



**UNIVERSIDADE FEDERAL DE OURO PRETO**  
**Faculdade de Ciência da Computação**

**Trabalho Prático 02 POO**

**Daniel Monteiro Valério**  
**Gabriel Mace dos Santos Ferreira**  
**Marcus Vinicius Souza Fernandes**

Outubro de 2020

## **Trabalho Prático 02**

Daniel Monteiro Valério  
Gabriel Mace dos Santos Ferreira  
Marcus Vinícius Souza Fernandes

Trabalho Prático do Curso de Ciência da  
Computação da Universidade Federal de Ouro Preto.

## INTRODUÇÃO

Inicialmente foram idealizadas as classes para o funcionamento do sistema de operações de uma oficina mecânica, foi feita uma análise externa-interna, ou seja, foram identificando os agentes específicos (Administrador, Mecânico, Vendedor, Cliente e Carro) com seus respectivos elementos (Item, mão de obra, serviço e ordem), para assim identificar os elementos em comum e dessa forma criar as classes bases das quais as demais iriam herdar elementos. Todo esse processo se tornou mais prático e completo, uma vez que nos situamos com base no primeiro trabalho prático, com o intuito de o aprimorar.

## DESCRIÇÃO DA ARQUITETURA E IMPLEMENTAÇÃO

O grupo construiu o código utilizando o programa NetBeans de forma a simular o software de uma oficina, dessa forma foram utilizados pacotes para apresentar diferentes menus para o usuário de acordo com sua função interna na loja (Administrador, Mecânico, Vendedor), possibilitando assim, uma melhor modularização do código e tornando os menus mais específicos para o usuário logado no momento.

- Pessoa: A classe base para todos os agentes no código, responsável por armazenar os dados pessoais e passá-los para seus herdeiros.
- Cliente: A necessária para solicitar uma ordem, possui uma lista de carros, visto que um cliente pode possuir mais de um carro.
- Carro: A classe carro contém três variáveis string para armazenar o modelo, placa e cor do veículo, respectivamente, ela também possui uma variável int para armazenar seu id, visto que se encontrar em uma lista, portanto são necessários identificadores únicos. Além disso contém uma lista de ordens para armazenar as diversas ordens realizadas no veículo, funcionando como um histórico.
- Funcionário: A classe base de todos os trabalhadores da oficina, responsável por herdar os dados pessoais da classe Pessoa e criar os itens básicos para o cadastro de um funcionário no sistema da oficina.
- Admin: Classe responsável por gerir os demais trabalhadores, ele deve armazenar todos os funcionários da oficina, sendo separados por funcionalidade (Mecânico e Vendedor), ademais possui métodos para adicionar um funcionário, alterá-lo ou excluí-lo.
- Vendedor: Classe responsável pelo gerenciamento das ordens, clientes e seus respectivos carros, ela possui os dados herdados da classe funcionário, no entanto seu campo user é único sendo formado pela palavra “vende” seguido do Id único do vendedor. A classe vendedor possui métodos para criação, alteração e exclusão de clientes e ordens.
- Mecânico: A classe responsável por adicionar os serviços realizados as ordens que possuam aprovação do cliente e ainda não tenham sido concluídas. O Mecânico deve selecionar uma ordem para um n número de serviços realizados, acrescentando a mão de obra utilizada e custo, bem como os itens usados. De forma análoga ao vendedor seu user é único contendo o prefixo “mecan” seguido do Id do mecânico.

- Ordem: A ordem é criada pelo vendedor, após a consulta com o cliente, ela contém strings que irão definir seu tipo(Orçamento ou não), a motivação para ordem , uma descrição do problema e o cpf do cliente que solicitou a ordem, bem como uma variável double para armazenar o custo dessa, duas variáveis bool definir se a ordem foi concluída ou o cliente a aprovou. Finalmente ela contém uma lista de serviços, visto que é uma mesma ordem podem ser solicitados diferentes serviços(Troca de pneus e pintura do carro).
- Serviço: O serviço deve conter uma variável mão de obra para armazenar o custo da mão de obra utilizada, além disso contém uma lista de itens, visto que um mesmo serviço pode utilizar diversos itens, a classe contém uma variável int para armazenar seu código, caso seja necessário retirar um serviço da lista contida em Ordens.
- Mão de Obra: Classe utilizada para compor os campos da variável serviço, possui uma variável string para armazenar o seu “nome” e uma variável double para armazenar seu custo.
- Item: Classe utilizada para compor um dos campos da variável serviço, possui uma variável string para nomear o item utilizado, duas variáveis int para armazenar a quantidade de itens utilizados bem como o código do item respectivamente, além disso contém uma variável double para armazenar o valor total dos itens.

