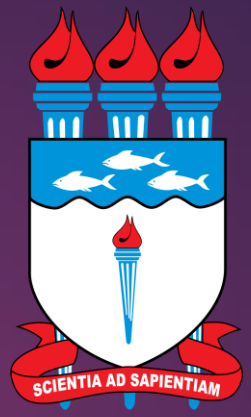
Sistema para Mercadinho



UNIVERSIDADE FEDERAL DE ALAGOAS

LABORATÓRIO DE PROGRAMAÇÃO 2

DOCENTE: RODOLFO CARNEIRO CAVALCANTE

DISCENTES: ANTONIO ANDRADE GOMES JÚNIOR

JONATHAS THIAGO CASSIANO

Contexto

- Desenvolvemos esse sistema, para um gerenciamento de um mercadinho/loja de pequeno porte em que, no sistema, a loja só é instanciada uma vez e há a presença de somente um estoque de produtos.
- Como todo mercadinho de pequeno porte há a adoção do famoso “prego” ou “fiado”, o sistema cuida disso, cadastrando todos os que os que devem, suas dividas e compras.
- Precisa de um refinamento ainda, vendo o pequeno tempo de entrega.

Diagramas para entendimento do Projeto

Diagrama de casos de uso ou de requisitos

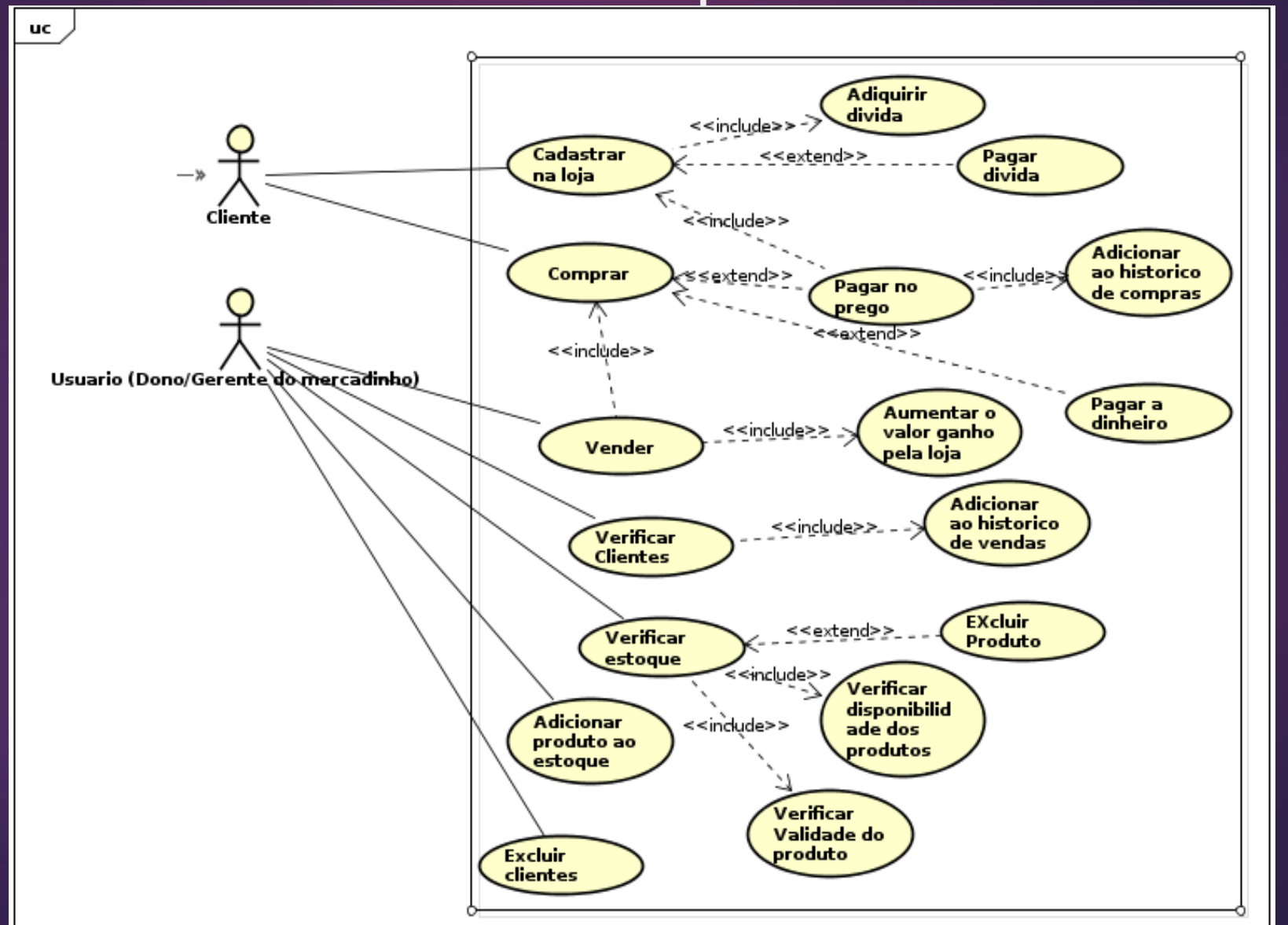
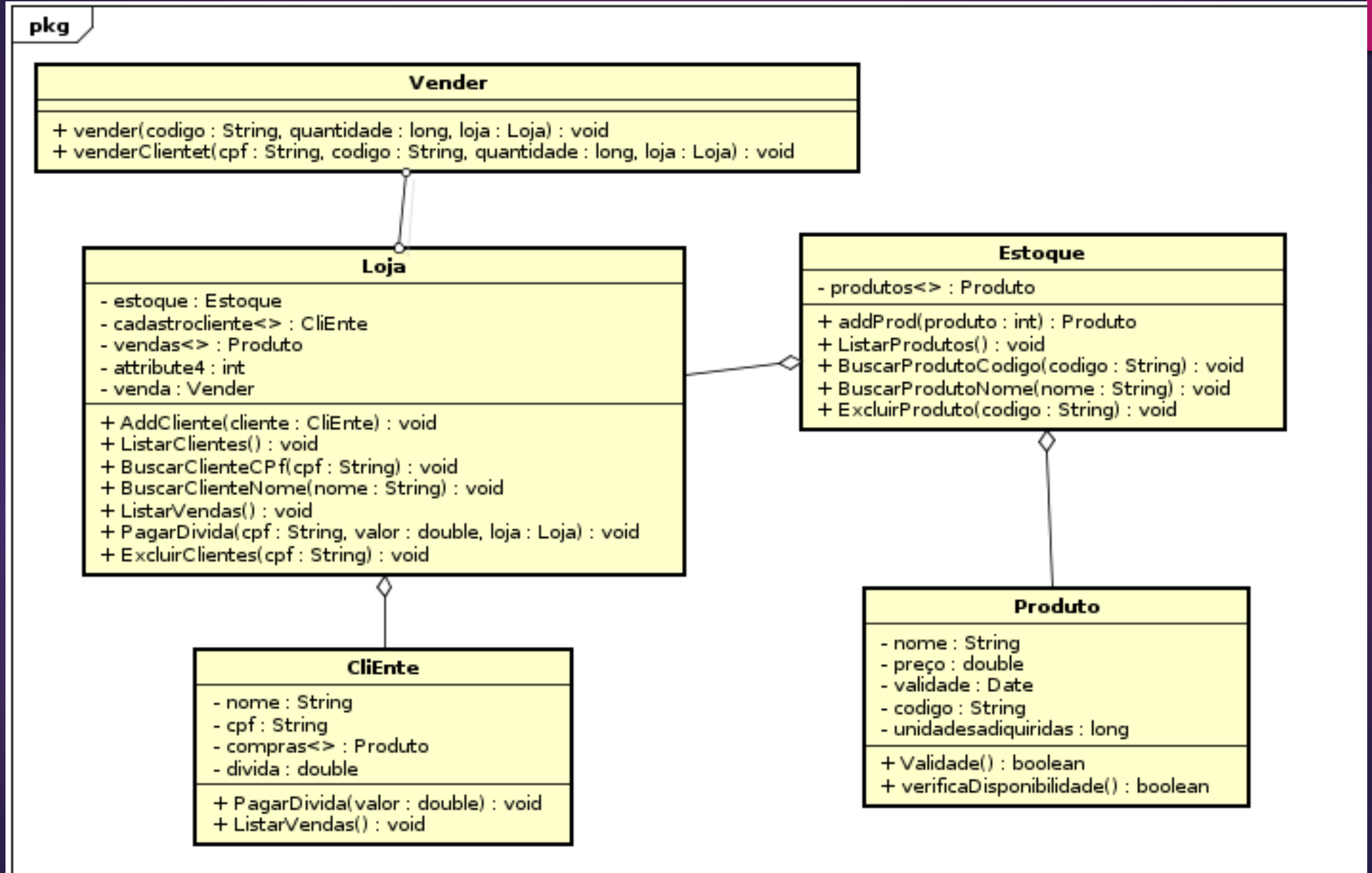


Diagrama de Classes



SINGLETON

Na classe Vender

PARA NÃO OCORRER ISSO

```
Vender um = new Vender();
Vender dois = new Vender();
um.vender("Abacaxi", 12, 1);
dois.vender("Abacaxi", 12, 1);
```

PROBLEMAS

- Dois objetos tentando vender ao mesmo tempo.
- Ocasionando problemas na fila de vendas.

