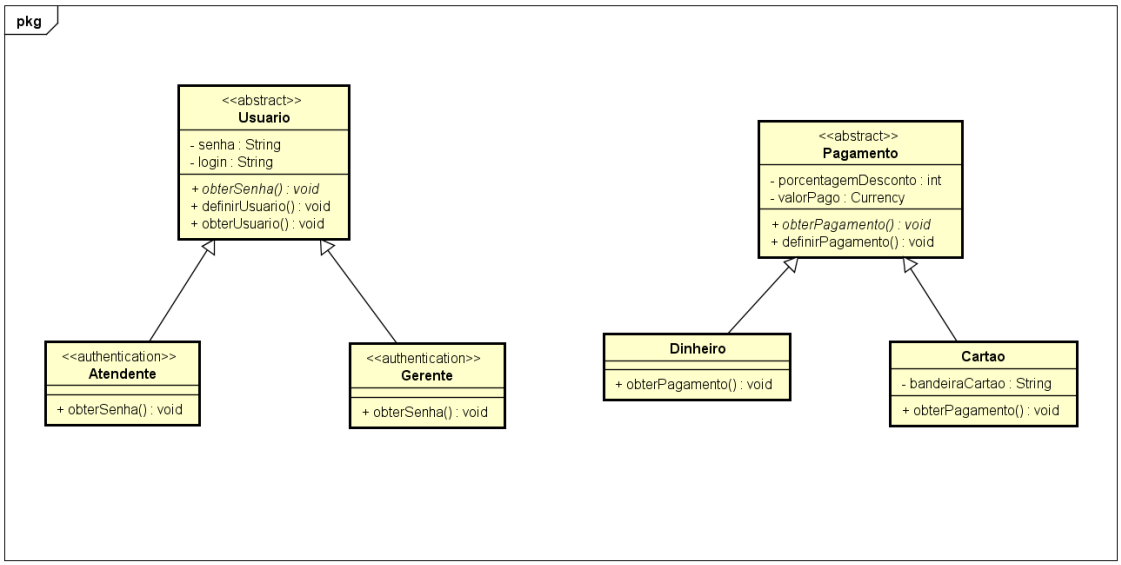
ANÁLISE E DESENVOLVIMENTO DE SISTEMAS – MA4

Engenharia de Software III – Prof. Msc. Wilson Vendramel

Alicia Schmoeller	RA: 1680481521004
Axl Brandon	RA: 141681024
Douglas Garcia	
Guilherme Augusto Ponte	RA: 141681251
Gustavo Antunes da Silva	RA: 1680481512009
Joice Elena Benedini Calister	RA: 1680481511024
Julia Ferez	RA: 1680481511026
Leonardo Costa	RA: 1680481512015
Matheus P. G.Braga	RA: 1680481512040

São Caetano do Sul, 2016

1-



2-

Não, nenhuma das gen/specs violam o princípio de Liskov, pois todo atendente ou gerente é um usuário, assim como o cartão e o dinheiro são formas de pagamento.

3- public class Usuario

```
  {
     public string Senha { get; set; }
     public string Login { get; set; }

     public void ObterSenha()
     {

     }

     public void DefinirUsuario()
     {

     }

     public void ObterUsuario()
     {

     }
  }

 public class Atendente : Usuario
   {
     public int Comissao { get; set; }

     public override void ObterSenha()
     {

     }

     public void DefinirComissao(int comissao)
```

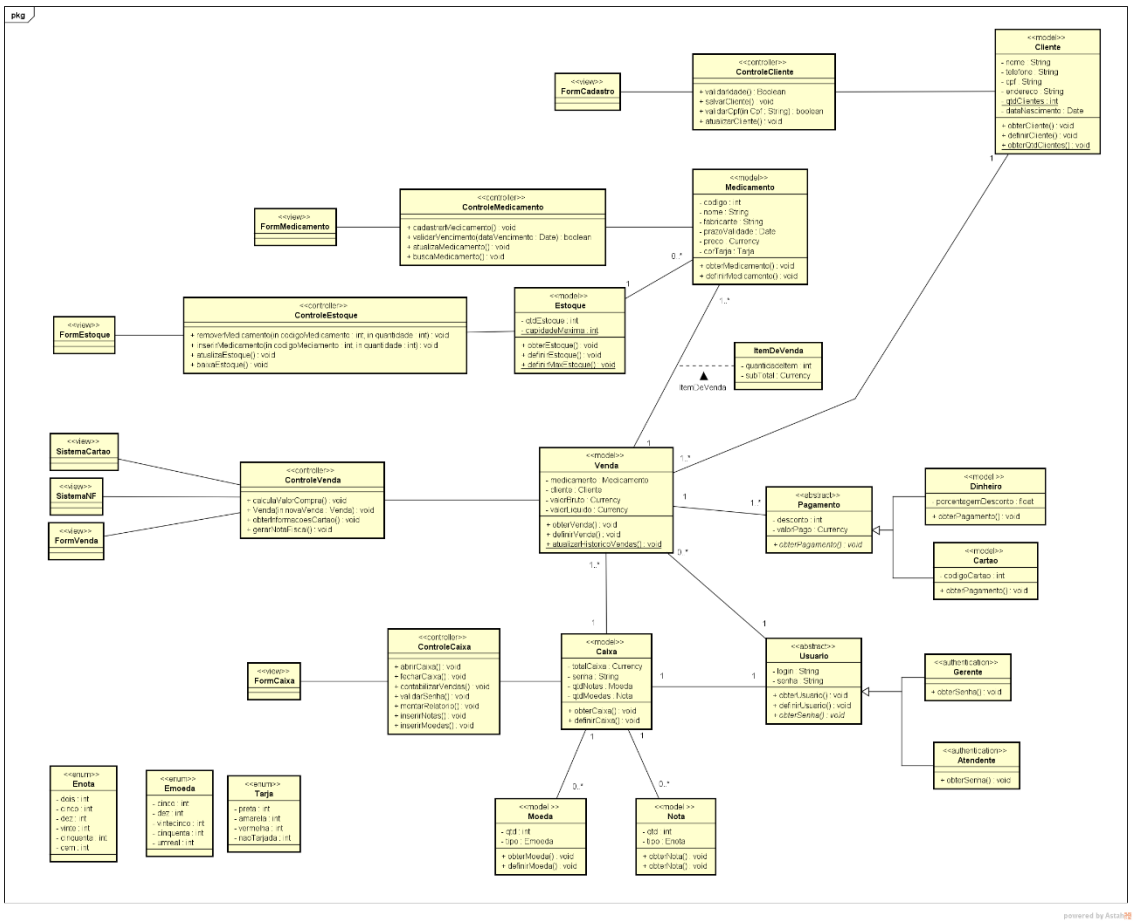
```
        {
            this.Comissao = comissao;
        }

        public int ObterComissao()
        {
            return this.Comissao;
        }
    }

 public class Gerente : Usuario
    {
        public int DescontoAtribuido { get; set; }

        public override void ObterSenha()
        {

        }

        public void DefinirDesconto(int desconto)
        {
            this.DescontoAtribuido = desconto;
        }

        public int ObterDesconto()
        {
            return this.DescontoAtribuido;
        }
    }
public abstract class FormaPagamento
    {
        public int CodigoPagamento { get; set; }

        public void ObterPagamento()
        {

        }

        public void DefinirPagamento()
        {

        }
    }

public class Dinheiro : FormaPagamento
    {
        public int PorcentagemDesconto { get; set; }

        public override void ObterPagamento()
        {

        }
    }

 public class Cartao : FormaPagamento
    {
        public string BandeiraCartao { get; set; }

        public override void ObterPagamento()
        {
```

```
    }
  }
```

