(onibus.getQtdPassag() - qtdDesembarque);

    // Solicita o embarque para as demais paradas
    System.out.print("Informe o numero de passageiros embarcando: ");
    qtdEmbarque = ler.nextInt();
```

```

        if (qtdEmbarque + onibus.getQtdPassag() > onibus.getCapMax()) {

            System.out.println("Numero de passageiros excede a capacidade maxima do
onibus!");

            gravador.close();

            return;

        }

        onibus.setQtdPassag(onibus.getQtdPassag() + qtdEmbarque);

        totalEmbarcados += qtdEmbarque; // Acumula a quantidade de embarques

    }

    onibus.setCapAtual(onibus.getCapMax() - onibus.getQtdPassag());

    // Registro das informacoes

    gravador.println("Parada " + parada + ":");

    gravador.println("Passageiros desembarcados: " + qtdDesembarque);

    gravador.println("Passageiros embarcados: " + qtdEmbarque);

    gravador.println("Passageiros restantes: " + onibus.getQtdPassag());

    gravador.println("=====");

}

gravador.println("Fim da viagem.");

gravador.close();

System.out.println("Viagem registrada com sucesso!");

System.out.println("Total de passageiros embarcados: " + totalEmbarcados); //
Exibe a soma de todos os passageiros que subiram

```

```
}
```

```
//Exibe os dados no console
```

```
private static void exibirDados() throws IOException {
```

```
    // Exibe os dados cadastrados no console
```

```
    System.out.println("Linhas:");
```

```
    for (Linha linha : linhas) {
```

```
        System.out.println("Nome: " + linha.getNmLinha() + ", Paradas: " +  
linha.getQtdParadas());
```

```
        for (Viagem viagem : linha.getViagens()) {
```

```
            System.out.println(" Viagem: " + viagem.getData() + " " + viagem.getHora() + ",  
Onibus: " + viagem.getOnibus());
```

```
        }
```

```
    }
```

```
    System.out.println("\nOnibus:");
```

```
    for (Onibus onibus : onibusList) {
```

```
        System.out.println("Placa: " + onibus.getCodPlaca()
```

```
        + ", Capacidade Maxima: " + onibus.getCapMax()
```

```
        + ", Passageiros Transportados: " + onibus.getQtdPassag());
```

```
    }
```

```
    System.out.println("\nViagens:");
```

```
    for (Viagem viagem : viagens) {
```

```
        System.out.println("Nome da Viagem: " + viagem.getNmViagem()
```

```
        + ", Data: " + viagem.getData()
```

```
        + ", Hora: " + viagem.getHora()
```

```
        + ", Onibus: " + viagem.getOnibus());
```

```
    }  
    }  
}
```

Onibus.java

```
/**
```

```
*
```

```
* @author wilian_g_cardoso
```

```
*/
```

```
public class Onibus {
```

```
    private String codPlaca;
```

```
    private int capMax;
```

```
    private int capAtual;
```

```
    private int qtdPassag;
```

```
    //Construtores
```

```
    public Onibus(String codPlaca, int capAtual, int qtdPassag,int capMax) {
```

```
        this.codPlaca = codPlaca;
```

```
        this.capAtual = capAtual;
```

```
        this.qtdPassag = 0;
```

```
        this.capMax = capMax;
```

```
    }
```

```
    public Onibus() {
```

```
    }
```

//Getters and Setters

```
public String getCodPlaca() {  
    return codPlaca;  
}
```

```
public void setCodPlaca(String codPlaca) {  
    this.codPlaca = codPlaca;  
}
```

```
public int getCapMax() {  
    return capMax;  
}
```

```
public void setCapMax(int capMax) {  
    this.capMax = capMax;  
}
```

```
public int getCapAtual() {  
    return capAtual;  
}
```

```
public void setCapAtual(int capAtual) {  
    this.capAtual = capAtual;  
}
```

```
public int getQtdPassag() {  
    return qtdPassag;  
}
```

```
    public void setQtdPassag(int qtdPassag) {  
        this.qtdPassag = qtdPassag;  
    }  
}
```

Viagem.java

```
/**
```

```
*
```

```
* @author wilian_g_cardoso
```

```
*/
```

```
public class Viagem {
```

```
    private String nmViagem;
```

```
    private String onibus;
```

```
    private String data;
```

```
    private String hora;
```

```
//construtores
```

```
public Viagem(String data, String hora, String nmViagem, String onibus) {
```

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

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

```
    this.nmViagem = nmViagem;
```

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

```
}
```

```
//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 String getOnibus() {  
    return onibus;  
}
```

```
public void setOnibus(String onibus) {  
    this.onibus = onibus;  
}
```

```
public String getNmViagem() {  
    return nmViagem;  
}
```

```
    public void setNmViagem(String nmViagem) {  
        this.nmViagem = nmViagem;  
    }  
}
```

Linha.java

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

```
/**
```

```
*
```

```
* @author wilian_g_cardoso
```

```
*/
```

```
public class Linha {
```

```
    private int qtdParadas = 0;
```

```
    private String nmLinha;
```

```
    private ArrayList<Viagem> viagens;
```

```
    //construtor
```

```
    public Linha(int qtdParadas, String nmLinha) {
```

```
        this.qtdParadas = qtdParadas;
```

```
        this.nmLinha = nmLinha;
```

```
        this.viagens = new ArrayList<>();
```

```
    }
```

```
    public int getQtdParadas() {
```

```
        return qtdParadas;
```

```
}
```

```
public void setQtdParadas(int qtdParadas) {  
    this.qtdParadas = qtdParadas;  
}
```

```
public String getNmLinha() {  
    return nmLinha;  
}
```

```
public void setNmLinha(String nmLinha) {  
    this.nmLinha = nmLinha;  
}
```

```
public ArrayList<Viagem> getViagens() {  
    return viagens;  
}
```

```
public void setViagens(ArrayList<Viagem> viagens) {  
    this.viagens = viagens;  
}
```

```
}
```

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

Descrição: Software utilizado, Miro.

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

Descrição: Linguagem utilizada para o desenvolvimento do projeto foi o Java.

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

Descrição: IDE utilizada, Apache NetBeans

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

Descrição: Para o funcionamento do algoritmo em sua máquina basta acessar esse link do GitHub e fazer a instalação dos arquivos

(<https://github.com/WilianCardoso/Senai/tree/main/T%C3%A9nico%20em%20Desenvolvimento%20de%20Sistemas/L%C3%B3gica%20de%20Programa%C3%A7%C3%A3o/Java/SA%20-%20Algoritmo%20contador%20de%20passageiros/ContPassageiros>).

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

Descrição: Terminado de concluir o tópico h, apenas será necessário abrir a pasta “SA - Algoritmo contador de passageiros” em sua IDE Apache NetBeans.