Trabalho prático Swing e JDBC.
Data da entrega: Ver moodle
Data das defesas: Ver moodle
Equipes: Máximo 4 pessoas.

O objetivo é refatorar o software produzido em LPOO I, incluindo o uso de banco de dados com JDBC e DAO e também o uso do conceito do padrão MVC (Model View Controller). Para aqueles que não fizeram LPOOI no semestre passado será necessário construir o programa do zero.

Uma empresa que loca veículos (automóveis, vans e motos) resolveu criar um sistema para gerenciar o ciclo de vida de seus veículos (da compra, passando pelas locações, até a venda do veículo). O novo sistema precisa ser em Java com interface SWING e seguindo o paradigma orientado a objetos.

O sistema terá os seguintes requisitos:

1. Uma tela para manter (incluir, atualizar, excluir e listar) os clientes da empresa (Nome, sobre nome, RG, CPF, Endereço)
    a. Nesta tela deve ser possível listar todos os clientes (Use AbstractTableModel)
    b. Deve ser possível atualizar os dados de um cliente.
    c. Deve ser possível excluir um cliente que não possua veículos locados. Se houver uma tentativa de exclusão de clientes com veículos locados uma mensagem do sistema deve informar ao usuário que o cliente não pode ser excluído.
    d. Deve ser possível listar todos os clientes em uma tabela (Utilize AbstractTableModel)
2. Uma tela para incluir os veículos novos quando são comprados
    a. Deve-se utilizar herança para definir os veículos. Uma classe Veículo (abstrata) deve ser criada. A classe Veículo deve implementar a seguinte interface:

```
public interface VeiculoI {
    //Muda estado para LOCADO. Cria uma instância de Locacao e armazena no atributo
locacao. Chama o método getValorDiariaLocacao para calcular o valor da locação.
    public void locar(int dias, Calendar data, Cliente cliente);
    //Muda estado para VENDIDO e não pode mais ser alugado
    public void vender();
    //Muda estado para DISPONIVEL
    public void devolver();
    public Estado getEstado();
    public Marca getMarca();
    public Categoria getCategoria();
    public Locacao getLocacao();
    public String getPlaca();
    public int getAno();
    //Método que calcula um valor para venda. Utilizar o seguinte cálculo:
    //valorParaVenda = valorDeCompra - idadeVeiculoEmAnos*0,15*valorDeCompra
    //Se o resultado for menor do que 10% do valorDeCompra ou negative, então
    //        varlorParaVenda = valorDeCompra*0,1
    public double getValorParaVenda();
    //Método que será abstrato na classe Veiculo
    public double getValorDiariaLocacao();

}
```

    b. A classe veículo deve ter os seguintes atributos:
        i. Criar um construtor com todos os atributos como parâmetro.
        ii. marca – Crie uma classe de nome Marca e do tipo enum para definir este atributo. Pode assumir os seguintes valores (VW, GM, Fiat, Honda, Mercedes, etc)
        iii. estado - - Crie uma classe de nome Estado e do tipo enum para definir este atributo. Pode assumir os seguintes valores (NOVO, LOCADO, DISPONIVEL, VENDIDO)
        iv. locacao – Crie uma classe Locacao especificada no item 3. Este objeto será nulo quando o estado do veículo for diferente de LOCADO. O objeto será instanciado somente quando o veiculo for alugado (método alugar) e será atribuído nulo quando o veículo for devolvido (método devolverAluguel).
        v. categoria - Crie uma classe de nome Categoria e do tipo enum para definir este atributo. Pode assumir os seguintes valores (POPULAR, INTERMEDIARIO, LUXO). Este atributo írá definir o preço da diária de um aluguel.
        vi. valorDeCompra – double que representa o valor de compra do veículo
        vii. placa – String. String no formato XXX-0000, onde X é um alfabético de A a Z e 0 é um numério de 0 a 9.
        viii. ano – int. Inteiro representado o ano modelo do veículo.
        ix. Não crie métodos de get e set para cada atributo. Crie somente os métodos que estão descritos na interface VeiculoI
    c. A classe Automovel deve herdar de Veiculo e ter os seguintes atributos e métodos:
        i. Criar um construtor com todos os atributos como parâmetro.

ii. modelo - Crie uma classe de nome ModeloAutomovel do tipo enum para definir este atributo. Pode assumir os seguintes valores (Gol, Celta, Palio, etc)
iii. ModeloAutomovel getModelo() – Método que retorna o valor do atributo modelo.
iv. double getValorDiariaLocacao() – Método que retorna o valor da diária de uma locação de automóvel. Deverá respeitar a seguinte tabela

Categoria	Valor da Diária
POPULAR	R$100,00
INTERMEDIARIO	R$300,00
LUXO	R$450,00

d. A classe Motocicleta deve herdar de Veiculo e ter os seguintes atributos:
i. Criar um construtor com todos os atributos como parâmetro.
ii. modelo - Crie uma classe de nome ModeloMotocicleta do tipo enum para definir este atributo. Pode assumir os seguintes valores (CG 125, CBR 500, etc) ,
iii. ModeloMotocicleta getModelo() – Método que retorna o valor do atributo modelo.
iv. double getValorDiariaLocacao() – Método que retorna o valor da diária de uma locação de automóvel. Deverá respeitar a seguinte tabela