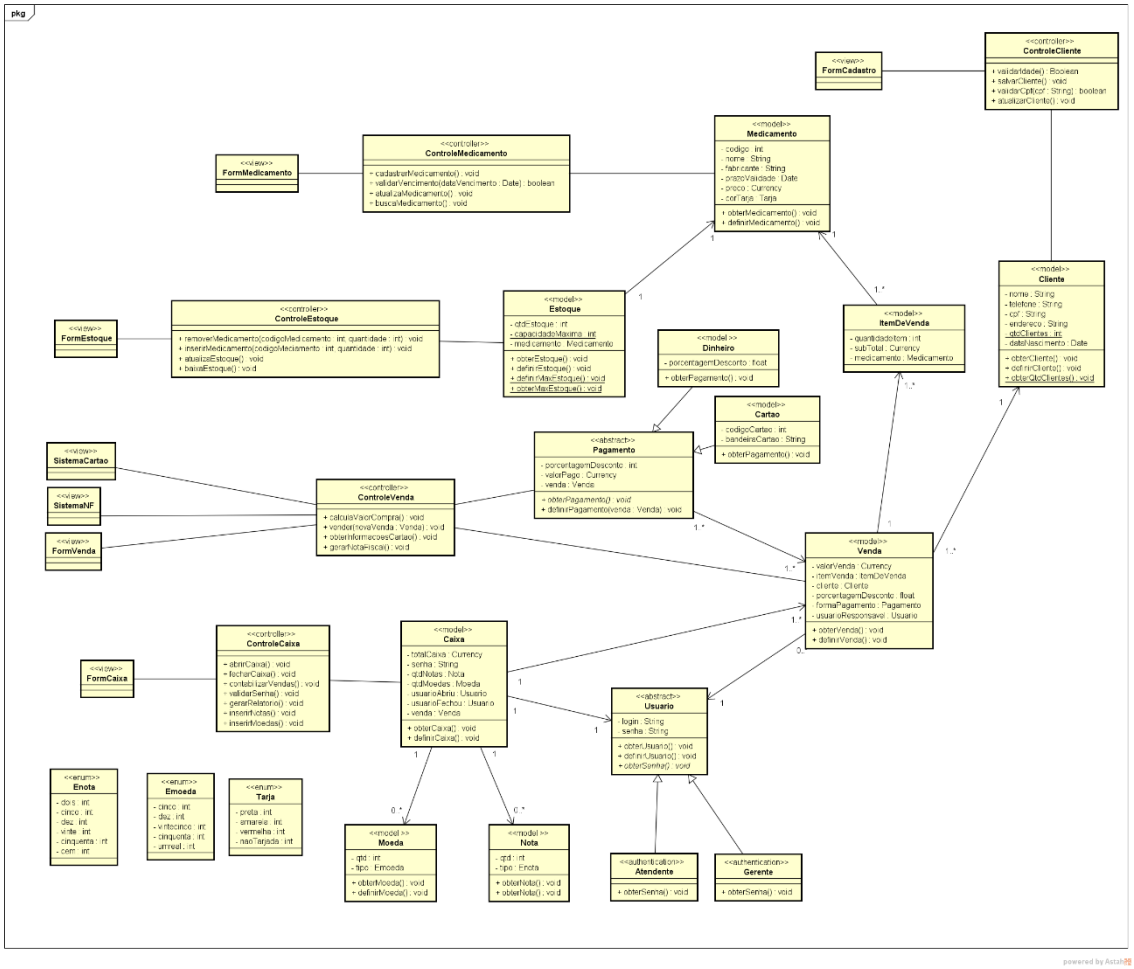
4-

A gen/spec Usuário não possui Classificação Dinâmica, pois um usuário sempre será do tipo gerente ou do tipo atendente, mas nunca os dois. Porém, a gen/spec FormaPagamento possui Classificação Dinâmica, pois um pagamento pode ser feito em dinheiro e cartão simultaneamente.

5-



6-



Vantagem: Aumento no desempenho.

Desvantagem: Queda no encapsulamento e aumento no acoplamento.

7-

```
public class Atendente : Usuario
{
        public float comissao { get; set; }
        public void obterSenha() {} {}
}

public class Caixa
{
        public Currency totalCaixa { get; set; }
        public string senha { get; set; }
```

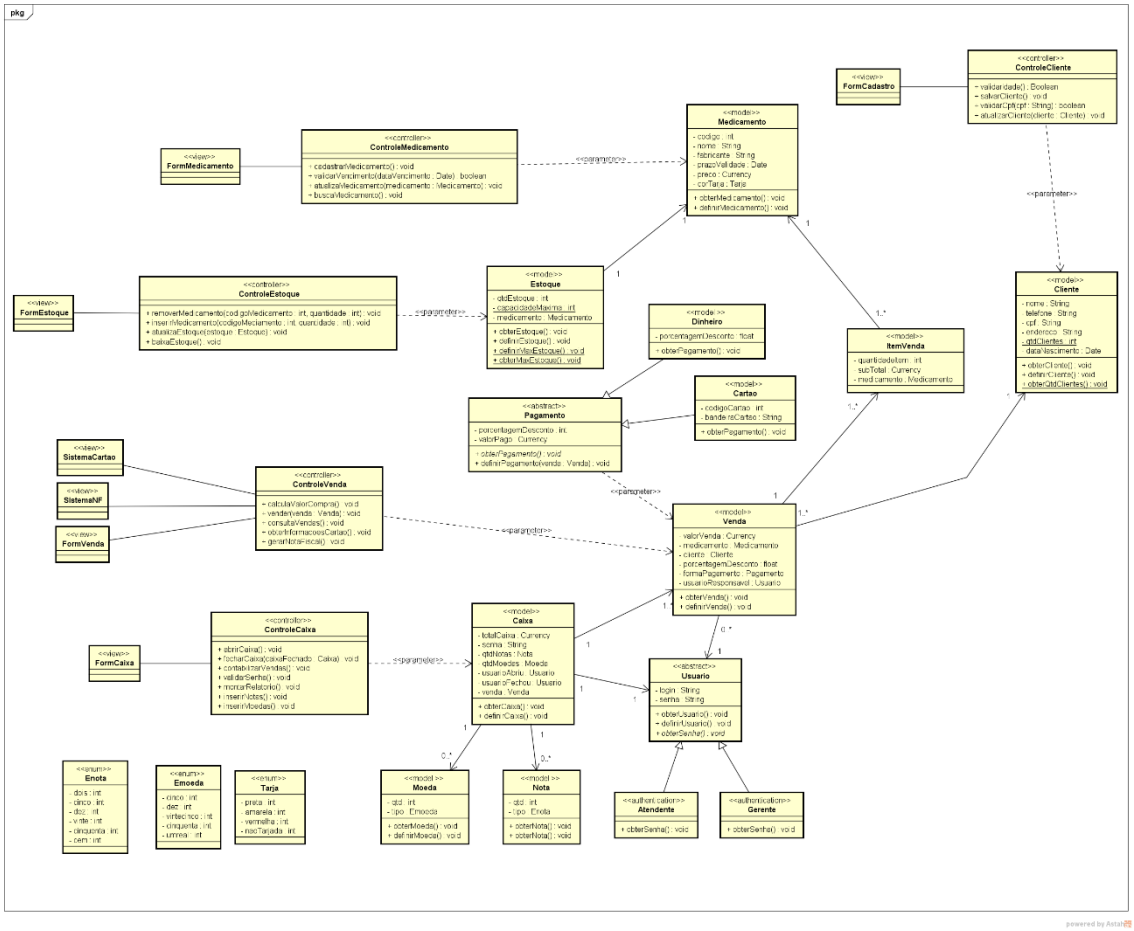
```
        public Nota qtdNotas { get; set; }
        public Moeda qtdMoedas { get; set; }
        public Usuario usuarioAbriu { get; set; }
        public Usuario usuarioFechou { get; set; }
        public ICollection<Nota> nota { get; set; }
        public ICollection<Venda> venda { get; set; }
        public ICollection<Moeda> moeda { get; set; }
        public void obterCaixa() {}
        public void definirCaixa() {}
}

public class Cartao : Pagamento
{
        public int codigoCartao { get; set; }
        public string bandeiraCartao { get; set; }
        public void obterPagamento() {}
}

public class Cliente
{
        public string nome { get; set; }
        public string telefone { get; set; }
        public string cpf { get; set; }
        public string endereco { get; set; }
        public static int qtdClientes { get; set; }
        public Date dataNascimento { get; set; }
        public void obterCliente() {}
        public void definirCliente() {}
        public static void obterQtdClientes() {}
}

public class Dinheiro : Pagamento
```

```
{
        public float porcentagemDesconto { get; set; }
        public void obterPagamento() {}
}

public class Estoque
{
        public int qtdEstoque { get; set; }
        public static int capidadeMaxima { get; set; }
        public ICollection<Medicamento> medicamento { get; set; }
        public void obterEstoque() {}
        public void definirEstoque() {}
        public static void definirMaxEstoque() {}
}

public class Gerente : Usuario
{
        public int descontoAtribuido { get; set; }
        public void atribuirDesconto() {}
        public void obterSenha() {}
}

public class Medicamento
{
        public int codigo { get; set; }
        public string nome { get; set; }
        public string fabricante { get; set; }
        public Date prazoValidade { get; set; }
        public Currency preco { get; set; }
        public Tarja corTarja { get; set; }

        public Estoque estoque { get; set; }
```

```
        public void obterMedicamento() {}
        public void definirMedicamento() {}
}

public class Moeda
{
        public int qtd { get; set; }
        public Emoeda tipo { get; set; }
        public Caixa caixa { get; set; }
        public void obterMoeda() {}
        public void definirMoeda() {}
}

public class Nota
{
        public int qtd { get; set; }
        public Enota tipo { get; set; }
        public void obterNota() {}
        public void obterNota() {}
}

public class Pagamento
{
        public int porcentagemDesconto { get; set; }
        public Currency valorPago { get; set; }
        public abstract void obterPagamento();
        public void definirPagamento(Venda venda)
}

public class Usuario
{
        public string login { get; set; }
```

```
        public string senha { get; set; }
        public void obterUsuario() {}
        public void definirUsuario() {}
        public abstract void obterSenha();
}

public class Venda
{
        public Currency valorVenda { get; set; }
        public float porcentagemDesconto { get; set; }
        public Usuario usuarioResponsavel { get; set; }
        public ICollection<Pagamento> pagamento { get; set; }
        public ICollection<Medicamento> medicamento { get; set; }
        public Cliente cliente { get; set; }
        public Usuario usuario { get; set; }
        public void obterVenda() {}
        public void definirVenda() {}
}
```

8-



Vantagem: Aumento encapsulamento e queda no acoplamento.

Desvantagem: Queda no desempenho.

