POO2

Integrantes:

Gustavo Grimaldi

Paulo Vianna

1. Apresentação

Este documento é um projeto de uma empresa fantasia de uma cooperativa de taxi, por isso pode não conter todos as características de uma empresa real. O trabalho pertence a disciplina de Programação Orientada a Objetos 2, cuja finalidade é aplicar o aprendizado de padrões de projetos criacionais aprendido em sala de aula. A seção 2 é uma descrição do propósito do sistema; A seção 3 detalha a metodologia aplicada ao trabalho; A seção 4 é uma descrição do minimundo apresentando o problema; A seção 5 apresenta a lista de requisitos de usuário do sistema; A seção 6 tem como objetivo explicar os padrões de projetos com base no diagrama de classe e/ou trecho de código; A ultima seção é sobre a utilização do sistema.

2. Propósito do Sistema

O sistema tem como propósito gerenciar uma cooperativa de taxi, que nada mais é que uma associação de taxistas para atender uma maior quantidade de gente com uma maior nível de confiabilidade por parte do cliente, pois ao contacta-la, o cliente sabe que muito provável achar um taxi disponível, alem de guardar registro dos taxistas, caso seja necessário.

3. Metodologia

A especificação do trabalho dava a liberdade para escolher qualquer padrão criacional a ser aplicados ao projeto, das opções, foram aplicados o Método Fábrica, Fábrica Abstrata e o Singleton. Além da escolha do padrão, o trabalho consistia em aplicar os padrões em outra linguagem orientada a objeto que não seja java, a linguagem escolhida foi o Python versão 3.4. Apesar das facilidades da linguagem, implementar alguns padrões foi um pouco complicado pela ausência do tipo interface e classes abstratas, na seção 6 vamos explicar como foi aplicado os padrões mesmo sem as classes.

4. Minimundo

A empresa Táxi Vix está há pouco mais de 5 anos no mercado de transporte de pessoas no estado do Espírito Santo. Porém, vem crescendo bastante no mercado por ser uma empresa que transporta pessoas de maneira segura e confiável. Para auxiliar suas tarefas, a empresa deseja um sistema para gerenciar suas atividades.

A empresa realiza viagens particulares em todo o estado do Espírito Santo. O Cliente entra em contato (por telefone) e um atendente realiza um cadastro, caso ainda não tenha. De um cliente deseja-se saber: Nome, CPF, telefone, e-mail, endereço. De um atendente deseja-se saber: Nome, CPF e endereço.

Depois de possuir um cadastro, o cliente solicita uma viagem ao atendente. O cliente informa o endereço de origem e o de destino, o tipo de táxi e a quantidade de pessoas que irão com ele (máximo de 15 pessoas). Os táxis podem ser divididos em três tipos, sendo eles: vip, luxo e básico . A empresa possui táxi básico de 4 lugares, vip de 7, e luxo de 15 lugares.

Ao fim do contato, o atendente em questão solicita a um motorista para que busque o cliente no local. Cada motorista é designado a apenas um táxi. De um motorista deseja-se saber: Nome, CPF, número da CNH, endereço.

A viagem possui uma tarifa que varia de acordo com o destino final e o tipo de táxi escolhido pelo cliente. No fim da corrida, é impresso por um taxímetro uma nota com o valor total da viagem, ou pode ser previamente negociado com a atendente o valor, para viagens mais longas.

5. Requisitos de usuário

**Requisitos Funcionais:**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Requisito | Prioridade | Dependência |
| RF01 | O sistema deve cadastrar clientes | Alta | RF01 |
| RF02 | O sistema deve cadastrar atendentes | Alta | RF02 |
| RF03 | O sistema deve cadastrar Taxista | Alta | RF05 |
| Rf04 | O sistema deve cadastrar viagem | Alta | RF01,RF02,RF03 |
| RF05 | O sistema deve criar taxis | Média | RF05 |

**Regras de Negócio**:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Descrição | Prioridade | Dependência |
| RN01 | O taxista deve ser cadastrado pelas atendentes | Alta | RF03, RF05 |
| RN02 | A atendente deve registrar viagens | Alta | RF01,RF02,RF03, RF04 |
| RN03 | A atendente deve registrar usuários | Alta | RF01 |
| RN04 | O taxista disponível deve pegar o passageiro | Media | RF05 |

**Requisitos Não Funcionais:**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Descrição | Prioridade | Dependência |
| RNF01 | O sistema deve notificar taxista | Média | RF05 |
| RNF02 | O sistema deve calcular valor da corrida | Baixa | RF01,RF02,RF03,RF04,RF05 |
| RNF03 | O sistema deve ter controle de acesso | Alta | RF02, RF03 |

6. Diagrama de Classe e Padrões

O diagrama de Classe está junto ao projeto no Github, não foi viável inserir neste documento, pois torna-se ilegível ao adaptar o tamanho.

