PROJECT DESIGN: OVERVIEW

**Curso:** Programação Orientada a Objetos

Alunos: Anderson Sales Lima

Fabiano S. Ferreira

**Projeto de sistema de aluguel de carros - Rent\_A\_Car**

**1 - Liguagem:**

Utilazada a linguagem Java para desenvolvimento do projeto.

**2 - Pacotes**

**Pacote controller:**

**\*Classes**

* Endereco (POJO):

Classe Endereco, contém informações relativas ao endereço.

- Atributos:

String estado, String cidade, String bairro, String cep, String rua, int numero;

- Métodos:

Um construtor vazio e um com todos os atributos, gets e sets, equals e toString;

* Pessoa (POJO):

Classe Pessoa, contém informações relativas à uma pessoa.

- Atributos:

String nome, String cpf, String login, String senha, LocalDate dataNascimento, Endereco endereco;

- Métodos:

Um construtor vazio e um com todos os atributos, gets e sets, equals e toString;

* Funcionario(POJO):

Classe Funcionario, contém informações relativas ao funcionário, herdando os atributos da classe Pessoa.

- Atributos:

String matricula, string cargo;

- Métodos:

Um construtor vazio e um com todos os atributos de Pessoa e de Funcionario, gets e sets, equals e toString;

* Cliente(POJO):

Classe Cliente, contém informações relativas ao cliente, herdando os atributos da classe Pessoa.

- Atributos:

String cnh, String categoriaCNH;

- Métodos:

Um construtor vazio e um para todos os atributos de Pessoa e Cliente, gets e sets, equals e toString;

* Placa(POJO):

Classe Placa, contém informações relativas à placa de algum veiculo.

- Atributos:

String estado, String cidade, String NumerosELetras;

- Métodos:

Um construtor vazio e um com todos os atributos, gets e sets, equals e toString;

* Veiculo(POJO):

Classe Veiculo, contém informações relativas ao veiculo.

- Atributos:

Placa placa, boolean disponibilidade, String marca, String Modelo, Double precoPorHora;

- Métodos:

Um construtor vazio e um com todos os atributos, gets e sets, equals e toString;

CustosAdicionais(): método abstrato, que deverá ser implementado pela classe que herdar.

* Automovel(POJO):

Classe Automovel, contém informações relativas ao automóvel, herdando os atributos da classe Veiculo.

- Atributos:

int qtdDePortas , int numeroDePassageiros;

- Métodos:

Um construtor vazio e um com todos os atributos de Veiculo e Automovel, gets e sets, equals e toString;

* Motocicleta(POJO):

Classe motocicleta, contém informações relativas à motocicleta, herdando os atributos da classe Veiculo.

- Atributos:

string estilo

- Métodos:

Um construtor vazio e um com todos os atributos de Veiculo e Motocicleta, gets e sets, equals e toString;

* Locacao:

Classe Locacao, contém informações relativas à locação feita.

- Atributos:

int id (identificador), String dataRetiradda, String dataDevolução, CustoDoAluguel, Pessoa, Veiculo veiculo;

- Métodos:

Um construtor vazio e um para todos os atributos, gets e sets, equals e toString.

calcValorLoc: método que calculará ovalor da locação. Retornando em double o valor da locação.

**\*Coleções**

* ColecaoPessoa:

Classe ColecaoPessoa, contém os métodos para criar e manipular uma coleção de classes do tipo Pessoa.

- Atributos:

ArrayList<Cliente> cliente;

- Métodos:

carregarArqDePessoas: método utilizado para acessar servidor de arquivos e carregar as informações contidas no arquivo "pessoas.obj" para o sistema. Retorna boolean, em caso sucesso true, e em caso de falha exibe mensagem e retorna false.

salvarArqdePessoas: método utilizado para salvar a coleção de pessoas no arquivo "pessoas.obj" contido no servidor.

addPessoa: método utilizado para adicionar pessoas a coleção, passando como parâmetro um objeto do tipo Pessoa, e retornando true em caso de êxito ou false em caso erro.

removerPessoa: método utilizado para remover pessoas da coleção, passando como parâmetro um tipo String relativo ao cpf da pessoa a ser removidas, e retornando true em caso de êxito ou false em caso erro.

pesquisarPessoa: método utilizado para pesquisar pessoas da coleção, retornando o objeto pessoa em caso de êxito ou uma mensagem de erro em caso de erro.

listarPessoa: método utilizado para listar pessoas da coleção, retornando o objeto ArrayList de todas as pessoas adicionadas na coleção em caso de êxito ou uma mensagem de erro em caso de erro.

listarCliente: método utilizado para listar Clientes da coleção, retornando um objeto ArrayList de todos os clientes adicionados na coleção em caso de êxito ou uma mensagem de erro em caso de erro.

listarFuncionario: método utilizado para listar Funcionarios da coleção, retornando o um objeto ArrayList de todas os funcionários adicionados na coleção em caso de êxito ou uma mensagem de erro em caso de erro.