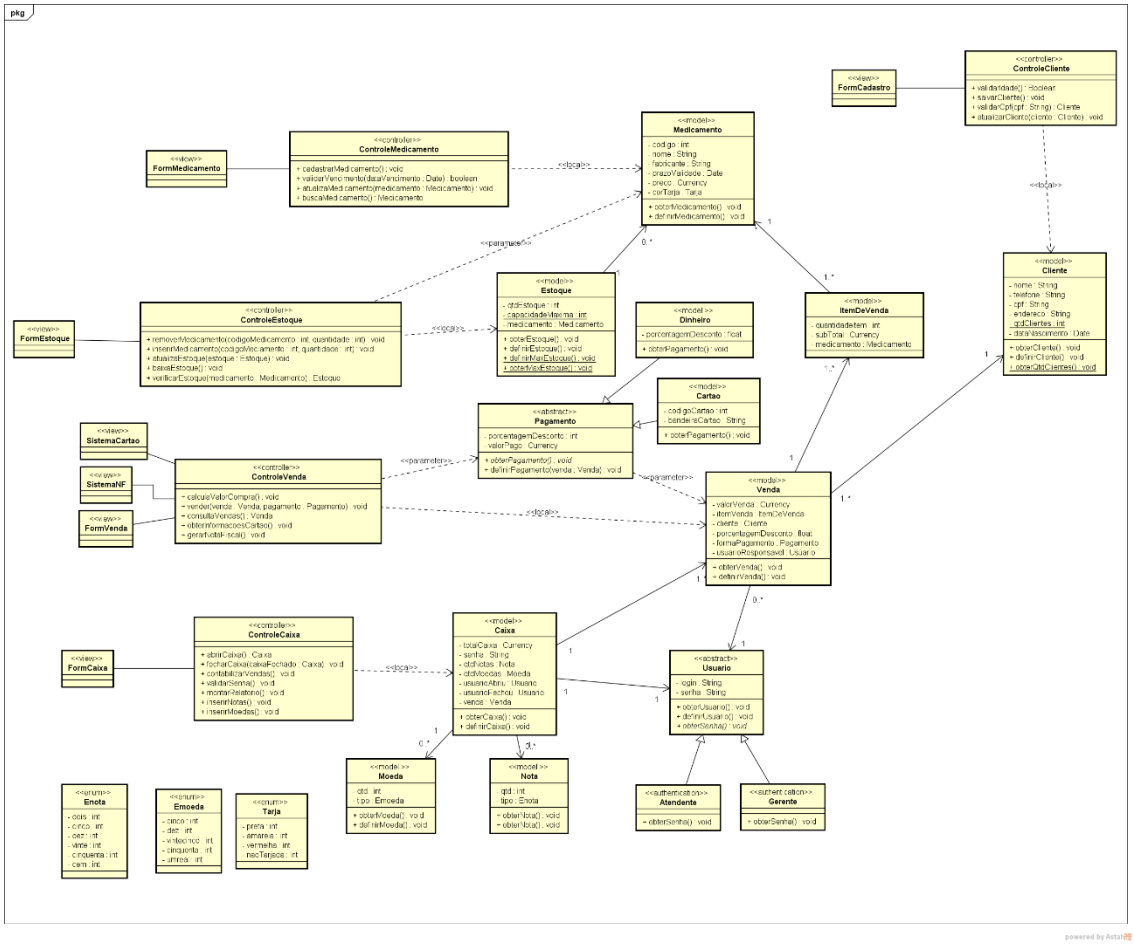
9-

```
public class ControleCaixa
{
        public void abrirCaixa() {}
        public void fecharCaixa(Caixa caixaFechado) {}
        public void contabilizarVendas() {}
        public void validarSenha() {}
        public void montarRelatorio() {}
        public void inserirNotas() {}
        public void inserirMoedas() {}
}
```

```
public class ControleCliente
{
	public Boolean validarIdade() {}
	public void salvarCliente() {}
	public boolean validarCpf(String Cpf) {}
	public void atualizarCliente(Cliente cliente) {}
}

public class ControleEstoque
{
	public void removerMedicamento(int codigoMedicamento, int
quantidade) {}
	public void inserirMedicamento(int codigoMeciamento, int
quantidade) {}
	public void atualizaEstoque(Estoque estoque) {}
	public void baixaEstoque() {}
}

public class ControleMedicamento
{
	public void cadastrarMedicamento() {}
	public boolean validarVencimento(Date dataVencimento) {}
	public void atualizaMedicamento(Medicamento medicamento) {}
	public void buscaMedicamento() {}
}
public class ControleVenda
{
	public void calculaValorCompra() {}
	public void vender(Venda venda) {}
	public void consultaVendas() {}
	public void obterInformacoesCartao() {}
	public void gerarNotaFiscal() {}
```

}

10-



Vantagem – Economia de memória, pois o objeto só existe em tempo de execução do método.

Desvantagem – Não é possível reaproveitar o objeto e utilizá-la em outros locais.

11-

```
public class ControleCaixa
{
        public void abrirCaixa()
        {
                var caixa = new Caixa();
        }
        public void fecharCaixa(Caixa caixaFechado) {}
        public void contabilizarVendas()
```

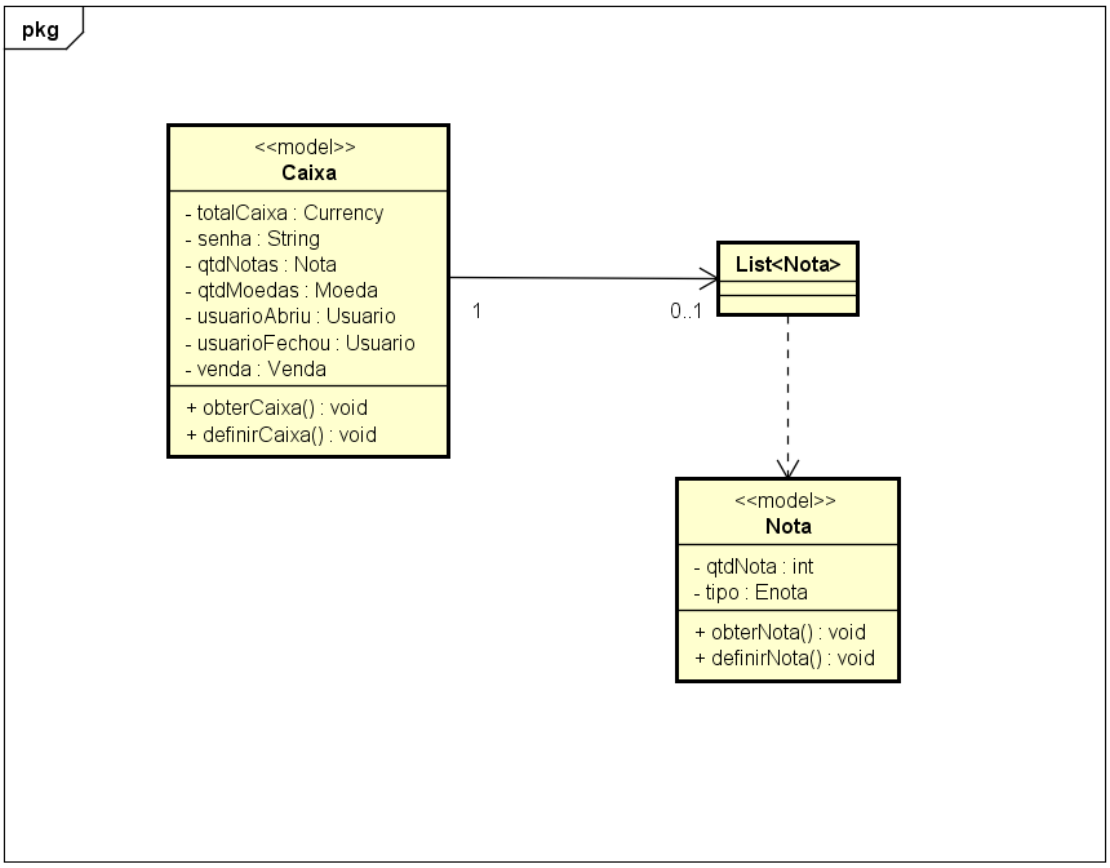
```
        {
                var caixa = new Caixa();
        }

        public void validarSenha()
        {
                var caixa = new Caixa();
        }
        public void montarRelatorio()
        {
                var caixa = new Caixa();
        }
        public void inserirNotas()
        {
                var caixa = new Caixa();
        }
        public void inserirMoedas()
        {
                var caixa = new Caixa();
        }
}

public class ControleCliente
{
        public Boolean validarIdade()
        {
                var cliente = new Cliente();
        }
        public void salvarCliente()
        {
                var cliente = new Cliente();
        }
```

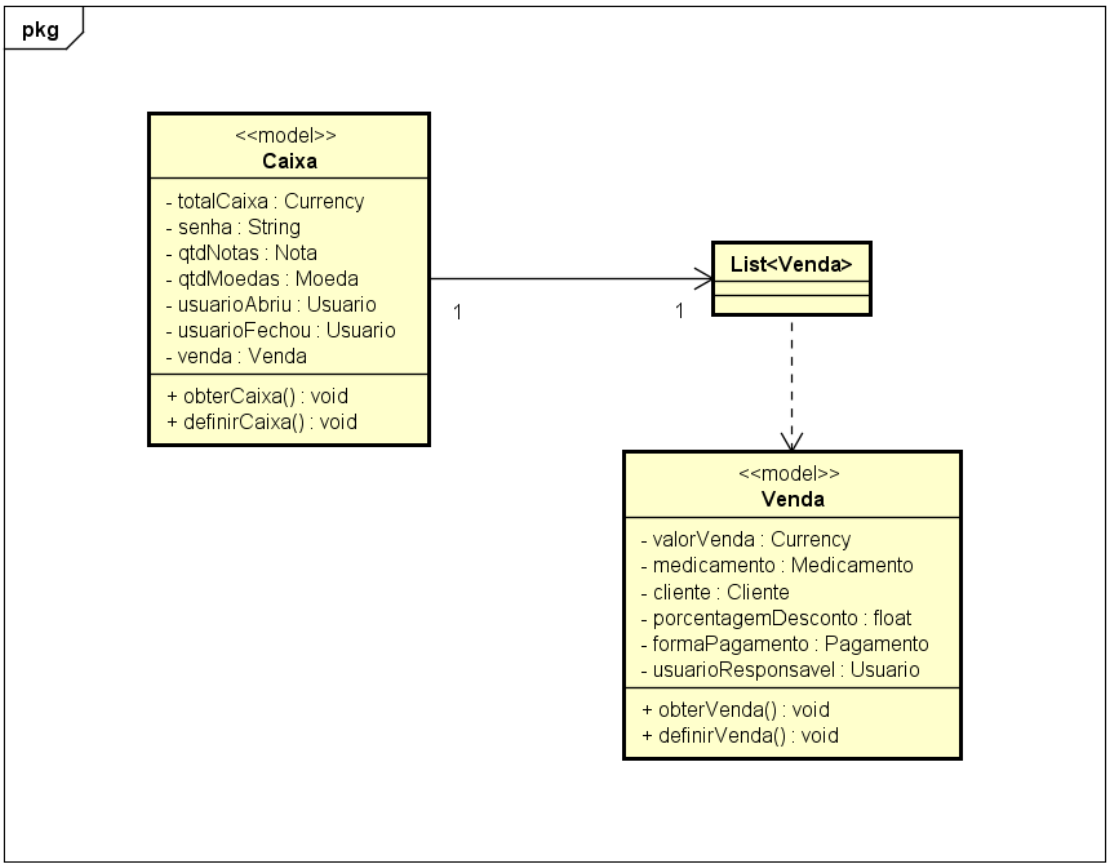
```
        public boolean validarCpf(String in Cpf)
        {
                var cliente = new Cliente();
        }
        public void atualizarCliente(Cliente cliente) {}
}


public class ControleEstoque
{
        public void removerMedicamento(int in codigoMedicamento, int in
quantidade)
        {
                var medicamento = new Medicamento();
        }
        public void inserirMedicamento(int in codigoMeciamento, int in
quantidade)
        {
                var medicamento = new Medicamento();
        }
        public void atualizaEstoque(Estoque estoque)
        {
                var medicamento = new Medicamento();
        }
        public void baixaEstoque()
        {
                var medicamento = new Medicamento();
        }
        public Estoque verificarEstoque(Medicamento medicamento)
        {
                var medicamento = new Medicamento();
        }
}
public class ControleMedicamento
```