Categoria	Valor da Diária
POPULAR	R$70,00
INTERMEDIARIO	R$200,00
LUXO	R$350,00

e. A classe Van deve herdar de Veiculo e ter os seguintes atributos:
i. Criar um construtor com todos os atributos como parâmetro.
ii. modelo - Crie uma classe de nome ModeloVan do tipo enum para definir este atributo. Pode assumir os seguintes valores (Kombi, Sprinter, etc)
iii. ModeloVan getModelo() – Método que retorna o valor do atributo modelo.
iv. double getValorDiariaLocacao() – Método que retorna o valor da diária de uma locação de automóvel. Deverá respeitar a seguinte tabela

Categoria	Valor da Diária
POPULAR	R$200,00
INTERMEDIARIO	R$400,00
LUXO	R$600,00

f. A classe Locacao deve ter os seguintes atributos:
i. Criar um construtor com todos os atributos como parâmetro.
ii. int dias – numero de dias da locação
iii. double valor – Valor da locação em reais
iv. Calendar data – Data da locação
v. Cliente cliente – Cliente da locação
vi. double getValor() – Retorna o valor da locação.
vii. Calendar getData() – Retorna a data da locação
viii. Cliente getCliente() – Retorna o cliente da locação

g. A tela para incluir veículos deve mapear os seguintes atributos e componentes:

Atributo	Componente
Marca	ComboBox
Estado	ComboBox
Categoria	ComboBox
Modelo	ComboBox (dependendo do tipo do veículo a Combo mostrará os valores para automóveis, motocicletas ou vans.
valorDeCompra	TextField com máscara para valores monetários
Placa	TextField com máscara para validar uma placa (XXX-0000)

h. Um botão para incluir o veículo deve estar disponível. Quando clicado o veículo deve ser instanciado e gravado na base de dados por seu respectivo DAO.

3. Uma tela para locar os veículos. Para simplificar, um cliente poderá locar um veículo por vez.
a. Neste tela deve ser possível selecionar um cliente por nome, sobrenome ou cpf
b. Deve ser possível filtrar os veículos por tipo (Automóvel, Van ou Motocicleta), por marca e/ou por categoria.
c. Depois de filtrado, deve aparecer uma tabela com os veículos que estão disponíveis para locação (Estado DISPONIVEL), com as seguintes colunas: Placa, Marca, Modelo, Ano, Preço da diária. O ano deve aparecer com 4 dígitos, o preço deve aparecer com a formatação R$XXX,XX (onde X é um numérico de 0 a 9. Utilize AbstractTableModel para isso.
d. Uma TextField deve constar nesta tela para o cliente incluir o número de dias da locação.
e. A data da locação deve ser inserida pelo usuário por meio de uma TextField ou outro componente de data.
f. Para locar o veículo, o usuário deve clicar em um veículo na tabela e depois clicar em um botão Locar.
g. Depois de clicar em Locar o sistema deve acionar o método locar do veículo instanciado (Descrição na interface VeiculoI). O veiculo deve ser atualizado na base de dados, bem como o objeto Locacao. Este veículo não deve mais aparecer na tabela para ser locado.

4. Uma tela para devolver os veículos.
    a. Nesta tela deve ser possível listar os veículos locados. Uma tabela com os Veículos locados deve aparecer nesta tela com as seguintes colunas (Nome do Cliente, Placa, Marca, Modelo, Ano, Data Locação, Preço Diária, Quantidade de dias locado e Valor Locação)
    b. O usuário poderá escolher uma linha da tabela e clicar em um botão “Devolver veículo” para devolver o veículo. O sistema deve acionar o método develver do veículo que coloca o objeto Locacao para null, apaga da base de dados e muda o estado do veículo para DISPONÌVEL.
5. Uma tela para vender os veículos.
    a. Deve ser possível filtrar os veículos por tipo (Automóvel, Van ou Motocicleta), por marca e/ou por categoria.
    b. Depois de filtrado, deve aparecer uma tabela com os veículos que estão disponíveis para venda (Estado DISPONIVEL), com as seguintes colunas: Placa, Marca, Modelo, Ano, Preço para venda. O ano deve aparecer com 4 dígitos, o preço deve aparecer com a formatação R$XXX,XX (onde X é um numérico de 0 a 9. Utilize AbstractTableModel para isso.
    c. Para vender o veículo, o usuário deve clicar em um veículo na tabela e depois clicar em um botão Vender.
    d. Depois de clicar em Vender o sistema deve acionar o método vender do veículo instanciado (Descrição na interface VeiculoI). O veiculo deve ser atualizado na base de dados. Este veículo não deve mais aparecer na tabela para ser locado.

O programa acima deve ser feito utilizando as seguintes tecnologias:

- Utilizar herança para definir as classes Veiculo, Automovel, Motocicleta e Van. Utilize classes abstratas quando necessário.
- Uitilize o modelo MVC (Model – View – Controller)
- Utilize JDBC para persistência dos objetos no banco de dados
- Utilizar polimorfismo para manipular os veiculos.
- Java Swing

Itens para serem entregues:

1. Diagrama de classes
2. Projeto na IDE Eclipse ou Netbeans com código fonte
3. Arquivo .jar executável

Avaliação:

- Material entregue
- Qualidade do software (bugs encontrados na defesa)
- Defesa do código e a nota será individual, considerando a defesa