## DIAGRAMA UML

O diagrama UML foi criado na plataforma Lucidchart, um website com extensões especializadas. Para uma análise precisa, com uma boa qualidade de leitura, inserimos um arquivo pdf junto ao arquivo do relatório para que assim seja possível realizar este passo com uma maior cautela.

## DECISÕES DE PROJETO

As decisões para o trabalho prático tangente ao software da oficina mecânica foram tomadas em totalidade de forma conjunta e democrática. Abordamos a estratégia de formalizar o escopo do projeto realizando o mapeamento das camadas externas e bases inicialmente, com elas bem definidas e compreendidas partimos para uma análise mais específica, caminhando internamente nas ramificações, tipos, relações e propriedades.

Os pacotes e classes foram baseados nos principais agentes identificados nas instruções oferecidas pelo professor, a seguir foram construídos os itens referenciados e por conseguinte seus derivados. A criação da classe ordem provou-se desafiadora, pois poderiam haver diferentes serviços realizados em uma mesma ordem, portanto foi criada uma classe serviço que armazenaria o custo de mão de obra envolvido com tal ação e os itens utilizados, ademais essa classe é inserida na ordem, por meio de listas, o que nos permitiu solucionar o problema previamente exposto.

As interfaces gráficas foram desenvolvidas de forma igualitária, visando com que todos praticassem e absorvesse ao máximo o conteúdo de Swing. As aulas mediadas por Grazielle foram essenciais para fixação do conteúdo.

As tarefas foram bem divididas e executadas seguindo um modelo de metodologia ágil, trabalhávamos de forma paralela em funcionalidades em comum, concluindo uma seção por vez e considerando como concluída após compreendida, integrada e testada.

## RECURSOS DE LINGUAGEM

A escolha dos recursos utilizados envolveu uma cautelosa ponderação, visto que sua integração ao código existente deveria torná-lo mais funcional, compreensivo e compacto. Pensando nisso foram utilizados as funcionalidades:

- **Bibliotecas:**

***string** : A biblioteca string foi utilizada para armazenar os dados de nomeação das diferentes variáveis, visando suprir a necessidade de um vetor de caracteres, no entanto sem a necessidade da definição prévia do número mínimo de caracteres .*

***arrayList** : A biblioteca ArrayList é uma implementação da interface List que utiliza um vetor para armazenar elementos. Uma vez que vetores tem tamanho fixo em Java, a classe ArrayList se encarrega de criar um novo vetor (internamente) com um tamanho maior e copiar seus elementos correntes para esse novo vetor sempre que for necessário.*

***list** : A biblioteca list foi utilizada com o intuito de armazenar grandes quantidades de estruturas de dados, visto que algumas classes são quase exclusivamente compostas por diferentes classes, por vezes sendo necessário armazenar um número dinâmico dessas, por exemplo a previamente citada classe Ordem.*

- **Externos:**

***javax.swing:** O Swing é um framework que disponibiliza um conjunto de elementos gráficos para ser utilizado na plataforma Java. O Swing é compatível com o Abstract Window Toolkit (AWT), mas trabalha de forma totalmente diferente. A API Swing, diferente do AWT, não delega a tarefa de renderização ao sistema operacional, ele renderiza os elementos por conta própria.*