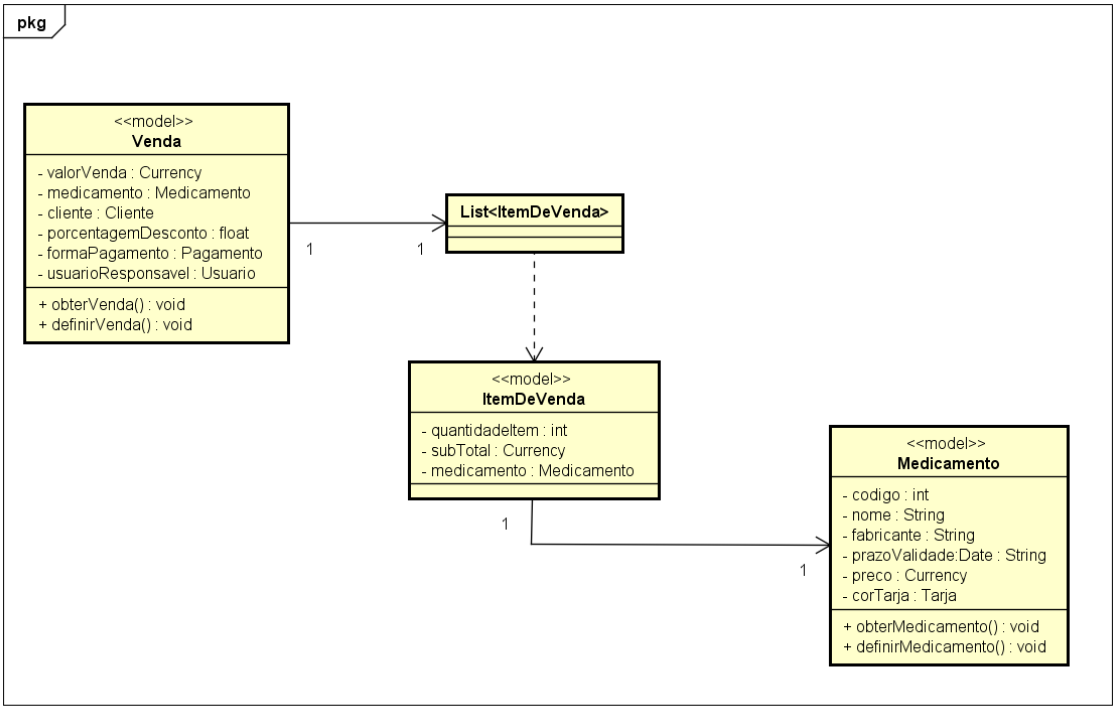
```
{
        public void cadastrarMedicamento()
        {
                var medicamento = new Medicamento();
        }
        public boolean validarVencimento(Date dataVencimento)
        {
                var medicamento = new Medicamento();
        }
        public void atualizaMedicamento(Medicamento medicamento)
        {
                var medicamento = new Medicamento();
        }
        public void buscaMedicamento()
        {
                var medicamento = new Medicamento();
        }
}
public class ControleVenda
{
        public void calculaValorCompra()
        {
                var medicamento = new Medicamento();
        }
        public void vender(Venda venda) {}
        public void consultaVendas() {}
        public void obterInformacoesCartao()
        {
                var cliente = new Cliente();
        }
        public void gerarNotaFiscal()
        {
```

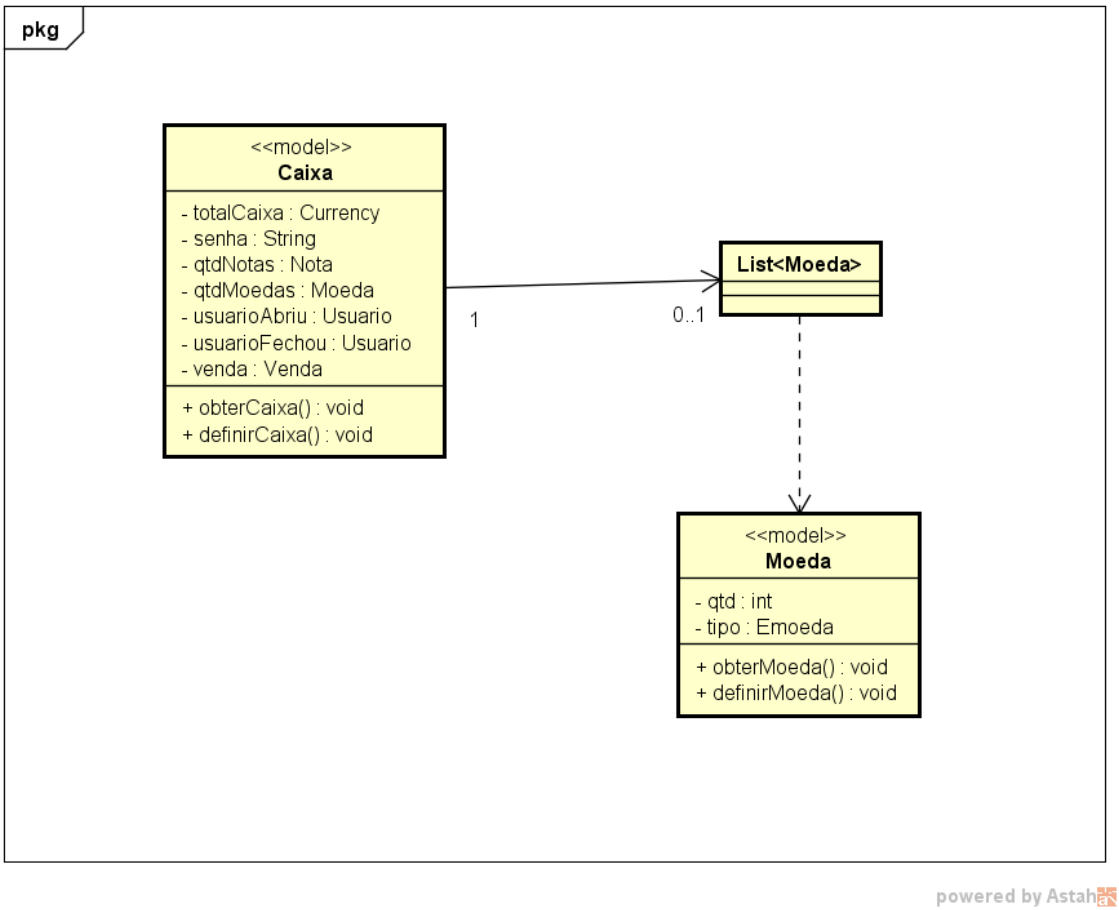
```
            var usuario = new Usuario();
        }
}
```

12-


pkg
<<model>>
Caixa
- totalCaixa : Currency
- senha : String
- qtdNotas : Nota
- qtdMoedas : Moeda
- usuarioAbriu : Usuario
- usuarioFechou : Usuario
- venda : Venda
+ obterCaixa() : void
+ definirCaixa() : void
1
0..1
List<Nota>
<<model>>
Nota
- qtdNota : int
- tipo : Enota
+ obterNota() : void
+ definirNota() : void
powered by Astah







As classes utilizando List foram modeladas para resolver o lado Muitos das relações. Dessa forma, as classes que precisam de vários objetos de outras

classes podem agrupá-los em uma única Collection List, que facilita a manipulação. Além disso, as Lists possibilitam a utilização de objetos repetidos.