Na classe Loja

PARA NÃO OCORRER ISSO

```
Loja a = new Loja();
Loja b = new Loja();
```

MODELO DA LOJA

- Só exista uma loja no sistema.
- Quando criar a loja, já criar um estoque, como colocado no código.

```java
public class Loja {

    private double montanteMensal; //Quanto a loja possui de dinheiro
    private Estoque estoque; //Para loja iniciar com um estoque.
    private ArrayList <Cliente> cadastrocliente; //Lista de clientes pertencentes a loja
    private ArrayList <Produto> vendas; //Criando lista de vendas
    private Vender venda; //Criando comando de vender.
    private static Loja singleton; //Singleton

    private Loja (){
        this.montanteMensal = 0.0; //Quanto a loja possui de dinheiro
        this.cadastrocliente = new ArrayList<Cliente>(); //Lista de clientes pertencentes a loja
        this.vendas = new ArrayList<>(); //Criando lista de vendas
        this.estoque = Estoque.getInstance(); //Para loja iniciar com um estoque.
        this.venda = Vender.getInstance(); //Criando  de vender.

    }

    public static synchronized Loja getInstance(){ //Método singleton
        if (singleton == null){
            singleton = new Loja();
        }
        return singleton;
    }

    public Vender getVenda() {
        return venda;
    }
```

Na classe Estoque

PARA NÃO OCORRER ISSO

```
Estoque a = new Estoque();
Estoque b = new Estoque();
```

Razões

- O intuito do programa é que só exista um estoque em um loja.
- Na hora de adicionar produto só ir para o estoque próprio da loja e não para outro.

```java
class Estoque{
    private ArrayList<Produto> produtos;
    private static Estoque singleton;
    private Estoque(){}
    public static synchronized Estoque getInstance(){
        if (singleton == null){
            singleton = new Estoque();
            singleton.produtos = new ArrayList<>();
        }
        return singleton;
    }

    public void addProd(Produto c){ //Adiciona produto ao estoque, e esse produto vai ser adicionado na loja.
        this.produtos.add(c); //Pegando referência do produto.
    }

    public ArrayList<Produto> getProdutos() {
        return produtos;
    }

    public void BuscarProdutoCodigo(String codigo){
        boolean verifica = false; // verificar se existe produto.
        for (int i = 0; i<produtos.size(); i++){
            if (produtos.get(i).getCodigo().equals(codigo)){
                verifica = true; // verificar se há produto no estoque para listar.
                break; // Quando achar pelo menos um produto.
            }
        }
        if (verifica == true){
```

CÓDIGOS PRINCIPAIS

Classe Loja

```java
public void ListarCliente (){
    if(this.cadastrocliente.size() > 0){
        System.out.println("cpf\t\t\tNome\t\t\tDivida");
        System.out.println("=============================================================================================");
        for (int i=0; i < cadastrocliente.size(); i++){
            //Dá pra fazer tudo em uma linha, mas não fica elegante.
            System.out.printf("%s", cadastrocliente.get(i).getCpf()); System.out.printf("\t\t\t%s", cadastrocliente.get(i).getNome());
            System.out.printf("\t\t\t%.2f\n", cadastrocliente.get(i).getDivida());
        }
        System.out.println("=============================================================================================");
    }else{
        System.out.println("Nao ha clientes cadastrados!!!");
    }
}
public void BuscarClienteCpf(String cpf){
    boolean verifica = false;
    for (int i = 0; i < this.cadastrocliente.size(); i++){
        if (this.cadastrocliente.get(i).getCpf().equals(cpf)){
            System.out.println("cpf\t\t\tNome\t\tDivida");
            System.out.println("=============================================================================================");
            System.out.printf("%s", cadastrocliente.get(i).getCpf());
            System.out.printf("\t\t\t%s", cadastrocliente.get(i).getNome());
            System.out.printf("\t\t%.2f\n", cadastrocliente.get(i).getDivida());
            System.out.println("=============================================================================================");
            verifica = true;
            }
    }
    if (verifica = false){
        System.out.println("Não existe clientes cadastros com esse CPF");
    }
}
```