- **Boas Práticas:**

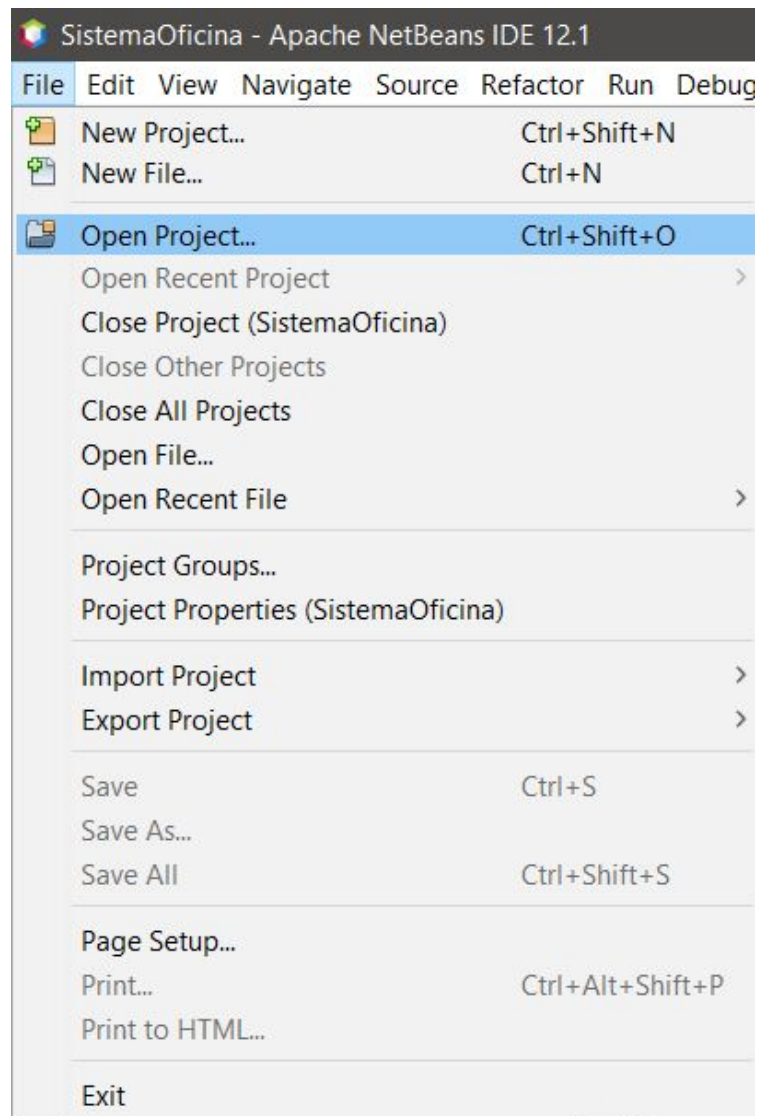
***Indentação:** Uma boa indentação é essencial para garantir uma boa legibilidade no código.*

***Comentários:** Comentários são de extrema importância para assegurar a compressão das funções e passos desenvolvidos.*

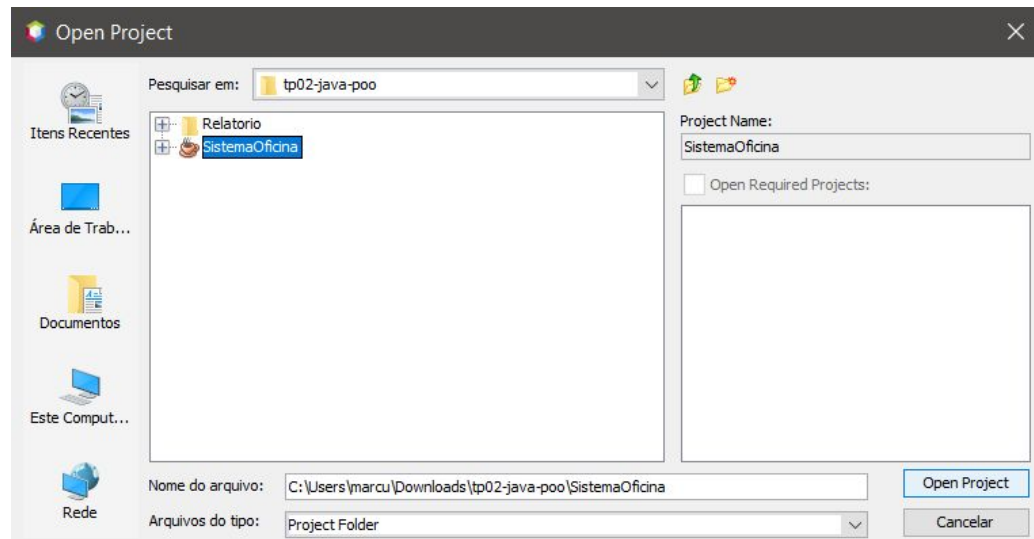
***Nomenclatura:** Seguir padrões de nomes de variáveis e funções também garantem uma excelente compressão e legibilidade do código.*