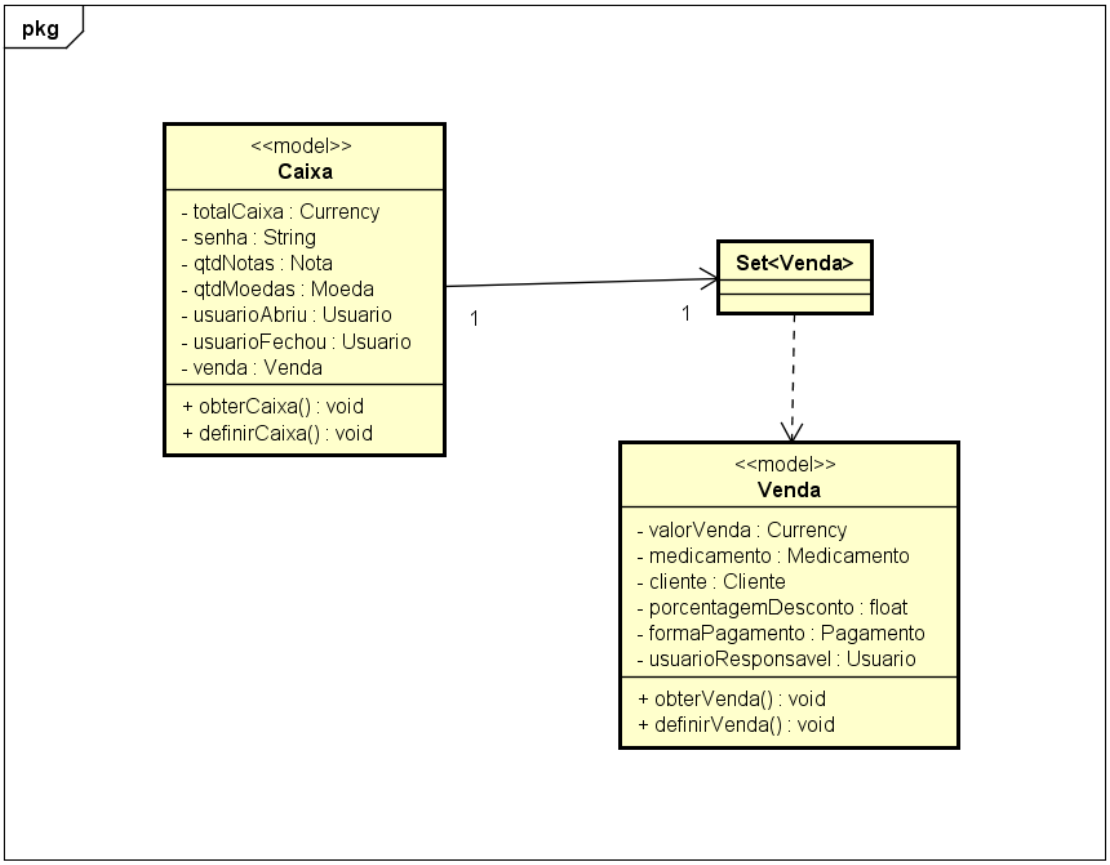
13-

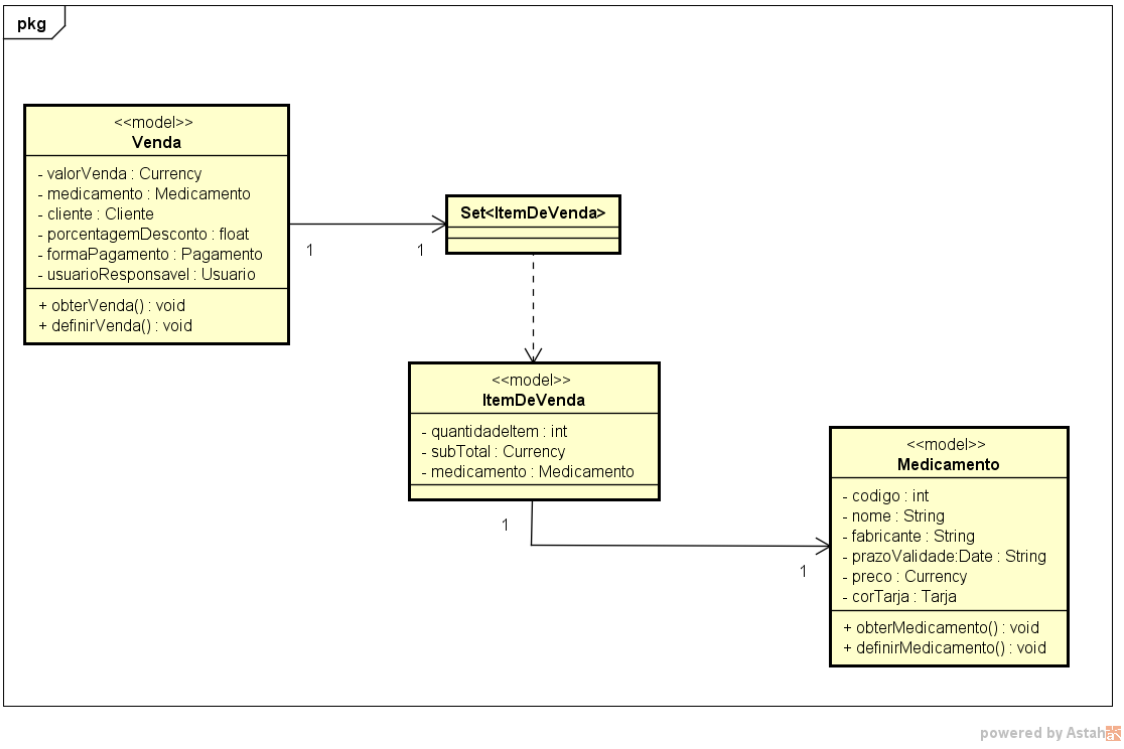
```
public class Estoque
{
        public int qtdEstoque { get; set; }
        public static int capidadeMaxima { get; set; }
        public List<Medicamento> medicamento { get; set; }
        public void obterEstoque() {}
        public void definirEstoque() {}
        public static void definirMaxEstoque() {}
}
public class Medicamento
{
        public int codigo { get; set; }
        public string nome { get; set; }
        public string fabricante { get; set; }
        public Date prazoValidade { get; set; }
        public Currency preco { get; set; }
        public Tarja corTarja { get; set; }
        public Estoque estoque { get; set; }
        public void obterMedicamento() {}
        public void definirMedicamento() {}
}
public class Venda
{
        public Currency valorVenda { get; set; }
        public float porcentagemDesconto { get; set; }
        public Usuario usuarioResponsavel { get; set; }
        public List<Pagamento> pagamento { get; set; }
        public List<Medicamento> medicamento { get; set; }
```

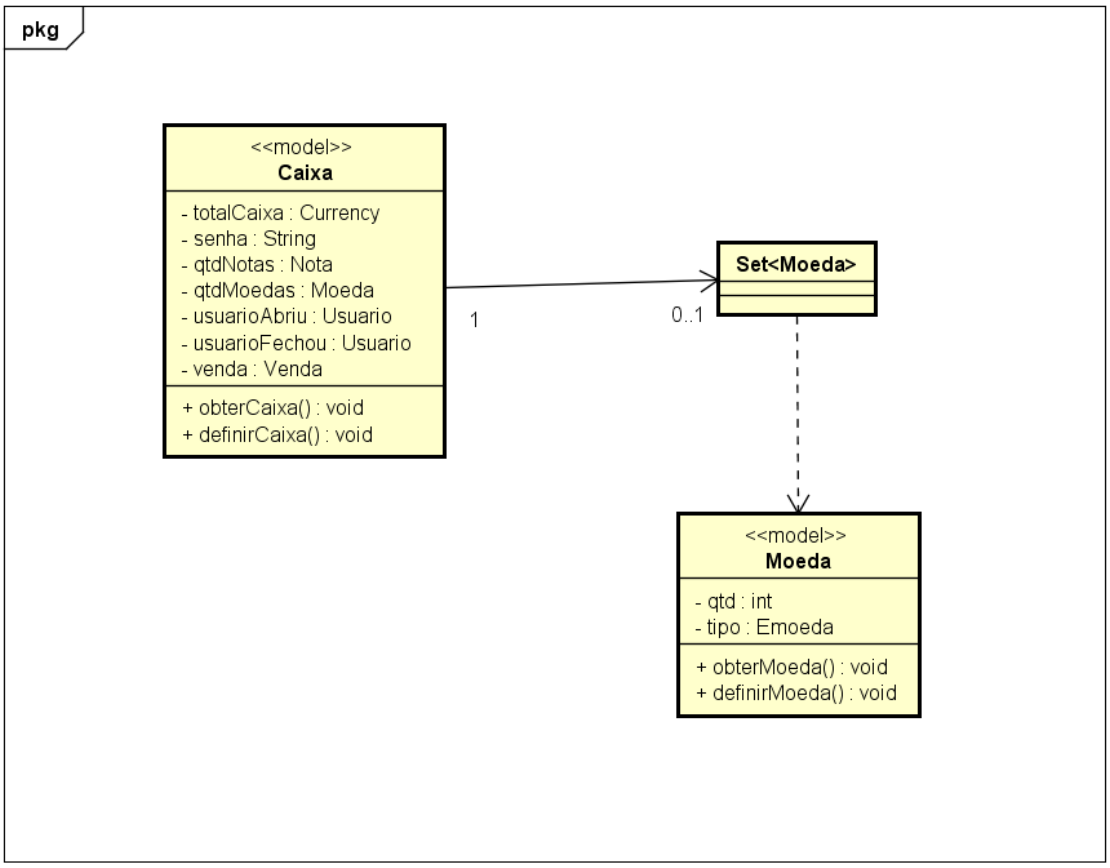
```
        public Cliente cliente { get; set; }
        public Usuario usuario { get; set; }
        public void obterVenda() {}
        public void definirVenda() {}
}
public class Pagamento
{
        public int porcentagemDesconto { get; set; }
        public Currency valorPago { get; set; }
        public abstract void obterPagamento();
        public void definirPagamento(Venda venda)
}
public class Caixa
{
        public Currency totalCaixa { get; set; }
        public string senha { get; set; }
        public Nota qtdNotas { get; set; }
        public Moeda qtdMoedas { get; set; }
        public Usuario usuarioAbriu { get; set; }
        public Usuario usuarioFechou { get; set; }
        public List<Nota> nota { get; set; }
        public List<Venda> venda { get; set; }
        public List<Moeda> moeda { get; set; }
        public void obterCaixa() {}
        public void definirCaixa() {}
}
public class Moeda
{
        public int qtd { get; set; }
        public Emoeda tipo { get; set; }
        public Caixa caixa { get; set; }
        public void obterMoeda() {}
```

