Relatório

Sistema de Estacionamento e Aluguel de Carros
Programação Orientada a Objetos

Alvaro P. Tirado RA:210656
Caio A. A. Nolasco RA:195181
Thais A. Bispo RA:187386

Technical Report - IC-18-00 - Relatório Técnico
July - 2018 - Julho

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Garagem e Sistema de Aluguel de Carros

Caio A.A. Nolasco[1], Álvaro Tirado[2], Thais A. Bispo[3]

[1] **Instituto de Computação Universidade Estadual de Campinas (UNICAMP), Caixa Postal 6176 13083-970 Campinas-SP, Brasil**

c195181 @g.unicamp.br

[2] **Instituto de Computação Universidade Estadual de Campinas (UNICAMP), Caixa Postal 6176 13083-970 Campinas-SP, Brasil**

a210656@g.unicamp.br

[3] **Instituto de Computação Universidade Estadual de Campinas (UNICAMP), Caixa Postal 6176 13083-970 Campinas-SP, Brasil**

t187386@g.unicamp.br

Resumo: O projeto a seguir representa um sistema de administração de um estacionamento, junto a um andar especializado no aluguel de 10 carros. As funções suportadas consistem em: estacionamento e de estacionamento de um carro, impressão de dados sobre um cliente, aluguel e retorno de um carro, impressão dos andares de garagem e impressão do catálogo de carros disponíveis para estacionamento.

O sistema é formado por diferentes hierarquias de generalização, que descrevem comportamentos específicos e gerais de componentes do estacionamento, Andar, Vaga, Carro, Cliente e suas respectivas subclasses, explorando o uso de classes abstratas e interfaces, especializadas em garagem ou aluguel, sendo a classe Estacionamento o elo que junta essas entidades.

Palavras-Chave: Gestão de um estacionamento; Garagem; Sistema de Aluguel de Carros; Gestão do sistema de aluguel de carros.

1. Introdução

Atualmente, o estacionamento tornou-se um elemento fundamental para o bom desenvolvimento de muitas atividades econômicas em qualquer país moderno. O estacionamento deixou de ser um luxo para ser uma necessidade para qualquer atividade, convertendo-se em um setor chave de forma direta e indireta. Se os cidadãos receberem um espaço físico para estacionar potencia-se a mobilidades de pessoas, bens e serviços que contribuem para qualquer atividade econômica.

O estacionamento passou de um aparente luxo de alguns para uma necessidade autêntica para todos. O estacionamento adequado melhora substancialmente atividade econômica, uma vez que facilita mais tempo para desenvolver uma atividade profissional, para comprar, para lazer, para economizar combustível e "melhorar o meio ambiente" através de redução das emissões de CO, reduzindo as situações de estresse, a possibilidade de acidentes ou outros conflitos.

Um fato importante é que mais do 18% dos veículos que circulam através das ruas das cidades eles fazem isso porque eles não podem encontrar um lugar para estacionar. É o que é conhecido no setor por "trânsito por agitação". Em outras cidades do mundo é até superior. Isto queremos dizer que nenhum tipo de atividade econômica está sendo desenvolvido, o que significa que se a os veículos das cidades é facilmente estacionada poderia melhorar em setores-chave como comércio, turismo, saúde, serviços e outros. Segundo os dados de alguns estudos cada vaga de estacionamento gera alguns benefícios na economia do seu entorno imediato cerca de 140 ou 160 reais por dia para empresas nas proximidades mas você também pode estimar que os próprios garagens são prestadores de serviços (limpeza, segurança, vigilância, fornecimento e manutenção de equipamentos e instalações), dando um serviço de qualidade, favorecendo o desenvolvimento normal da atividade econômico.

Por esses motivos, nosso grupo -G10 (osSemNome)- decidiu criar um sistema de estacionamento de carros que pode ser gestionado com poucos recursos e um sistema de aluguel de carros ecológicos.

2. Pacote Gerenciador

O pacote Gerenciador foi criada para fosse possível agrupar as classes que comportam os principais métodos do sistema, ele é formado por quatro classes e uma interface. Neste pacote os conceitos de Programação Orientada a Objetos implementados são: interface,polimorfismo e padrão singleton.

