Documentação do Trabalho de Orientação a Objetos

Luciano Machado - 180126130
Lucas Rocha - 150137478
Kathlyn Lara - 180042378

Este documento tem o fim de expor toda a arquitetura, design e idéias para a resolução do trabalho dado pelo Prof. Lanna. (Leitura Aconselhável)

Para a inicialização da construção da arquitetura de software, deve-se identificar todos os tipos de usuários. Estes são os operadores que usarão o sistema, identificando eles poderemos estabelecer em seguida uma interface bem definida contendo todas as operações que esses usuários poderão utilizar. Ao ler o projeto, podemos identificar somente um e único tipo de usuário que chamaremos de Funcionário. Somente o funcionário do estacionamento usará o programa. Para este Funcionário, pode-se identificar as seguintes operações, as operações gerais do programa (Uso-Caso):

- Cadastrar Veículo

Esta função será executada pelo Funcionário somente antes do primeiro registro de um veículo

- Cadastrar Cliente

Esta função será executada na necessidade de cadastrar os dados de uma pessoa, não sendo dessa vez que uma pessoa possua mais de um veículo.

- Cadastrar Acesso

Esta função será executada toda vez que um veículo acessar o estacionamento.

- Buscar Veículo

Esta função será executada pelo usuário toda vez que ele gostaria de visualizar os veículos registrados no sistema

- Buscar Cliente

Esta função será executada pelo usuário toda vez que ele gostaria de visualizar os clientes registrados no sistema

- Buscar Acesso

Esta função será executada pelo usuário toda vez que ele gostaria de visualizar os acessos registrados no sistema que ainda não foram finalizados

- Deletar Veículo

Esta função será executada pelo usuário toda vez que ele gostaria de remover um veículo registrado no sistema

- Deletar Cliente

Esta função será executada pelo usuário toda vez que ele gostaria de remover um cliente registrado no sistema

- Finalizar Acesso

Esta função será executada pelo usuário toda vez que ele gostaria de remover um acesso registrado no sistema e mostrar o valor resultante para o pagamento com o fim de finalizá-lo

Sendo esta interface bem estabelecida, podemos construir nosso primeiro diagrama significativo para o projeto. O diagrama chama-se Diagrama de Uso-Caso e é de onde se deve começar a fazer qualquer arquitetura na maioria das vezes como nesse, que seguirá abaixo:

Em seguida devemos estabelecer quais são os meios que o programa fará interação com o mundo. Neste caso, o programa tem natureza gráfica. Devido ao fato de um dos requerimentos do software ser escrito em Java, utilizaremos a biblioteca Swing que já é fornecida. Podemos perceber também que o programa fará somente uma janela. Uma vez que essa janela se mudará muitas vezes no decorrer da vida do programa devemos conceitualizar duas coisas: Framework e Procedimento.

Para conceitualizar Framework, suponha que você deverá cortar um papel em um determinado formato. Medições são feitas e o corte é feito. Agora suponha que você deverá cortar da mesma forma mil papéis. É inviável fazer mil medições. Se faz um Framework do corte e se corta os mil papéis de uma só vez. Da mesma forma funcionará a janela do programa. Um Framework deverá ser feito para ela contendo todas as informações e instruções necessárias para se modificar de forma desejável e bem estabelecida a janela. Assim, o programador não chamará instruções do Swing toda vez, o Framework da janela automatizará todos os processos.

Para conceitualizar Procedimento se deve ter a noção do Framework dito previamente. Um procedimento nada mais é que uma série de instruções com sua própria lógica que fará uso de um Framework. Logo, para este projeto, procedimentos nada mais são que mutações que a janela sofrerá.

A interação com o usuário deverá ser feita por via de botões e edição de texto, portanto o framework da janela deve possibilitar colocar texto na janela (label), botão, editor de texto (text fields), modificar os textos já colocados e receber as strings digitadas nos editores,etc. Cada tipo de modificação da janela terá seu respectivo procedimento que construirá em ordem os objetos colocados na janela.