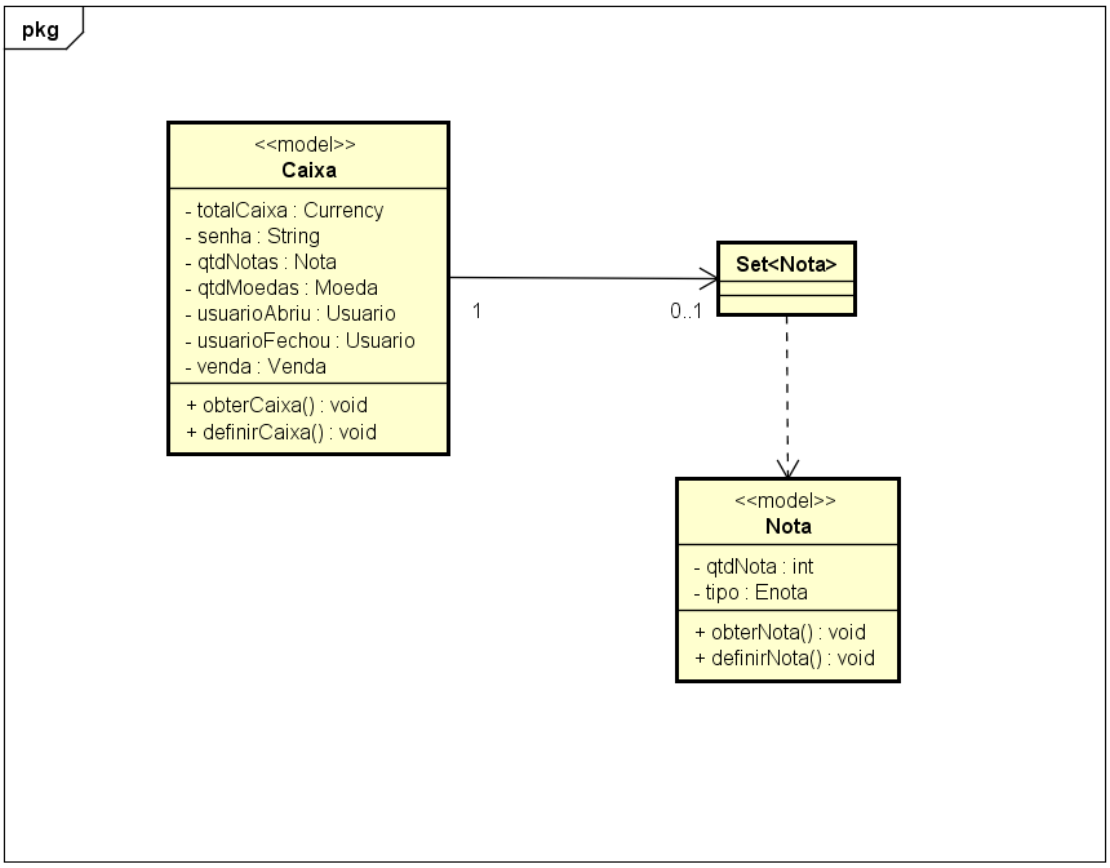
```
        public void definirMoeda() {}
}
public class Nota
{
        public int qtd { get; set; }
        public Enota tipo { get; set; }
        public void obterNota() {}
        public void obterNota() {}
}
public class Usuario
{
        public string login { get; set; }
        public string senha { get; set; }
        public void obterUsuario() {}
        public void definirUsuario() {}
        public abstract void obterSenha();
}
```

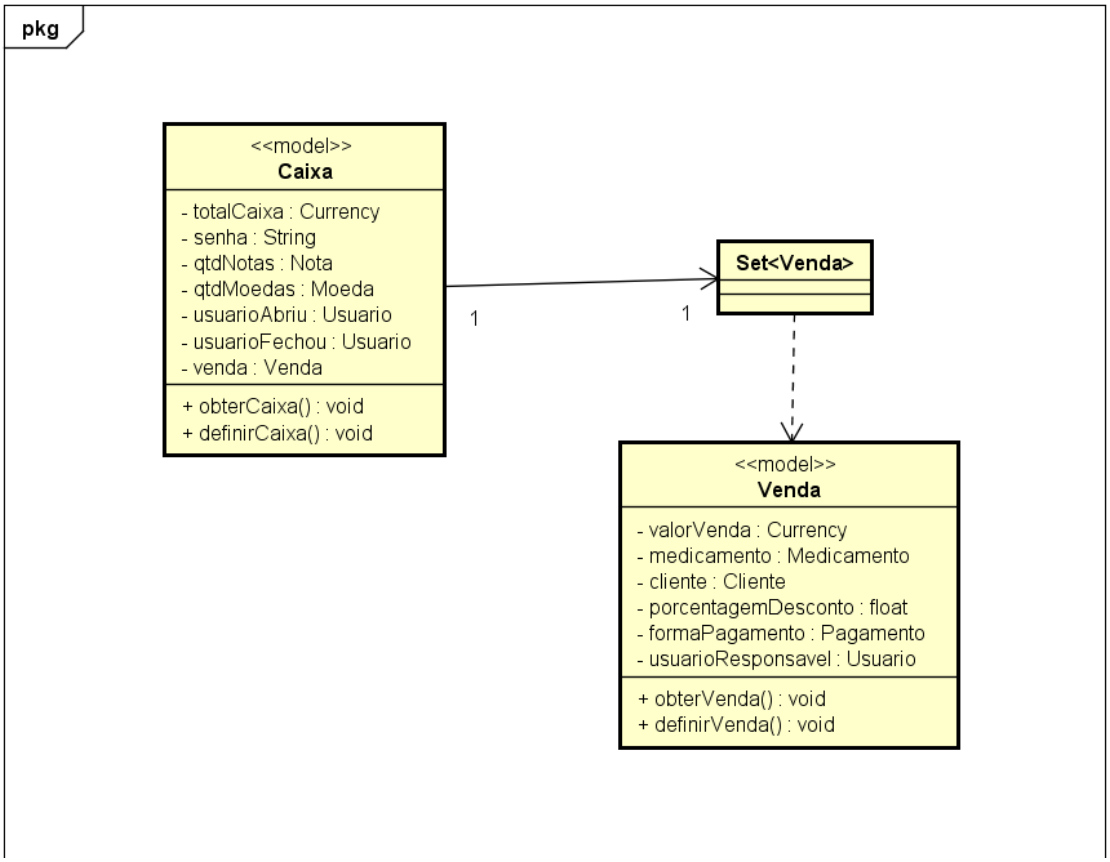
14-


pkg
<<model>>
Caixa
- totalCaixa : Currency
- senha : String
- qtdNotas : Nota
- qtdMoedas : Moeda
- usuarioAbriu : Usuario
- usuarioFechou : Usuario
- venda : Venda
+ obterCaixa() : void
+ definirCaixa() : void
1
1
Set<Venda>
<<model>>
Venda
- valorVenda : Currency
- medicamento : Medicamento
- cliente : Cliente
- porcentagemDesconto : float
- formaPagamento : Pagamento
- usuarioResponsavel : Usuario
+ obterVenda() : void
+ definirVenda() : void
powered by Astah


pkg
<<model>>
Venda
- valorVenda : Currency
- medicamento : Medicamento
- cliente : Cliente
- porcentagemDesconto : float
- formaPagamento : Pagamento
- usuarioResponsavel : Usuario
+ obterVenda() : void
+ definirVenda() : void
1
1
Set<ItemDeVenda>
<<model>>
ItemDeVenda
- quantidadeItem : int
- subTotal : Currency
- medicamento : Medicamento
1
1
<<model>>
Medicamento
- codigo : int
- nome : String
- fabricante : String
- prazoValidade:Date : String
- preco : Currency
- corTarja : Tarja
+ obterMedicamento() : void
+ definirMedicamento() : void
powered by Astah







As classes utilizando Set foram modeladas para resolver o lado Muitos das relações. Dessa forma, as classes que precisam de vários objetos de outras classes podem agrupá-los em uma única Collection Set, que facilita a manipulação dos objetos. Entretanto, a Collection do tipo Set não permite a repetição de seus objetos, o que a torna mais eficiente para alguns casos.

15-

```
public class Estoque
{
        public int qtdEstoque { get; set; }
        public static int capidadeMaxima { get; set; }
        public HashSet<Medicamento> medicamento { get; set; }
        public void obterEstoque() {}
        public void definirEstoque() {}
        public static void definirMaxEstoque() {}
}
public class Medicamento
{
        public int codigo { get; set; }
        public string nome { get; set; }
        public string fabricante { get; set; }
        public Date prazoValidade { get; set; }
        public Currency preco { get; set; }
        public Tarja corTarja { get; set; }
        public Estoque estoque { get; set; }
        public void obterMedicamento() {}
        public void definirMedicamento() {}
}
public class Venda
{
        public Currency valorVenda { get; set; }
        public float porcentagemDesconto { get; set; }
        public Usuario usuarioResponsavel { get; set; }
        public HashSet<Pagamento> pagamento { get; set; }
        public HashSet<Medicamento> medicamento { get; set; }
```

```
        public Cliente cliente { get; set; }
        public Usuario usuario { get; set; }
        public void obterVenda() {}
        public void definirVenda() {}
}
public class Pagamento
{
        public int porcentagemDesconto { get; set; }
        public Currency valorPago { get; set; }
        public abstract void obterPagamento();
        public void definirPagamento(Venda venda)
}
public class Caixa
{
        public Currency totalCaixa { get; set; }
        public string senha { get; set; }
        public Nota qtdNotas { get; set; }
        public Moeda qtdMoedas { get; set; }
        public Usuario usuarioAbriu { get; set; }
        public Usuario usuarioFechou { get; set; }
        public HashSet<Nota> nota { get; set; }
        public HashSet<Venda> venda { get; set; }
        public HashSet<Moeda> moeda { get; set; }
        public void obterCaixa() {}
        public void definirCaixa() {}
}
public class Moeda
{
        public int qtd { get; set; }
        public Emoeda tipo { get; set; }
        public Caixa caixa { get; set; }
        public void obterMoeda() {}
```

```
        public void definirMoeda() {}
}
public class Nota
{
        public int qtd { get; set; }
        public Enota tipo { get; set; }
        public void obterNota() {}
        public void obterNota() {}
}
public class Usuario
{
        public string login { get; set; }
        public string senha { get; set; }
        public void obterUsuario() {}
        public void definirUsuario() {}
        public abstract void obterSenha();
}
```