A linguagem Python não possui classes abstratas nem interfaces, porém tem como criar classes com métodos e implementa-los nas classes que herdarem a classe pai. O método fica assim:

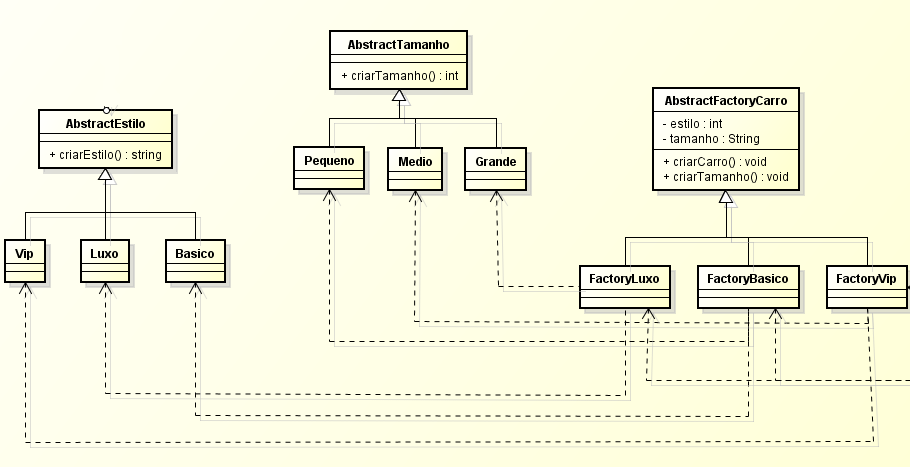
**def criarCarro(self):**

**raise Exception("Preciso ser implementado")**

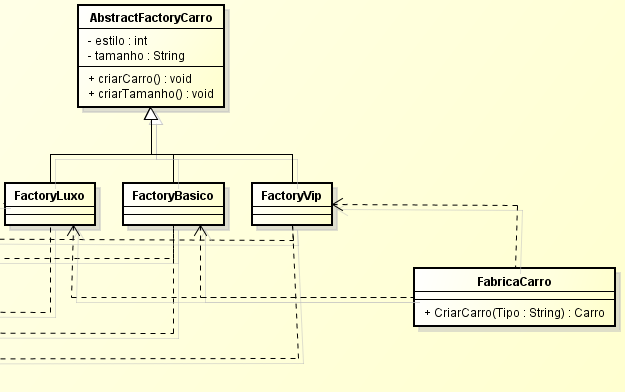
Os padrões de projeto utilizados foram:

**Fábrica abstrata**

Como mostra o diagrama abaixo, foram criados estilos de carros diferentes e tamanhos diferentes. Como informa no minimundo existe a fabrica de taxi Luxo, Vip e a de Básico. Cada uma cria o carro de acordo com suas respectivas características.

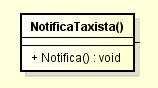


**Fábrica**

Uma fábrica de carros foi criada para a entrada seja o tipo do carro e a saída seja o carro desejado.

**Singleton**

Consiste em existir apenas uma instância da classe, no caso a de notificar motorista sobre alguem que contratou os serviços da empresa. A escolha do Singleton para essa classe é para que dois ou mais motoristas não irão buscar o mesmo cliente.



7. Modo de controle do sistema

O usuario seleciona se é um atendente ou um taxista, se for atendente abrira opções para o cadastro de motoristas, clientes, atendentes, e viagens, caso seja um motorista, o sistema vai ficar atualizando ate informar que existe uma viagem disponivel.

**Link do Projeto**

<https://github.com/GrimaG/POO2-Ruby/>

P.s: O repositório tem o nome de Ruby porque foi criando antes de mudarmos a linguagem.