RELATÓRIO TRABALHO DE LAB. PROGRAMAÇÃO ORIENTADA A OBJETOS

Nomes: Flávia Narumi Nitto e Miguel Donizeti Da Silva E Silva.

Dados técnicos

Nosso trabalho possui 7 classes, são elas: Data, Motorista, Onibus, Operacoes, Passageiro, Rota e a classe Principal.

Data

É a classe utilizada para parametrização dos elementos utilizados que possuem data e hora, onde a classe possui métodos que são utilizados em todas as outras classes, essa classe possui todos os gets e sets também, de todos os elementos definidos como datas e horas, e também seu construtor.

Motorista

É a classe utilizada para definir os atributos de um objeto do tipo motorista, e também seus respectivos getters, setters e construtores.

Onibus

É a classe que carrega todas as informações contidas em um objeto Ônibus, também possui o número da CNH do motorista responsável pelo mesmo. Assim também consta seu array de assentos, possuindo um total de 40 lugares, em uma matriz de 4x10, métodos getters e setters, e métodos para a manipulação dos assentos, entre eles: iniciar a matriz de assentos com 0, reservar um assento, mostrar assentos numerados e em forma de 0 e 1, e seus construtores.

Operacoes

Nesta classe decidimos juntar todos os métodos que poderiam vir a ser utilizados no resto das classes, como adicionar, editar e excluir passageiros, rotas, ônibus, e motoristas, tal como mostrar cada uma das opções de assentos, visualizar rotas, ônibus, motoristas e passageiros existentes. Essa classe foi criada com a finalidade de facilitar e centralizar os métodos em uma só classe em vez de sobrecarregar a principal.

Passageiro

É a classe que possui todos os atributos de um passageiro, como nome, documento de identificação, e até o seu número de assento em alguma de suas rotas, como também o código da mesma. Também possui os métodos getter e setters adequados para cada atributo.

Rota

É a classe que carrega os atributos das rotas, assim como sua origem e destino, e um ArrayList das paradas de cada rota, como também possui objetos de data para regular seus horários e datas de saída e chegada. Também possui seus métodos getters e setters adequados aos seus atributos e um método de editaParadas, para facilitar a edição de suas paradas.

Principal

É a classe que possui o public static void main, onde começa a execução declarando o scanner e um objeto da classe Operacoes. A classe é distribuída em switch cases distribuídos em um menu para interação do usuário com o sistema, possuindo mais de 10 opções diferentes entre si, e utilizando os métodos criados em outras classes, principalmente os métodos da classe Operacoes.

Resumo do funcionamento

Primeiramente o usuário precisa cadastrar motoristas, ônibus e rotas, respectivamente, simulando o início do funcionamento do sistema ao começo do dia, onde necessita-se de algumas informações que são essenciais para seu funcionamento, como as ditas. Logo após o cadastro destes itens, se abre um menu onde podem ser feitas até 14 operações distintas entre si, onde a alteração de finalizar o programa se atende ao 0.

Prints de tela úteis

Inicializando o sistema cadastrando alguns motoristas e ônibus. Em seguida inserimos até 5 rotas.

```
-----Bem vindo ao Sistema Rodoviário da Aviação Asteróide!-----

SISTEMA: Comece inserindo alguns dados!
-Insira quantos motoristas existirão: 2
------Cadastro do 1º motorista------
---Cadastro do motorista---
Número da CNH: 123456
Data de admissão (dd mm aaaa): 13 06 2000
Nome motorista: Everton

---Mostrando lista completa---
Numero CNH: 123456
Data admissão: 13/06/2000
Nome motorista: Everton

------Cadastro do 2º motorista------
---Cadastro do motorista---
Número da CNH: 654321
Data de admissão (dd mm aaaa): 15 05 2005
Nome motorista: Andre

---Mostrando lista completa---
Numero CNH: 123456
Data admissão: 13/06/2000
Nome motorista: Everton

Numero CNH: 654321
Data admissão: 15/05/2005
Nome motorista: Andre
```

```
-Insira quantos ônibus existirão: 3
------Cadastro do 1º ônibus------
---Cadastro do Onibus---
Código: 1
Modelo: Executivo
Ano de fabricação: 2018
Marca do onibus: Mercedes Benz
Quilometragem total: 550000
Número da CNH do motorista: 123456

Assentos numerados:
[ 01 ][ 02 ][ 03 ][ 04 ][ 05 ][ 06 ][ 07 ][ 08 ][ 09 ][ 10 ]
[ 11 ][ 12 ][ 13 ][ 14 ][ 15 ][ 16 ][ 17 ][ 18 ][ 19 ][ 20 ]
[ 21 ][ 22 ][ 23 ][ 24 ][ 25 ][ 26 ][ 27 ][ 28 ][ 29 ][ 30 ]
[ 31 ][ 32 ][ 33 ][ 34 ][ 35 ][ 36 ][ 37 ][ 38 ][ 39 ][ 40 ]


---Mostrando lista completa---
Código: 1
Modelo: Executivo
Ano de fabricação: 2018
Marca: Mercedes Benz
Quilometragem: 550000.0
CNH do motorista: 123456
```

```
------Cadastro da 2º rota------
---Cadastro de rota---
Codigo rota: 2
Origem: Uberaba
Destino: Sao Jose do Rio Preto
Data de saída (dd mm aaaa): 12 09 2021
Horario de saída (hh mm): 23 50
Data de chegada (dd mm aaaa): 13 09 2021
Horario de chegada (hh mm): 04 30
Preço cheio da passagem (sem paradas): 150
Insira o código do ônibus que traçará a rota: 1
Quantidade de paradas (0~3): 3
Parada 1: Barretos
Parada 2: Planura
Parada 3: Conceiçao da Alagoa

---Mostrando lista completa---
Código da rota: 1
Código do ônibus: 2
Origem: Sao Paulo
Destino: Rio de Janeiro
Data saída: 13/06/2021
Horario saída: 13:50
Data chegada: 13/06/2021
Horario chegada: 20:45
Paradas: [Bauru, Campinas]
Preço da passagem: R$192.0

Código da rota: 2
Código do ônibus: 1
Origem: Uberaba
Destino: Sao Jose do Rio Preto
Data saída: 12/09/2021
Horario saída: 23:50
Data chegada: 13/09/2021
Horario chegada: 4:30
Paradas: [Barretos, Planura, Conceiao da Alagoa]
Preço da passagem: R$105.0
```

```
-----Bem vindo ao Sistema Rodoviário da Aviação Asteróide!-----

SISTEMA: Insira uma das opções que deseja abaixo:
1- Cadastrar passageiro
2- Alterar passageiro
3- Excluir passageiro
4- Cadastrar ônibus
5- Alterar ônibus
6- Excluir ônibus
7- Cadastrar motorista
8- Alterar motorista
9- Excluir motorista
10- Cadastrar rota
11- Alterar rota
12- Excluir rota
13- Comprar passagem
14- Cancelar passagem
0- Sair do programa
Opção:
```