16-

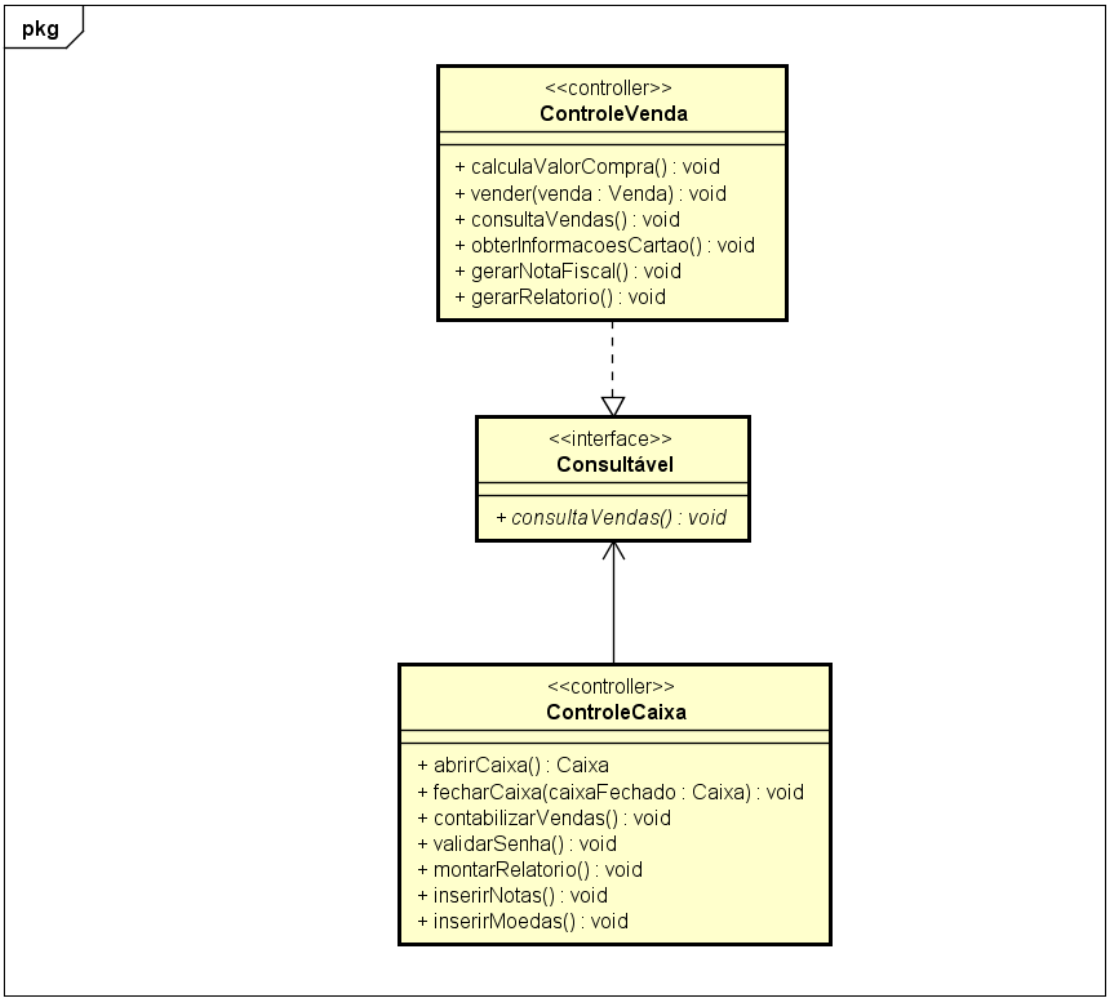
A classe parametrizada é uma classe que define outras classes e possui atributos ou operações com a definição feita em função de parâmetros.

A collection são classes que podem conter vários elementos dentro de si, assim como métodos que os tratam. São genéricas por natureza, podendo receber quaisquer dados de qualquer tipo. Para definir um só tipo para a collection, usa-se uma classe parametrizada.

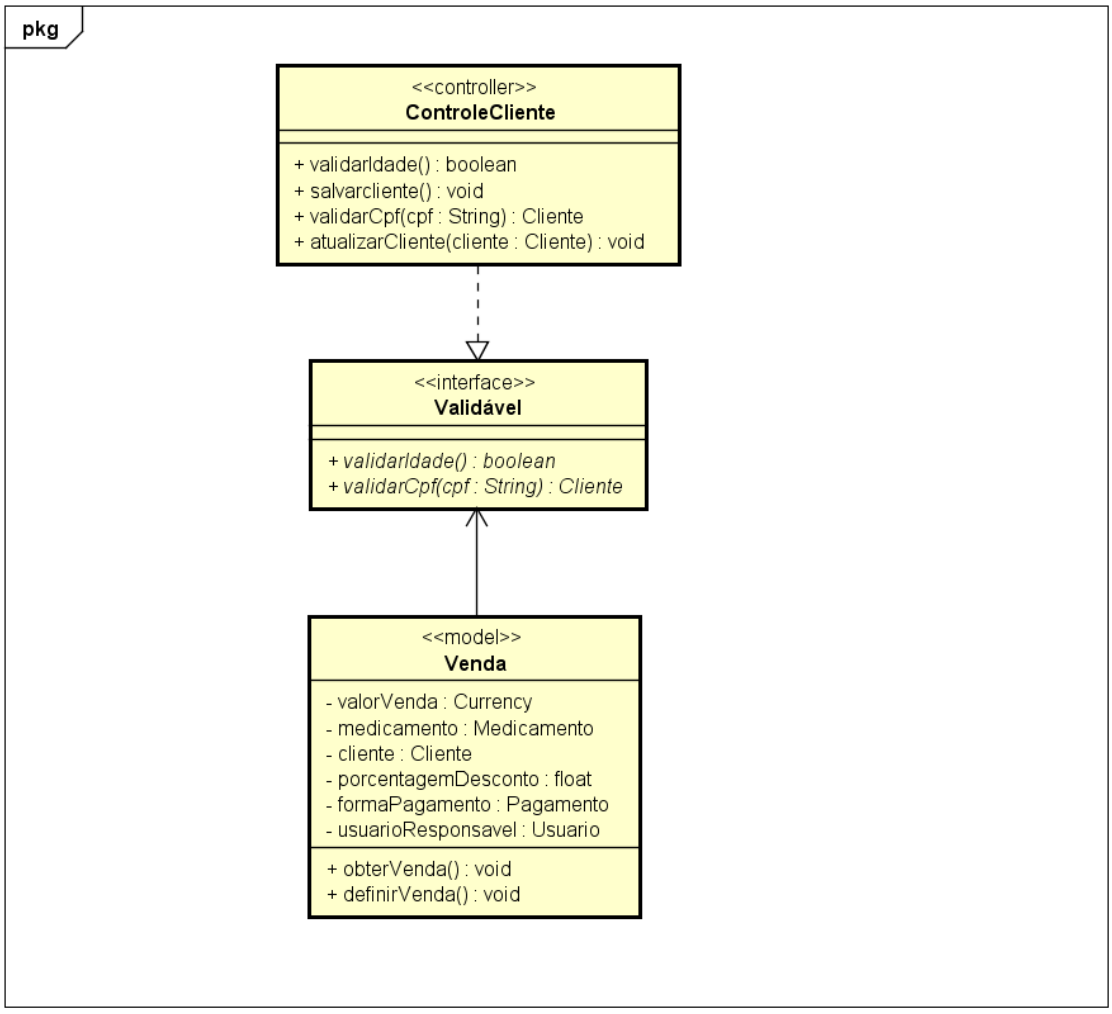
Um multiobjeto é o nome usado pela UML para uma coleção de objetos de uma mesma classe, ou seja, uma collection com tipo já definido, pois já está em tempo de execução.

Ná prática, os três conceitos representam uma coleção de objetos: classe parametrizada no diagrama de classes, collection na programação e multiobjeto no diagrama de sequência.