* ColecaoVeiculo:

Classe ColecaoVeiculo, contém os métodos para criar e manipular uma coleção de classes do tipo Veiculo.

- Atributos: ArrayList<Funcinario> func;

- Métodos:

carregarArqDeVeiculos: método utilizado para acessar servidor de arquivos e carregar as informações contidas no arquivo "veiculos.obj" para o sistema. Retorna boolean, em caso sucesso true, e em caso de falha exibe mensagem e retorna false.

salvarArqdeVeiculos: método utilizado para salvar a coleção de veículos no arquivo "veiculos.obj" contido no servidor.

addVeiculo: método utilizado para adicionar veiculos a coleção, passando como parâmetro um objeto do tipo Veiculo, e retornando true em caso de êxito ou false em caso de erro.

listarVeiculo: método utilizado para listar veículos da coleção, retornando o objeto ArrayList de todas as veículos adicionados na coleção em caso de êxito ou uma mensagem de erro em caso de erro.

listarAutomoveis: método utilizado para listar automóveis da coleção, retornando o objeto ArrayList de todas os automóveis adicionados na coleção em caso de êxito ou uma mensagem de erro em caso de erro.

listarMotocicletas: método utilizado para listar motocicletas da coleção, retornando o objeto ArrayList de todas as motocicletas adicionados na coleção em caso de êxito ou uma mensagem de erro em caso de erro.

pesquisarVeiculo: método utilizado para pesquisar veículos da coleção, retornando o objeto Veiculo em caso de êxito ou uma mensagem de erro em caso de erro.

listarVeiculosDisponiveis: método utilizado para listar veículos com o atributo disponibilidade válido como true da coleção, retornando o objeto ArrayList de todas os veículos adicionados na coleção em caso de êxito ou uma mensagem de erro em caso de erro.

removeVeiculo: método utilizado para remover veículos da coleção, passando como parâmetro um tipo String relativo ao número da placa dos veículos a serem removidos, e retornando true em caso de êxito ou false em caso de erro.

* ColecaoLocacao:

Classe ColecaoLocacao, contém os métodos para criar e manipular uma coleção de classes do tipo Locacao.

- Atributos:

ArrayList<Locacao> locacao;

- Metodos:

carregarArqDeLocacoes: método utilizado para acessar servidor de arquivos e carregar as informações contidas no arquivo "locacoes.obj" para o sistema. Retorna boolean, em caso sucesso true, e em caso de falha exibe mensagem e retorna false.

salvarArqdeLocacoes: método utilizado para salvar a coleção de veículos no arquivo "locacoes.obj" contido no servidor.

addLocacao: método utilizado para adicionar locacoes a coleção, passando como parâmetro um objeto do tipo Veiculo, e retornando true em caso de êxito ou false em caso erro.

pesquisaLocacao: método utilizado para pesquisar locações da coleção, retornando o objeto Locacao em caso de êxito ou uma mensagem de erro em caso de erro.

pesquisarLocacaoPorPessoa: método utilizado para pesquisar locações da coleção passndo por parametro uma String referente ao cpf do Cliente, retornando o objeto Locacao em caso de êxito ou uma mensagem de erro em caso de erro.

pesquisarLocacaoPorVeiculo: método utilizado para pesquisar locações da coleção passando por parametro uma String referente aos numeros e Letras da placa do veículo, retornando o objeto Locacao em caso de êxito ou uma mensagem de erro em caso de erro.

listarLocacao: método utilizado para listar locações da coleção, retornando o objeto ArrayList de todas os locações adicionadas na coleção em caso de êxito ou uma mensagem de erro em caso de erro.

removerLocacao: método utilizado para remover locações da coleção, passando como parâmetro um tipo int relativo ao número do identificador(id) da locação a ser removida, e retornando true em caso de êxito ou false em caso erro.

**\*Interfaces**

IController: interface que contém os métodos para uso das coleções desse pacote, devendo-se ser implementado os métodos na classe que implementar essa interface.

Controller: implementa os métodos da interface IController;

**Pacote dao:**

**\*Classes**

* Idao: interface que contém os métodos para salvamento dos arquivos, devendo-se ser implementado os métodos na classe que implementar essa interface.
* ControllerDao;

Contém os métodos herdados da interface Idao e suas implementações:

salvarEmArq: salva objetos passados por parametro em um arquivo, com o caminho e nome do arquivo a ser salvo também passado por parâmetro;

carregarPessoas: recupera os arquivos salvos, passando como parâmetro uma String com o caminho e nome do arquivo ;

**Pacote net:**

Conterá os sockets necessários para a conexão com servidor.