Para o Java, Procedimento deve ser uma interface na qual ela cabe muito bem com esse conceito. Isso por dois motivos: primeiro, todos os procedimentos devem haver uma

função de atualizar, que recebe um objeto do framework dito acima e atualize a janela vista pelo usuário em tempo real. Uma vez que cada janela tem um funcionamento diferente, ela é composta de várias funções uma após a outra, daí o nome de Procedimento que as classes receberam. Devemos lembrar que a função atualizar não é a mesma coisa que o construtor de sua classe, pois a classe deve operar seus atributos conforme o usuário mexe na tela e de procedimentos posteriores ou anteriores. Segundamente, os procedimentos deverão em alguns casos passar como atributos outros procedimentos quaisquer. Sendo assim, não é viável criar um método para cada situação. Neste caso, passar o objeto de procedimento como a interface Procedimento soluciona para todos os casos. Logo, a interface Procedimento cabe muito bem em seu lugar, que pode ser encontrada no pacote gui (Graphical User Interfaces).

Essa versatilidade da interface nos leva a necessidade de truncar para um procedimento qualquer caso queiramos receber um valor de outro objeto procedimento. Podemos ver essa necessidade na classe ProcedimentoRegistrarCliente por exemplo em seu método de botão procedimento_registrar(). Nem sempre o procedimento anterior dado pelo atributo procedimento_anterior é o procedimento ProcedimentoRegistrarVeiculo. Neste caso houve uma necessidade de realmente usar Exceções. Todas as Exceções foram utilizadas para este fim no decorrer do programa uma vez que truncar algo intrucável gera um erro. Não há nenhuma Exceção do java que foi utilizado sem a real necessidade. Caso houvesse um vetor vazio e algum procedimento deveria lê-lo, por exemplo, a janela deveria se atualizar de outra forma e não utilizar uma exceção.

A maioria dos procedimentos contém os atributos procedimento_atual e procedimento_anterior. Eles são usados para a movimentação de um procedimento para o outro, geralmente utilizados nos botões “Voltar” em seus respectivos métodos privados. Foi necessário “atributar” certos valores pois como pode se perceber dentro das classes procedimentos, cada botão tem seu respectivo método privado que retorna uma instância de ActionListener. Esses ActionListeners são (Hidden Inner Class), classes que não tem o mesmo valor do “this” das classes que estão. Logo, foi necessário criar atributos para os métodos privados dos botões poderem acessar.

Podemos perceber tanto no código como na UML que alguns procedimentos herdam de ProcedimentoProcurar. Isso porque esta classe têm uma função abstrata que é atribuída a uma função de botão que pode fazer qualquer coisa dependendo das classes filhos. Polimorfismo pode ser encontrada na relação entre essas classes, pois ProcedimentoProcurar permite o uso da mesma função por diferentes classes. Perceba também o uso de Generics nessa classe. Isso devido o fato de ela poder representar qualquer tipo de objeto em botões, sendo cada botão em sua listagem (representada por números no diagrama de interfaces na UML), um objeto. Exemplo: Mostrar, Deletar… Cada classe filho deverá declarar o método para os botões da lista fazendo o seu dever.

Percebe-se também que quase todas as classes do gui são protegidas com exceção do procedimento ProcedimentoPaginaPrincipal. Isso porque ProcedimentoPaginaPrincipal é um ponto de entrada, acessada pela main, nenhum outro procedimento deverá instanciar outro procedimento senão as que já estão no pacote (gui).

O pacote dados contém classes de natureza de registro. Inclusive, há uma classe Registro. Ela é responsável por armazenar todos os objetos em vetores que terão dados salvos

do programa. Acesso, Veiculo e Cliente deverão ser públicos, pois os procedimentos poderão usá-las como bem entenderem, mas sempre salvarão e obterão a partir de Registro.

Acima houve a explicação da arquitetura do projeto. Uma vez que o código não está comentado, há consistência em todas as suas formas. Todas as classes foram organizadas de forma padronizada permitindo que nosso grupo pudesse usar o paradigma da caixa preta. Em outras palavras: façam seus métodos confiando totalmente que o método do amiguinho vai funcionar. Assim, dividimos de forma concisa todo o trabalho.

Técnicalidades: todo o código está dentro da pasta “src”. Como houve problemas durante o semestre com o Eclipse, caso você também tenha (incompatibilidade de versões, etc..) basta criar um novo projeto no eclipse e passar os códigos do “src”, ademais todos os arquivos necessários para o Eclipse funcionar também estão fora do “src”, tipo “bin”, “.project”, “.classpath”, etc…

Agradecemos a leitura,