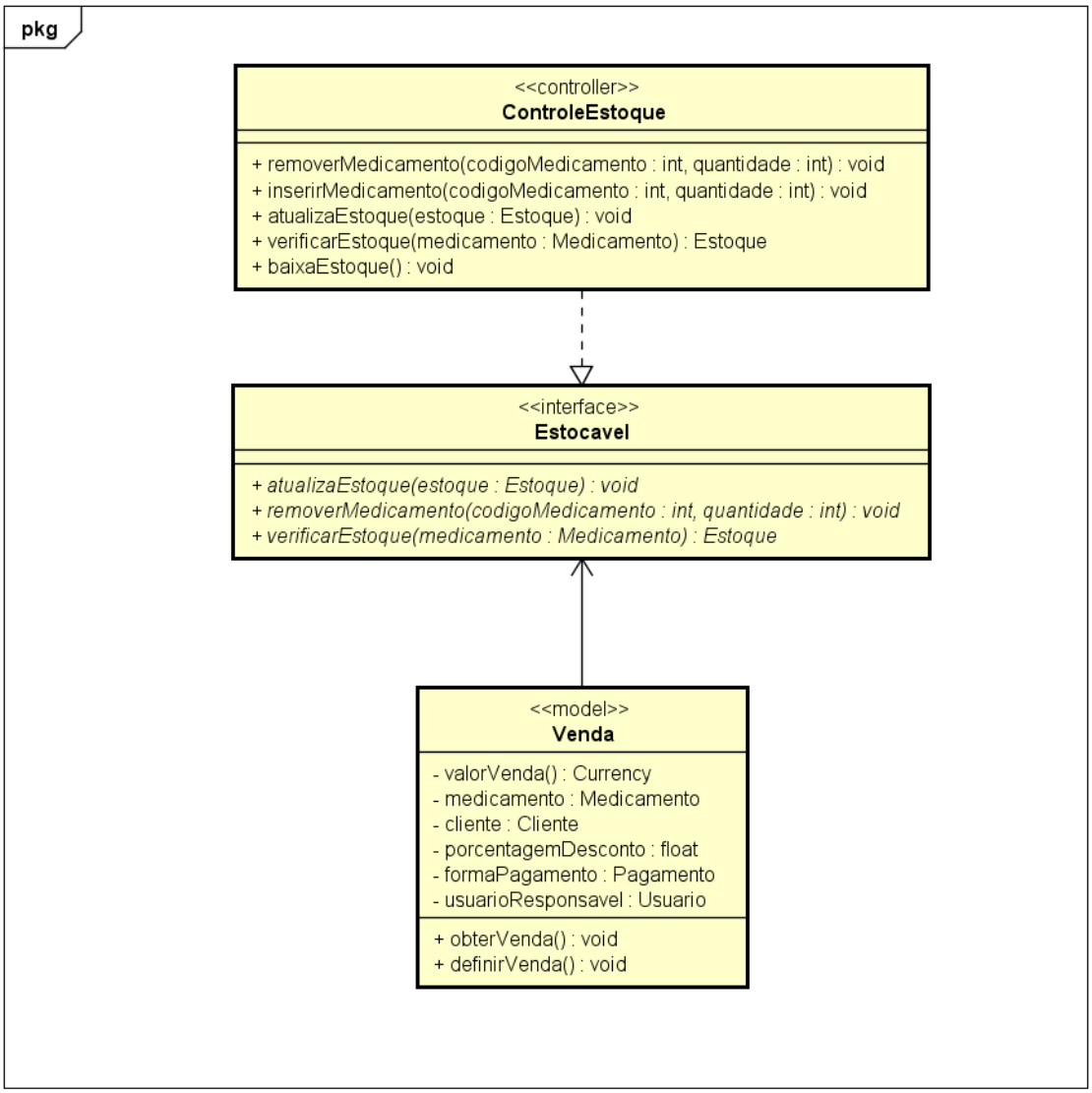
17-



A) Para montar o relatório, a classe ControleCaixa precisa realizar a consulta de algumas vendas. Foi montada a interface “Consultável” para evitar a exposição da classe inteira de ControleVenda.



B) Para ver se o cliente é cadastrado ou não, a classe de Venda precisa buscar o cliente por meio do CPF. E para ver se o cliente é idoso ou não, precisa utilizar o método de validarIdade. Assim, surge a interface para não expor o controle de cliente inteiramente.



C) Como a classe Venda precisa fazer buscas e alterações no Estoque, ela utiliza alguns métodos do ControleEstoque para isso. Buscando a não-exposição da classe toda, violando o encapsulamento, foi escolhido fazer uma interface que utilizasse somente os métodos importantes da classe ControleEstoque.

18-

========== a ==========

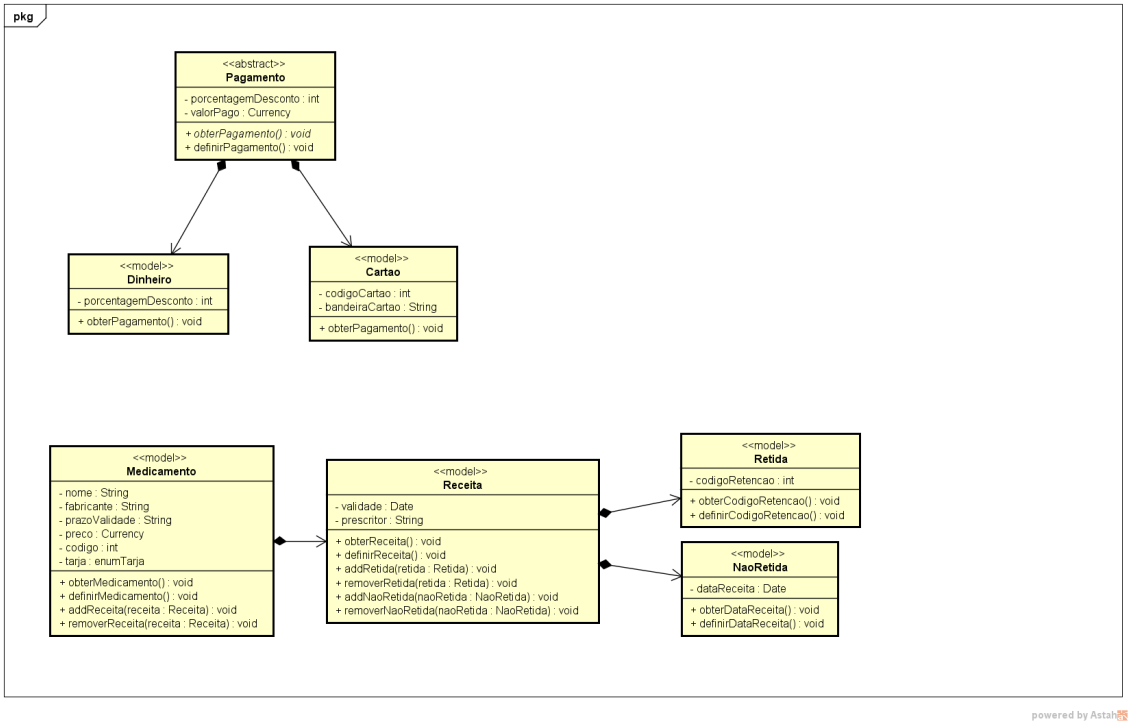
```
public class ControleEstoque : Estocavel
{
    public void removerMedicamento(int codigoMedicamento, int quantidade) {}
    public void inserirMedicamento(int codigoMedicamento, int quantidade) {}
```

```
        public void atualizaEstoque(Estoque estoque) {}
        public Estoque verificarEstoque(Medicamento medicamento) {}
        public void baixaEstoque() {}
}
public interface Estocavel
{
        void atualizaEstoque(Estoque estoque);
        void removerMedicamento(int codigoMedicamento, int quantidade);
        Estoque verificarEstoque(Medicamento medicamento);
}
public class Venda
{
        private Currency valorVenda{ get; set; }
        private Medicamento medicamento{ get; set; }
        private Cliente cliente{ get; set; }
        private float porcentagemDesconto{ get; set; }
        private Pagamento formaPagamento{ get; set; }
        private Usuario usuarioResponsavel{ get; set; }
        private Estocavel iEstoque{ get; set; }
        public void obterVenda() {}
        public void definirVenda() {}
}


==================== b ====================
public class ControleCaixa
{
        private Consultavel iVenda{ get; set; }
        public Caixa abrirCaixa() {}
        public void fecharCaixa(Caixa caixaFechado) {}
        public void contabilizarVendas() {}
        public void validarSenha() {}
        public void montarRelatorio() {}
```

```
        public void inserirNotas() {}
        public void inserirMoedas() {}
}
public interface Consultavel
{
        Venda obterHistorico();
        void consultaVendas();
}
public class ControleVenda : Consultavel
{
        public void calculaValorCompra() {}
        public void vender(Venda venda) {}
        public void consultaVendas() {}
        public void obterInformacoesCartao() {}
        public void gerarNotaFiscal() {}
        public Venda obterHistorico() {}
}
==================== c ====================
public class ControleCliente : Validavel
{
        public boolean validarIdade() {}
        public void salvarcliente() {}
        public Cliente validarCpf(String cpf) {}
        public void atualizarCliente(Cliente cliente) {}
}
public interface Validavel
{
        boolean validarIdade();
        Cliente validarCpf(String cpf);
}
public class Venda
{
```

```
        private Currency valorVenda{ get; set; }
        private Medicamento medicamento{ get; set; }
        private Cliente cliente{ get; set; }
        private float porcentagemDesconto{ get; set; }
        private Pagamento formaPagamento{ get; set; }
        private Usuario usuarioResponsavel{ get; set; }
        private Validavel iValidacaoCliente{ get; set; }
        public void obterVenda() {}
        public void definirVenda() {}
}
```

19-



Delegação 1 – FormaPagamento: Foi criada para tentar resolver o problema de Classificação Dinâmica que surgia na herança.

Delegação 2 – Medicamento: Como nem todo medicamento exige receita, foi criada a delegação Receita, e nela a Retida e Não Retida, pois alguns medicamentos exigem apenas a apresentação da receita e a verificação da validade da receita, e para os que exigem que a receita fique retida, é necessário manter o código dela para que haja um controle das cópias.

20-

```
public class Cartao
{
	private int codigoCartao{ get; set; }
	private String bandeiraCartao{ get; set; }
	public void obterPagamento() {}
}
public class Pagamento
{
	private int porcentagemDesconto{ get; set; }
	private Currency valorPago{ get; set; }
	private Pagamento pagamento = new Pagamento();
	public void obterPagamento() {}
	{
		pagamento.obterPagamento();
	}
	public void definirPagamento() {}
}
public class Dinheiro : Pagamento
{
	private int porcentagemDesconto{ get; set; }
	public void obterPagamento() {}
}
public class Medicamento
{
	private String nome{ get; set; }
	private String fabricante{ get; set; }
	private String prazoValidade{ get; set; }
	private Currency preco{ get; set; }
	private int codigo{ get; set; }
	private enumTarja tarja{ get; set; }
	public void addReceita(Receita receita)
```

```
        {
                receita.addRetida(new Retida());
                // ou
                receita.addNaoRetida(new NaoRetida());
        }
        public void removerReceita(Receita receita)
        {
                receita.removerRetida(new Retida());
                // ou
                receita.removerNaoRetida(new NaoRetida());
        }
}
public class Receita
{
        private Date validade{ get; set; }
        private String prescritor{ get; set; }
        public void addRetida(Retida retida) {}
        public void removerRetida(Retida retida) {}
        public void addNaoRetida(NaoRetida naoRetida) {}
        public void removerNaoRetida(NaoRetida naoRetida) {}
}
public class Retida
{
        private int codigoRetencao{ get; set; }
}
public class NaoRetida
{
        private Date dataReceita{ get; set; }
}
```

21-

	Generalização	Realização	Delegação
Vantagens	• Fácil implementação • Estática	• Melhora no encapsulamento • Melhora no acoplamento	• O reuso é em tempo de execução • Melhora no encapsulamento
Desvantagens	• Possível violação do princípio de Liskov • Violação do encapsulamento	• Não instanciável • Possibilidade de implementação única	• Perda de desempenho • Não pode ser utilizada quando há classes parcialmente abstratas envolvidas.