## INSTRUÇÕES DE COMPILAÇÃO

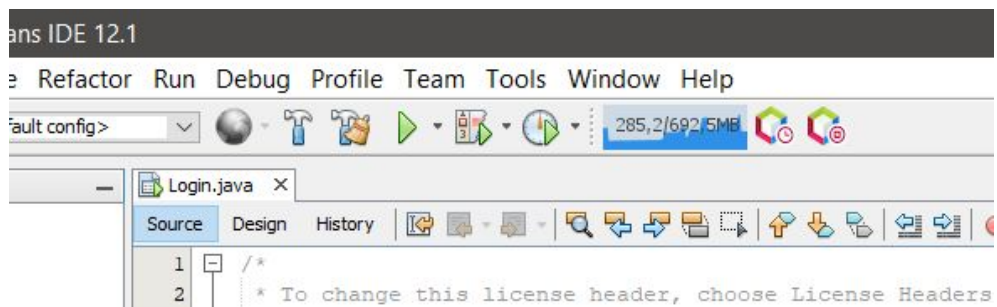
- 1) Abra o software NetBeans e seguindo o exemplo abaixo selecione a opção “Open Project”.



2) Em seguida selecione o projeto “Sistema Oficina” e clique no botão ressaltado “Open Project”.

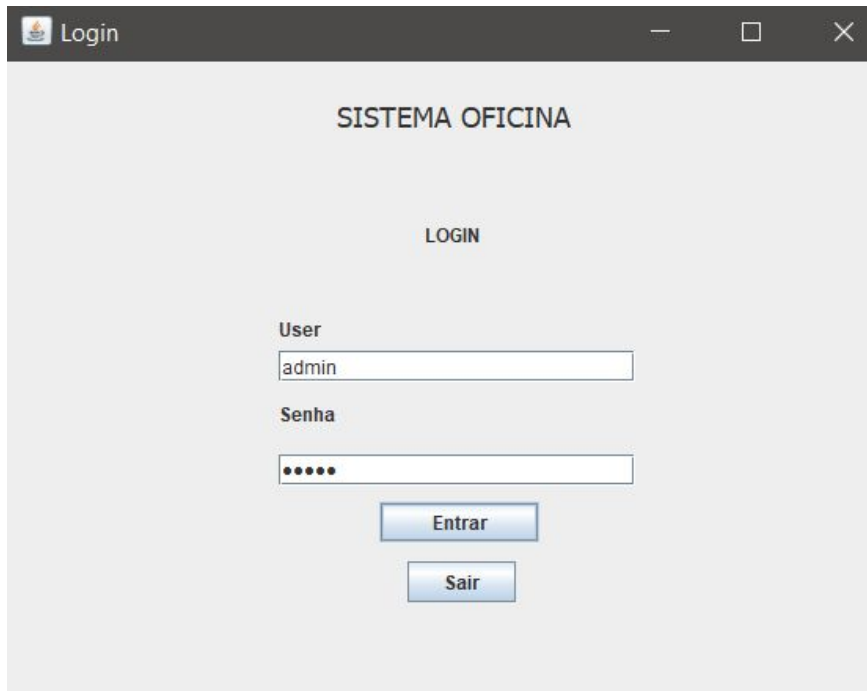


3) Com o projeto aberto podemos o compilar clicando no ícone “run”.



4) Você irá se deparar com a tela inicial do sistema (Login), como ainda não possuímos funcionários cadastrados, torna-se necessário efetuar login no sistema com os dados do administrador criado estaticamente.

- User: *admin*
- Senha: *admin*



The screenshot shows a web application window titled "Login". The window's title bar includes a small icon and the text "Login", along with standard minimize, maximize, and close buttons. The main content area has a light gray background. At the top, the text "SISTEMA OFICINA" is displayed. Below it, the word "LOGIN" is centered. There are two input fields: the first is labeled "User" and contains the text "admin"; the second is labeled "Senha" and contains five dots, indicating a password field. Below the input fields are two buttons: "Entrar" (Login) and "Sair" (Logout).

5) Assim, você possui acesso a todas as funções do administrador, poderá criar/editar/excluir e visualizar novos funcionários e em seguida realizar login com os dados cadastrados, permitindo que execute as respectivas funções que cada um deles possui a devida autonomia.