2.1 Gerenciador Aluguel

A classe Gerenciador Aluguel contém dois métodos estático, pois esses métodos está atribuído a classe e não ao uma instância dessa classe. O primeiro método estático implementado é *alugarCarro* possui como parâmetro um Cliente do tipo aluguel e um vetor de Carro Aluguel, este vetor representa o catálogo disponível para aluguel. Neste método o vetor é percorrido com o objetivo de encontrar um carro disponível que será associado ao Cliente e o Carro encontrado tem o atributo alugado setado para true. Caso não tenha nenhum Carro Aluguel disponível para aluguel é retornado null caso contrário é retornado o carro que disponível para aluguel.

O segundo método implementado refere-se ao retorno do carro alugado (*retonarCarro*) seus parâmetros são Cliente Aluguel e o vetor de Carro Aluguel que refere-se ao catálogo de carro, no método o carro alugado é setado para false a variável referente ao aluguel. O ID do carro é usado como índice do vetor para procura do carro associado ao cliente, e por fim o cliente associado ao carro é setado para null e o método possui como retorno o carro alugado.

2.2 Gerenciador Garagem

A classe Gerenciador Garagem tem como funcionalidade métodos semelhantes ao classe Gerenciador Aluguel porém especializados ao local Garagem, as duas funções contidas nesta classe

são: *desestacionarCarro* e *estacionarCarro*.

Os parâmetros do método estático estaionarCarro são o Cliente Garagem, linha, coluna, andar e a matriz de vagas do estacionamento. Neste método é verificado se a localização da vaga (linha,coluna e andar) está livre, também há um casting de char para int na variável linha, caso o lugar esteja vago, é criado um vetor *lugarOcupado* que armazena os dados referente a localização do carro para que seja possível setar a posição do carro e retornar um boolean true, caso a vaga não exista é retornado false.

O segundo método *desestacionarCarro* que recebe como parâmetro Cliente Garagem e matriz de vaga, os dados da localização do carro é usada como índice para matriz de vaga e por fim é realizada uma chamada do método *desocuparVaga* da classe Vaga que seta a vaga para null e por fim é retornado o carro que foi desestacionado.

2.3 Gerenciador Cliente

A interface Gerenciador Cliente concentra os métodos referente aos Clientes como cadastrar, descadastrar e métodos de busca com diferentes parâmetros. A opção do uso de interface ao invés de classe abstrata consiste no fato que em uma interface não é necessário a implementação dos métodos declarados, apesar de que no sistema esta interface é implementada em classes referente ao Gerenciador de Cliente (aluguel e garagem).

2.4 Gerenciador Cliente Aluguel

Esta classe possui atributos referente ao cálculo do montante do aluguel (*limiteMultas, taxaMultas, diariaValor, seguroValor*) e nesta classe é criado um ArrayList de Cliente Aluguel. O padrão de projeto de criação Singleton foi implementado nesta classe dado que no sistema não há necessidade de instanciar mais uma classe Gerenciador Cliente Aluguel, além dos métodos de set dos atributos e do cálculo do montante do aluguel nesta classe ocorre o polimorfismo das funções: cadastrar, descadastrar e busca que precisam ser implementadas devido a implementação da interface Gerenciador Cliente.

2.5 Gerenciador Cliente Garagem

Nesta classe é implementado a interface Gerenciador Cliente e também é usado o padrão Singleton para que não seja possível ter mais um objeto. Ela possui a mesma finalidade que a classe Gerenciador Cliente Aluguel portanto possui os principais métodos que envolvem o Cliente Garagem para isso a classe contém o ArrayList de Cliente Garagem e assim como na classe Gerenciador Cliente Aluguel também há polimorfismo com os métodos declarados na interface.

3. Pacote Principal

O pacote principal possui apenas uma classe chamada pelo mesmo nome que contém o método main, é o pacote mais importante do programa nela é instanciado estacionamento e

definido o valor da diária do aluguel, taxa de multa, preço para opção do seguro, limite de multas aceito pelo sistema e preço por andar da garagem.

3.1 Principal

Na classe principal é instanciado um estacionamento e definido os seus atributos para que sejam passados como parâmetros. Nela há vincula com a classes referente a interface gráfica (contidas no pacote janelas) com a chamada do método iniciar menu que possui como parâmetro um objeto da classe estacionamento.

4. Pacote Locais

4.1 Andar

Classe abstrata do projeto. Em essa classe temos métodos como *getNomeAndar* e *setNomeAndar* e o construtor com o atributo do *nomeAndar.*

4.2 Andar Aluguel

Classe que herda de Andar. Nessa classe temos o vetor Carro Aluguel com os 10 carros disponíveis para aluguel. O construtor Andar Aluguel cria objetos de CarroAluguel em um vetor com os atributos de marca, modelo, placa, cor, quilometragem, o tamanho -pequeno, médio ou grande- e número de referência do vetor.

