Discente: Maria Clara Alves Coelho

Estacionamento: Algoritmo em Linguagem Natural

1. Definir classe Cliente, que será responsável pelo veículo:

- Propriedades:
    - Nome “nome”: String;
    - CPF “cpf”: int;
    - Telefone “telefone”: int;
- Implantar os métodos:
    - toString para retornar as propriedades de cliente em formato de texto;
- Implementar construtor com os parâmetros: String, int, int (para definir nome, cpf e telefone, respectivamente).

2. Definir classe Veiculo, que será alocado:

- Propriedades:
    - Cliente responsável pelo veículo “cliente”: Cliente;
    - Placa “placa”: String;
    - Modelo do veículo “tipo”: String;
- Implementar os métodos:
    - getPlaca: retorna número da placa;
    - getTipo: retorna modelo do veículo;
- Implementar construtor com os parâmetros: Cliente, String, String (para definir cliente, placa e tipo, respectivamente);

3. Definir classe Vaga, referente às vagas existentes no estacionamento:

- Propriedade:
    - Número da vaga/localização numero: int;
- Implementar métodos:
    - getNumero: retorna número da vaga;
    - toString: retorna as propriedades da Vaga em formato texto;
- Implementar construtor sem parâmetros, mas definindo número de vaga pelo incremento do último número de vaga, de forma que as vagas tenham valores sequenciais e únicos.

4. Definir classe Alocacao, referente as alocações de vagas:

- Propriedades:
    - hora de entrada do veículo “horaEntrada”: int;
    - hora de saída do veículo “horaSaida”: int;
    - Veiculo “veiculo”: Veiculo;
    - Vaga “vaga”: Vaga;
- Implementar métodos:
    - calcularTempoPermanencia: para calcular tempo de alocação = tempo saída - tempo de entrada (quando hora de saída > hora de entrada, ex: entrada 5 h e saída 11 horas) ou 24-tempo de entrada + tempo de saída (quando hora saída < hora de entrada, ex; entrada 22 h e saída 5h da madrugada).
    - calcularCusto: para calcular custo = preço por hora x tempo de permanência;

    - getVaga: devolve vaga;
    - getVeiculo: devolve veículo;
    - setHoraSaida: definir hora de saída do veículo;
    - toString: retorna as propriedades da Alocação em formato texto;
- Implementar construtor com parâmetros: int, Vaga, Veiculo (para definir horaEntrada, vaga e veículo, respectivamente).

5. Definir classe Estacionamento, com os registros e alocações:

- Propriedades:
    - Valor do preço por hora “precoHora”: int (definido na regra de negócio para valer 10).
    - Conjunto de todas as vagas do estacionamento “vagasTotais”: ArrayList<Vaga>;
    - Conjunto de todas as vagas disponiveis “vagasDisponiveis”: ArrayList<Vaga>;
    - Conjunto de veículos registrados “veiculosRegistrados”: ArrayList<Veiculo>;
    - Registro das alocações “alocs”: ArrayList<Alocacao>;
    - Referência a classe com mensagens de erro “e”: Excecao;
- Implementar métodos:
    - registrarVeiculo(): inclui veículo no conjunto de veículos do estacionamento;
    - registrarVaga(): inclui vaga nos conjuntos de vagas totais e disponíveis do estacionamento;
    - alocar: Alocar a vaga a partir dos parâmetros referentes a hora de entrada, número de vaga, placa, respectivamente.
    - desalocar: Desalocar a vaga a partir dos parâmetros referentes a hora de saída e número de vaga, respectivamente;
    - buscarAlocacao: retorna a alocação dado um número de vaga;
    - buscarVaga: retorna a vaga dado um número de vaga;
    - buscarVeiculo: retorna veículo dado um número de placa;
    - getPrecoHora: retorna o preço por hora da alocação de vaga;
    - toString: retorna as propriedades do Estacionamento em formato texto;
    - mensagemSaida: Mensagem de saída para quando finalizar alocação;
    - verVagasTotais: retorna todas as vagas juntamento com seus estados (Disponível ou Ocupado) em formato texto;
    - verVeiculosRegistrados: retorna todos os veículos registrados em formato texto;

6. Definir classe Telas, que serve para telas usadas como interface entre as operações e entradas de dados das propriedades da classe Estacionamento:

- Propriedades:
    - Define uma instância de objeto Estacionamento “est” para manipular os registros e operações do estacionamento: Estacionamento;
- Implementar métodos de telas:
    - telaMenu: Mostra um menu para em loop para que o usuário possa registrar e realizar operações;
    - telaRegistroCliente: Mostra tela de entrada de dados para registro do cliente;

    - telaRegistroVeiculo: Mostra tela de entrada de dados para registro de veículos;
    - telaAlocar: Mostra tela para realizar alocação;
    - telaDesalocar: Mostra tela para realizar a desalocação;
    - telaVerAlocacoes: Tela com registro de alocações atuais;
    - telaVagas: Tela com vagas do estacionamento;
    - telaVeiculos: Tela com o registro de veículos;
    - telaSair: para sair do loop menu e parar a execução do programa;
    - telaErroOpIndiponivel: Tela que mostra mensagem de erro de opções do menu para usuário.
- mplementar construtores:
    - Tela(): sem parâmetro, cria uma nova instância de estacionamento que será manipulada;
    - Tela(Estacionamento): com parâmetro Estacionamento, que ser manipulado;

7. Definir classe Excecao, para envio de mensagem de exceções:

- Implementar métodos:
    - msgVagaInexistente: Mostra mensagem de vaga inexistente;
    - msgVagaIndisponivel: Mostra mensagem de vaga indisponível;
    - msgNoInexistente: Mostra mensagem de vaga inexistente;
    - msgVeiculoNRegistrado: Mostra mensagem de falta de registro de alocação;

8. Definir classe Main:

Usada para testar Estacionamento juntamente com as outras classes: Primeiramente, foi criada a instância de Estacionamento para realizar o registro de vagas, veículos e alocações. Depois, chama as telas usadas como interface entre as operações e entradas de dados das propriedades da classe Estacionamento. Como já foi criado o objeto *est* do tipo estacionamento para entrada de dados, o construtor de tela é enviando *est* como parâmetro, mas também tem a opção de criar uma nova instância de classe Estacionamento, basta usar o construtor Telas() sem parâmetros.