Classe Cliente

```java
public void PagarDivida(double valor, Loja b){ //Criado referência para alterar montante mensal da loja com o que pagar.
    if (valor >= 0){ //Verifica se valor não é negativo
        if (valor > getDivida()){ //Dinheiro maior que divida, então possui troco.
            b.setMontanteMensal(b.getMontanteMensal() + getDivida()); //Adicionando dinheiro pago ao motante mensal.
            double troco = valor - getDivida();
            setDivida(0); //Se valor é maior que divida, logo sempre vai ser zero a divida
            System.out.printf("O troco é de %.2f\n"
                    +"Divida paga com sucesso\n", troco);
        }else if (valor <= getDivida()){ //Verifica se valor é menor que a dívida
            b.setMontanteMensal(b.getMontanteMensal() + valor); //Adicionando dinheiro pago ao motante mensal.
            setDivida(getDivida() - valor);
            System.out.printf("Sua divida continua em %.2f\n", getDivida());
        }
    }else{ //Valor negativo, errado.
        System.out.println("Valor incorreto.");
    }
}
public void ListarVendas(){ //lista o historico de compras do cliente
    if (this.compras.size() > 0){
        System.out.println("Codigo\t\t\tNome\t\tQuantidade\t\tPreço\t\tValidade");
        System.out.println("=============================================================================================");
        for (int i=0; i < this.compras.size(); i++){
            //Dá pra fazer tudo em uma linha, mas não fica elegante.
            System.out.printf("%s", this.compras.get(i).getCodigo());
            System.out.printf("\t\t\t%s", this.compras.get(i).getNome());
            System.out.printf("\t\t%d", this.compras.get(i).getUnidadesAdiquiridas());
            System.out.printf("\t\t\t%.2f", this.compras.get(i).getPreco());
            System.out.printf("\t\t%s\n", this.compras.get(i).getValidade2());


        }
        System.out.println("=============================================================================================");
        System.out.printf("Dívida: R$ %.2f\n", getDivida());
    }else{
        System.out.println("Não foi realizada nenhuma compra");
    }
}
```

Classe Vender

```java
public void vender(String codigo, long quantidade, Loja b) throws ParseException{
    boolean verifica = false; // Variavel criada para verificar possibilidades.
    boolean verifica2 = false;
    for (int i = 0; i < b.getEstoque().getProdutos().size(); i++){
        if (b.getEstoque().getProdutos().get(i).getCodigo().equals(codigo)){
            long qtd = b.getEstoque().getProdutos().get(i).getUnidadesAdiquiridas() - quantidade; // Quantidade do estoque menos as unidades compradas
            verifica = true;
            if (qtd > 0){
                b.getVendas().add(b.getEstoque().getProdutos().get(i)); //Adicionando produto as vendas.
                b.getEstoque().getProdutos().get(i).setUnidadesAdiquiridas(qtd); //Alterando a quantidade do produto.
                int index = indexVenda(codigo, b); // Pegando posição do produto.
                b.getVendas().get(index).setUnidadesAdiquiridas(quantidade); //Alterando para quantidades vendidas.
                verifica2 = true;
                double preco = b.getVendas().get(index).getPreco() * b.getVendas().get(index).getUnidadesAdiquiridas(); // Preço do produto.
                b.setMontanteMensal(b.getMontanteMensal() + preco);
            }else if (qtd == 0){ // Se as unidades menos a quantidade forem igual a zero, ou seja, acabou o produto.
                b.getVendas().add(b.getEstoque().getProdutos().get(i)); //Adicionando produto as vendas
                int index = indexVenda(codigo, b); // Pegando posição do produto.
                b.getVendas().get(index).setUnidadesAdiquiridas(quantidade); //Alterando a quantidade do produto.
                b.getEstoque().getProdutos().remove(i); //Removendo o produto
                verifica2 = true;
                double preco = b.getVendas().get(index).getPreco() * b.getVendas().get(index).getUnidadesAdiquiridas(); // Preço do produto.
                b.setMontanteMensal(b.getMontanteMensal() + preco);
            }
        }
    }
    //Central de Verificações.
    if (verifica == false){
        System.out.println("Código errado.");
    }else if(verifica == true && verifica2 == true){
        System.out.println("Compra realizada com Sucesso.");
    }else if(verifica == true && verifica2 == false){
        System.out.println("Não foi possivel realizar a compra.");
    }
}
```


That's all Folks!