O método alugarCarro é o responsável por avisar o sistema que um carro foi alugado, chamando o método do Gerenciador Aluguel e retornando o Carro Aluguel.

Como temos um método alugarCarro também precisamos de um método para retornar o carro. O método retornaCarro é o responsável de avisar o sistema que um carro foi devolvido, chamando o método do Gerenciador Aluguel retornando, finalmente, o Carro Alugado devolvido.

Outras funcionalidades da classe Andar Aluguel é que imprime o mapa de carros a ser alugados, checar se tem carros para alugar e aumentar ou diminuir eles quando um carro é devolvido o alugado.

4.3 Andar Garagem

Classe que herda de Andar. Nessa classe temos o mapa de vagas -uma matriz de carros que é null para vagas desocupadas.

O método estacionarCarro é o responsável por avisar o sistema que um carro vai ser estacionado, chamando o método do Gerenciador Garagem que diminui a quantidade de vagas e se é possível estacionar.

Como temos um método *estacionarCarro* também precisamos de um método para *desestacionar* o carro. O método desestacionaCarro avisa o sistema que um carro vai sair do estacionamento, chamando o método do Gerenciador Garagem para aumentar as vagas do estacionamento.

Outras funcionalidades da classe Andar Garagem é que imprime o mapa de vagas, checa se tem vagas e aumenta ou diminui elas.

4.4 Estacionamento

Na classe estacionamento temos o vetor de andares para depois atualizar, modificar as vagas do estacionamento.

As funcionalidades mais importantes que contém essa classe são: método que busca um cliente da garagem pela placa do carro ou pelo CPF, na hora de estacionar tem métodos que verificam se o cliente é maior de idade, a localização para estacionar o veículo no caso de ter vagas...

Essa classe também imprime os dados do cliente da garagem com base na busca do CPF e se deseja-se alugar um carro ele invoca o método de Andar Aluguel para buscar se tem carros disponíveis ou não. Quando deseja-se retornar o carro alugado essa classe verifica se o carro pertence á nossa garagem mediante o ID do aluguel ou a placa do carro. Por último, imprime as vagas por andar e imprime os aluguéis feitos.

4.5 Vaga

As funcionalidades mais importantes dessa classe são os métodos de ocupar vaga ou desocupar vaga e saber se tem alguma vaga disponível.

5. Pacote Janelas

O pacote “Janelas” agrupa as classes relacionadas a interface gráfica desenvolvida para o projeto usando a biblioteca Swing do Java. Genericamente, as classes são conectadas entre si por um Menu Principal que espera que o cliente informe a operação desejada. Feito a escolha, é invocado o método da classe correspondente para ler dados de entrada e chamar o método relevante de um objeto Estacionamento.

5.1 Menu Principal

Classe responsável por implementar a parte da interface gráfica principal que liga todos os outros elementos do programa. Para isso, utiliza ferramentas da biblioteca Swing do java, contendo botões para Estacionar, Desestacionar, Alugar, Retornar, Procurar Carro, Imprimir Cliente, Imprimir Andares e Imprimir Aluguéis.

5.2 Janela Estacionar

Classe que implementa a leitura de dados pessoais e do automóvel para estacionar um carro. Disponibiliza campos para leitura de nome, CPF, data de nascimento, andar e vaga desejada, número de diárias, tipo de carro (pequeno, médio, grande), placa do carro, modelo, marca, cor. Com estas informações, chama método de um objeto de Estacionamento para realizar a operação especificada.

5.3 Janela Alugar

Classe da interface gráfica que comanda a operação de aluguel. Disponibiliza campos para ler nome, CPF, data de nascimento, número de multas do cliente, número de diárias e se o usuário quer ou não seguro para o carro.

5.4 Janela Desestacionar:

Classe da interface gráfica responsável ler a placa do carros procurado e invocar o método de Estacionamento correspondente.

5.5 Janela Retornar

Janela para leitura do carro que está sendo retornado, sua quilometragem atual para o cálculo do montante, e impressão de um pop-up informando o valor calculado por Estacionamento e seus gerenciadores.

5.6 Janela Catálogo Aluguel

Parte da interface gráfica que informa o cliente sobre os carros disponíveis para aluguel.

5.7 Janela Imprimir Cliente

Classe da interface gráfica que imprimir um pop up dos dados de um cliente buscado nos Gerenciadores Clientes por seu CPF.

5.8 JanelaProcurarCarro

Classe da interface gráfica que imprime um pop up dos dados de um carro de acordo com sua placa.

