Relatório
Trabalho prático 1
BCC – 221

Alunos:

Carlos Eduardo Gonzaga Romaniello de Souza - 19.1.4003;

Vinicius Gabriel Angelozzi Verona de Resende - 19.1.4005;

● Arquitetura e implementação do projeto

Para garantir a boa execução do programa e para facilitar o manuseio dos dados, foram criadas classes, modularizando o projeto. As classes construídas estão presentes no arquivo '***Poo - TP1.pdf***' caso tenha interesse em visualizar o diagrama UML do projeto, além disso todas as funções estão comentadas no código fonte e caso aberto no VS code, informações serão fornecidas em hover ao passar o mouse em cima das funções.

- **User:**

  A superclasse User foi criada para conter os dados de todos os usuários registrados no sistema da oficina. Ela contém o nome, CPF, email e senha do mesmo.

- **Admin, Vendor e Mechanic:**

  Essas classe são subclasses da classe User e possuem um ID para poderem se diferenciar entre si no momento do login. O ID 1 se refere ao Admin, 2 ao Mechanic e 3 ao Vendor.

- **Service:**

  Essa classe foi criada para poder armazenar todos os serviços que podem ser realizados dentro da oficina. Possui o nome do serviço e o preço.

- **AutoPart:**

Essa classe possui a mesma finalidade da classe service e possui os campos que guardam o nome da peça, seu valor individual e a quantidade utilizada.

- **ServiceOrder:**

  Essa classe é uma classe que agrega Service e AutoPart e representa a ordem de serviço que é gerada pelo vendedor. Ela possui um atributo static chamado count que é utilizado para preencher o atributo ID que será utilizado mais tarde para encontrar a ordem de serviço. Além desses dois, temos os atributos tipo (orçamento ou manutenção), motivo que descreve a ordem de serviço, a placa do carro e o CPF do dono, a quilometragem do veículo, dois atributos booleanos que indicam se a ordem de serviço foi aprovada e se ela já foi concluída, o custo, e dois mapas que armazenam os serviços prestados e as peças utilizadas.

- **Vehicle:**

Essa classe armazena os dados de um veículo, contendo sua placa, quilometragem, o tipo de veículo, o CPF do dono e um mapa com todas as ordens de serviço geradas para o veículo em específico.

- **Client:**

  Essa classe armazena os dados de um cliente da oficina, armazenando seu nome, CPF e um mapa que contém os veículos do dessa pessoa.

- **Oficina:**

  Essa é a classe mais geral e possui relação de composição com Admin, Vendor, Mechanic, Client, ServiceOrder, Service e AutoPart. Ela possui um Admin, e possui oito mapas que armazenam os vendedores da loja, os mecânicos, os clientes, os serviços pendentes, os serviços confirmados, os serviços concluídos, os serviços e as peças disponíveis na oficina.

● Decisões do projeto e recursos utilizados

Todas as decisões do projeto foram feitas após a criação do UML e a medida que desafios surgiam, como por exemplo a criação de uma estrutura geral para a oficina, que teria como atributo os usuários e também clientes e os dados.

Para tais decisões levamos em conta a facilidade de entendimento do código e relação entre as estruturas e facilidade de implementação, para evitar situações inesperadas.

- Recursos Utilizados:
  - Mapas ordenados: escolhemos por utilizar a estrutura de mapas ordenados por facilitar os métodos de acesso. Além disso, por possuir seus métodos com complexidade O(log n) para todas as operações, isto é, pesquisa, inserção remoção de conteúdos.

● Compilação e execução

- Para executar o programa em um sistema operacional Linux, basta executar o arquivo '***compile_and_run.sh***' *através do comando de terminal '****sh compile_and_run.sh****'. Caso prefira compilar primeiro e executá-lo depois, execute os comandos no terminal:*
  - ***g++ *.cpp -o main*** para compilar
  - ***./main*** para executar

- Para compilar o programa, em caso de um sistema operacional Windows basta executar o arquivo '***compile_and_run.bat***' ou executar os comandos:
  - ***g++ *.cpp -o main.exe*** para compilar
  - ***main.exe*** para executar