Agora, o menu foi liberado para realizar novos cadastros, alterações, exclusões e compras e cancelamentos de passagens.

Quando formos realizar a compra, o sistema sempre vai pedir uma autenticação do RG do usuário.

```
public void compraPassagem() {
    int codRota;
    int numAssento;
    String doc;
    System.out.printf("\nConfirme seu RG: ");
    doc = sc.nextLine();
    for(int j=0; j<passageiros.size(); j++) {
        if(passageiros.get(j).getDoc().equals(doc)) {
            System.out.println("Passageiro " + passageiros.get(j).getNome() + " autenticado!");
```

Para mostrar a data e hora formatada, fazemos as seguintes atribuições:

```
public String getData() {
    if(dia<=9 && mes<=9) {
        return (0+dia+ "/" + 0+mes + "/" + ano);
    }
    else if(dia>9 && mes<=9) {
        return (dia + "/" + 0+mes + "/" + ano);
    }
    return (dia + "/" + mes + "/" + ano);
}
```

```
public String getHorario() {
    if(hora<=9 && min<=9) {
        return (0+hora + ":" + 0+min);
    }
    else if(hora>9 && min<=9){
        return (hora + ":" + 0+min);
    }
    return (hora + ":" + min);
}
```

Demonstrando o cadastro e alteração do ArrayList de String para paradas:

```
public void cadastraParadas(int qtdParadas) {
    String parada;
    int i;
    paradas = new ArrayList<>();
    if(qtdParadas > 3 || qtdParadas < 0) {
        System.out.println("\nNúmero inválido de paradas!");
    }
    else {
        for(i=0; i<qtdParadas; i++) {
            System.out.printf("Parada " + (i+1) + ": ");
            parada = sc.nextLine();
            paradas.add(parada);
        }
    }
}

public void editaParadas(int qtdParadas) {
    String parada;
    int i;
    if(qtdParadas > 3 || qtdParadas < 0) {
        System.out.println("\nNúmero inválido de paradas!");
    }
    else {
        for(i=0; i<qtdParadas; i++) {
            System.out.printf("Insira a nova parada " + (i+1) + ": ");
            parada = sc.nextLine();
            paradas.set(i, parada);
        }
    }
}
```

Conclusão

Um dos maiores desafios para este projeto foi a execução em mais de uma pessoa, pois quando fomos juntar as classes que tínhamos feito houve certa confusão em usar alguns métodos feitos por outra pessoa. Uma coisa que facilitou bastante após sofrermos um tempo foi a extensão do Visual Studio Code “Live share”, onde descobrimos que podíamos compartilhar todos os códigos e programar ao mesmo tempo, assim houve uma certa facilitação em termos mais técnicos. Em contrapartida, é visível que a execução e produção de códigos/algoritmos como esse é quase impossível para uma pessoa só, no mercado hoje em dia, onde se exige cada vez menos tempo para produzir algo e com mais qualidade, assim é necessário saber trabalhar em equipe até “codando”. Também possui um pró, onde é mais fácil identificar erros e métodos mais rápidos/fáceis em mais de uma pessoa. Em suma, é visível que o trabalho em equipe para problemas mais complexos e extensos como este é 100% recomendável e deve cada vez mais ser estimulado, portanto se faz necessário em nossas vidas trabalhos como este.