6. Pacote Usuários

O pacote “Usuários” agrupa as classes representantes dos clientes e seus veículos, caracterizadas por terem várias instâncias de seus objetos na execução do programa em comparação a objetos de classes do pacote “Locais”. Tanto para objetos de clientes quanto para seus carros, existem variantes especializadas de para garagem e aluguel, que descrevem comportamentos

diferentes relevantes ao contexto de seu uso.

6.1 Carro

Classe que descreve um objeto carro, armazenados em Estacionamento e Andar Garagem e Andar Aluguel. Seus atributos são “tipo”, (pequeno, médio, grande), “marca”, “modelo”, “placa”, “cor”, “quilometragem”, “andar”, e “localizacao”, um vetor no qual a posição 0 guarda a linha de sua vaga, e a posição 1 sua coluna. Além disso, a classe suporta métodos get e set para seus atributos e um método de impressão de dados.

6.2 CarroAluguel

Subclasse de Carro, a classe CarroAluguel descreve o comportamento de carros especializados para aluguel, que são diferenciados de carros normais por terem um atributo “ID”, que marca o número do carro no catálogo de aluguéis, e um booleano alugado, que marca se o carro está ou não disponível no momento. O método de impressão de dados é sobrescrito para acomodar as diferenças em relação a superclasse.

6.3 Cliente (classe abstrata)

Um objeto de Cliente tem como atributos “nome”, “data de nascimento”, “CPF”, e uma agregação com um objeto Carro. Como métodos, existem gets e sets para atributos, método para cálculo da idade para verificar se Cliente é maior de idade, e um método abstrato de impressão.

6.4 Cliente Garagem

Variante de Cliente voltada para os serviços de estacionamento, esta subclasse guarda o número de diárias que cliente ficará estacionado.

6.5 Cliente Aluguel

Subclasse de Cliente orientada para o sistema de aluguéis, tem um atributo “seguro”, para saber se cliente optou ou não por contratar seguro, “qtdMultas”, que guarda o número de multas que o cliente tem registradas em sua carteira para verificar se ultrapassou o limite estabelecido.

7. Conclusão

O projeto desenvolvido buscou amenizar o grande transtorno que o trânsito se tornou na vida diária, criando um sistema dinâmico de estacionamento, para que motoristas tenham um opção inteligente onde deixar seus veículos. Devido ao fato de o estacionamento ser gerenciado por um

sistema computacional, as operações são realizadas de forma muito mais rápida, aumentando a capacidade de demanda do estabelecimento

O sistema em questão aplicou os conceitos aprendidos de programação orientada a objetos aprendidos na disciplina de MC302, como herança entre as classes Andar, Andar Garagem e Andar Aluguel, a interface Gerenciador Cliente e a classe abstrata Cliente. O objetivo foi desenvolver um programa que estivesse de acordo com a emenda do curso, aplicando na prática o material estudado, e tivesse presença no contexto social.

Durante o correr do semestre, foi preciso muitas vezes repensar o código para chegar a uma conclusão de como incorporar ao código novos conceitos conforme eram ensinado, levando a reestruturação do código algumas vezes durante o desenvolvimento do projeto. Por exemplo, para a etapa 4, foram transferidas todas as tarefas de que promoviam alguma alteração no mapa de vagas de um andar ou que precisavam de acesso ao vetor de carros para aluguel, assim como o armazenamento da lista de clientes, para novas classes gerenciadoras, assim como o armazenamento da lista de , que formariam o pacote Gerenciadores. Procedimentos parecidos foram tomados para implementação de herança, classes abstratas e interfaces. Estas consequentes remodelagens do programa foram benéficas para o aprendizado, pois levavam a uma ponderação da participação dos aprendizados da disciplina em uma situação real de desenvolvimento de software.

De forma geral, o criação deste sistema de estacionamento foi um experiência nova para nós estudantes, sendo o primeiro envolvimento de alguns membros do grupo no criação de um projeto de escala maior, principalmente seguindo a ideologia de programação orientada a objetos. Isso propôs uma oportunidade especial de lidar com questões, tanto didáticas quanto em relação a organização de um projeto em grupo, que eventualmente podem aparecer em nossas carreiras profissional na área de computação.

8. Referências

<http://www.asesga.org/documentos/revista_aparcar/Aparcar_40.pdf>

<https://g1.globo.com/carros/noticia/compartilhamento-de-carros-cresce-e-ganha-mais-opcoes-no-brasil.ghtml>