FIAP

TURMA 1TDSPV

DOMAIN DRIVEN DESIGN USING JAVA

PROFESSOR: RAFAEL DESIDERIO

DEDAFIO PORTO SERGURO

DIAGRAMA DE CLASSES OFICINA VIRTUAL

SPRINT 2

PAULO ANDRÉ CARMINATI
RM 557881

CARLOS EDUARDO RODRIGUES COELHO PACHECO
RM 557323

GUSTAVO GOMES MARTINS
RM 555999

SUMÁRIO

DESCRITIVO

O seguinte diagrama de classes é uma versão prévia que será desenvolvida em Java. De início, serve para suprir as demandas de entregas exigidas pela faculdade referente a avaliação de nota.

Objetivo atual de sprint 2 é demostrar as classes bean e suas ligações com outras classes e atributos de referência, assim, como métodos de orçamento e pagamento feitos com valores de entrada criados pelo usuário e demostrados na classes de execução, om algum tratamento de erro inicial. Criou-se ainda onde necessário os métodos toString que podem se desejar ser implementados na classe de execução.

Em resumo, no momento o projeto conta com 10 classes, 1 superclasse veículo que manda seus dados como herança para carro e moto. E todas as outras classes trabalhando com objetos por referência. Nas classes Orcamento e Pagamento temos em cada uma um método de cálculo de pagamento sendo que nesta última ela trabalha com um método que usa a classe orçamento para fazer um cálculo com seu dado.

Como justificativa, temos o pedido de desenvolvimento de um sistema por parte da Cliente Porto Seguro, que supra suas necessidades quando ao mercado de oficinas mecânicas, para atrair novos clientes voltados a este tipo de serviço. Portanto, trata-se de um protótipo inicial.

O diagrama de classes atual e inicial comporta atualmente as seguintes classes:

1. **Cliente;**
2. **Veiculo;**
3. **Carro;**
4. **Moto;**
5. **Contato;**
6. **Endereco;**

7. **Agenda**
8. **Oficina;**
9. **Orcamento;**
10. **Pagamento;**

DIAGRAMA DE CLASSES

1 – Classe Cliente:

Comporta os atributos descritos no diagrama e recebe como referência todas as outras classes do diagrama (com exceção das classes carro e moto, que se ligam por herança com veículo).

2 – Classe Agenda:

Comporta os atributos descritos no diagrama.

3 – Classe Veiculo:

Comporta os atributos descritos no diagrama e trabalha como superclasses de herança de Moto e Carro.

6 – Classe Contato:

Comporta os atributos descritos no diagrama e serve como uma referência para a classe Cliente.

7 – Classe Endereco:

Comporta os atributos descritos no diagrama e serve como uma referência para a classe Cliente.

9 – Classe Oficina:

Classe que serve como referência para a classe Cliente, mas que serve para a questão dos problemas apresentados pelo veículo do cliente, assim como, os materiais usados pela oficina no conserto do veículo. Oficina recebe como referência a classe Agenda e Veículo.

10 – Classe Orcamento:

Serve como referência para a classe Cliente, mas recebe como referência à classe oficina, para que se possa fazer um histórico de tudo que foi feito no conserto do veículo e gerar um custo para o serviço prestado (método no caso).

11 – Classe Pagamento:

Também, está classe, faz se referência a classes Cliente. Isso ocorre para que o cliente possa fazer o pagamento pelo serviço prestado tendo de forma transparente tudo que foi feito em seu veículo. Está classe também recebe como referência a classe orçamento, exatamente, para manter a transparência com o cliente.

OBSERVAÇÃO:

Para que CERTOSDETALHES não fiquem repetitivos, explico-os neste momento:

1- Todas as classes recebem um método construtor de classe vazio;
2- Todas as classes recebem um método construtor cheio;
3- Todas as classes tem a geração dos métodos get e set.
4- Todas as classes com exceção de Veículo, receberam um método TO String, para futuro uso se necessário.
5- As classes Orçamento e Pagamento receberam um método específico cada que na primeira calcula todo o valor do serviço prestado e na última calcula o valor total do serviço mais o desconto.
6- Foi feito uma classe de execução ao estilo menu com apresentação para que se possa fazer um melhor teste das funcionalidades vias System.out .
7- Todas as classes em específico possuem um atributo código que posteriormente será autogerado pelo SGBD, e já pensando nele. Desta forma, preferimos não gerar seus métodos get e set, assim como, dentro do construtor cheio, para não sermos obrigados em caso de criação na classe teste de entrada de dados via construtor deste campo. Por isso, ele foi criado, mas não com tais métodos. E também para não ficar estranho a todo o momento na classe de execução o usuário digitar o código da classe.

DIAGRAMA DE CLASSES DESENHO