* INetClient: interface que contém os métodos para conexão com o servidor de arquivos, devendo-se ser implementado os métodos pela classe que implementar essa interface;
* ControllerNetClient: Contém os métodos herdados da interface INetClient e suas implementações:

enviarObject: recebe como parâmetro um tipo Object e um tipo String, o tipo Object será enviado para o servidor e a string serve para informar o nome do arquivo a ser enviado ao servidor, sendo que existe apenas três nomes possíveis para envio: pessoas.obj, veículos.obj, locações.obj .

receberObject: recebe como parâmetro um tipo String, a string server para informar o nome do arquivo a ser obtido do servidor, sendo que existe apenas três nomes possíveis para recebiemento: pessoas.obj, veículos.obj, locações.obj.

**Pacote netServer**

Conterá os sockets do servidor.

* INetServer: interface que contém os métodos para inicialização do servidor de arquivos, devendo-se ser implementado os métodos pela classe que implementar essa interface;
* ControllerNetServer: Contém os métodos herdados da interface INetServer e suas implementações:

initServer: método utilizado para inicializar o servidor, recebendo o primeiro índice de do cliente, referente ao fato do cliente desejar enviar ou receber algo para o servidor.

receberObject: recebe como parâmetro um tipo Socket, referente a conexão com o cliente, tendo o objetivo de receber objetos por parte do cliente e salvado estes em arquivos, sendo a opção de salvamento reservado a três tipos de arquivos: pessoas.obj, veículos.obj, locações.obj. Retorna true em caso de êxito ou false em caso de falha.

enviarObject: recebe como parâmetro um tipo Socket, referente a conexão com o cliente, tendo o objetivo de enviar objetos para o cliente e que estará contido em arquivos salvos, sendo reservado a três tipos de arquivos: pessoas.obj, veículos.obj, locações.obj. Retorna true em caso de êxito ou false em caso de falha.

* TesteNetServer: Classe principal que conterá o método main para inicialização do servidor.

**Pacote view**

Cotenterá a classe Principal, onde possuirá o método main. Metodos adicionais forão utilizados para interação com o usuário.

MENU:

0- Sair

Pessoa:

1- Adicionar Pessoa

2- Listar Pessoal

3- Listar Clientes

4- Listar Funcionarios

5- Pesquisar Pessoa

6- Remover Pessoa

Veiculo:

7- Adicionar Veiculo

8- Listar Veiculos

9- Listar Veiculos Disponiveis

10- Listar Automoveis

11- Listar Motocicletas

12- Procurar Veiculo

13- Remover Veiculo

Locacao:

14- Adiconar Locacao

15- Listar Locacoes

16- Procurar Locacao por ID

17- Procurar Locacao por Pessoa

18- Procurar Locacao por veiculo

19- Remover Locacao

- Métodos adicionais:

public boolean adicionarPesssoa() : faz a interatividade com o usuário para obtenção dos valores a serem atribuídos aos atributos de Pessoa e seus herdeiros ao adicionar no sistema. Retornará true em caso de êxito e false em caso de erro.

public boolean lerdadosPessoa(): faz a interatividade com usuário para obtenção dos atributos específicos da classe Pessoa. Retornará true em caso de êxito e false em caso de erro.

public boolean adicionarVeiculo(): faz a interatividade com o usuário para obtenção dos valores a serem atribuídos aos atributos de Veiculo e seus herdeiros ao adicionar no sistema. Retornará true em caso de êxito e false em caso de erro.

public boolean lerdadosVeiculo(): faz a interatividade com usuário para obtenção dos atributos específicos da classe Veiculo. Retornará true em caso de êxito e false em caso de erro.

public boolean adicionarLocacao(): faz a interatividade com o usuário para obtenção dos valores a serem atribuídos aos atributos de Locacao ao adicionar no sistema. Retornará true em caso de êxito e false em caso de erro.

lerNome(): ler apenas String, caso digitado qualquer valor como número, será solicitado que se digite o nome novamente, repetindo-se enquanto um nome não for digitado.

lerInteiro(): ler apenas inteiro, caso digitado qualquer valor diferente de um inteiro, será solicitado que se digite novamente, repetindo-se enquanto um inteiro não for digitado.

lerDouble(): ler apenas números do tipo Duble, caso digitado qualquer valor diferente de um numero, será solicitado que se digite novamente, repetindo-se enquanto um número não for digitado.

lerDocumento(): ler apenas números, caso digitado qualquer valor diferente de um número, será solicitado que se digite novamente, repetindo-se enquanto um número não for digitado.

lerData(): ler apenas a data no formato especificado, caso digitado qualquer valor diferente da data especificado, será solicitado que se digite novamente, repetindo-se enquanto o formato da data não for compatível.

getData(): ler data no formato LocalDateTime. Retornando o tipo LocalDateTime.

getData2(): ler data no formato LocalDate. Retornando o tipo LocalDate.

lerAno(): interatividade com usuario para ler ano em formato de numero.

lerMes(): interatividade com usuario para ler mes em formato de numero.

lerDia(): interatividade com usuario para ler dia em formato de numero.

lerHora(): interatividade com usuario para ler ano em formato de numero.