FCT/Unesp – Presidente Prudente
Programação Orientada a Objetos
Prof. Dr. Danilo Medeiros Eler

Trabalho Prático 01

Instruções de Envio: enviar somente para **daniloelerunesp@gmail.com** (por favor, enviar **somente** para esse e-mail). No assunto do email você deve colocar: **[POO2019] Trabalho Prático 01.** No corpo do email você deve identificar o nome dos integrantes do grupo e RA. Anexar o código fonte e dados utilizados.
DICA: não envie como ZIP para o gmail não barrar o seu email, caso haja um executável (EXE ou JAR). Você deve renomear o arquivo *zip* para *renomearParaZIP*. **Por exemplo:** trabalho01-grupoTal.renomearParaZIP. Uma alternativa é colocar o trabalho em algum lugar – por exemplo, github, google drive, one drive, ou outro – para eu fazer download.

O desenvolvimento do trabalho deve ser feito por grupos de no máximo três pessoas.

Caso haja evidencia de **cópia**, os trabalhos envolvidos terão **nota zero**.

Data máxima para envio: O trabalho deve ser enviado por email até o dia **12/05/2019**.

Especificações do trabalho

Implemente um sistema de controle de vendas utilizando programação orientada a objetos. Quando um cliente efetua uma compra um registro de venda é aberto no sistema. A venda contém os itens que descrevem os produtos comprados e suas respectivas quantidades. O cliente pode efetuar o pagamento utilizando dinheiro, cheque ou cartão de crédito.

Um item da compra deve armazenar o preço do produto no dia em que ela foi efetuada, para evitar inconsistências. Portanto, o item deve armazenar o código, a descrição e o valor. Alternativamente, você pode armazenar uma cópia do objeto Produto. Para isso, você pode criar um novo objeto e atribuir os dados, ou criar um método que faça isso, ou ainda aproveitar o método clone(), que é herdado da classe base Object (ver instrução nas observações que estão no fim deste documento).

Além de armazenar dados de clientes, produtos e vendas, o sistema deve emitir os seguintes relatórios:

- **Clientes Geral:** exibe dados de todos os clientes cadastrados
- **Cliente Específico:** exibe dados de um determinado cliente
- **Gastos de um Cliente:** exibe o total gasto por um determinado cliente, exibindo os detalhes gerais das compras, isto é, número de nota fiscal, data da

compra, total gasto com a compra, informação sobre o pagamento. Ao fim, o relatório deve exibir o total geral gasto pelo cliente

- **Produtos Geral:** exibe dados dos produtos
- **Produto Específico:** exibe dados de um determinado produto
- **Vendas Geral:** exibe dados de todas as vendas. Deve exibir nome do cliente, CPF, número de nota fiscal, data da compra, total gasto, tipo de pagamento. No fim, deve ser exibido o total geral acumulado
- **Vendas Específico:** exibição detalhada de uma venda específica, exibe todos os dados
- **Vendas por tipo de pagamento – versão simplificada:** o usuário escolhe um tipo de pagamento (dinheiro, cheque ou cartão) e são exibidas as informações básicas da venda (cliente, total, data, dados do pagamento)
- **Vendas por tipo de pagamento – versão detalhada:** o usuário escolhe um tipo de pagamento (dinheiro, cheque ou cartão) e são exibidos todos os detalhes de uma venda.

Procure modularizar o máximo possível as operações do sistema, criando métodos bem específicos, os quais poderão ser reutilizados por outros métodos. Por exemplo, métodos que buscam e retornam um cliente ou uma venda.

Modelo Conceitual

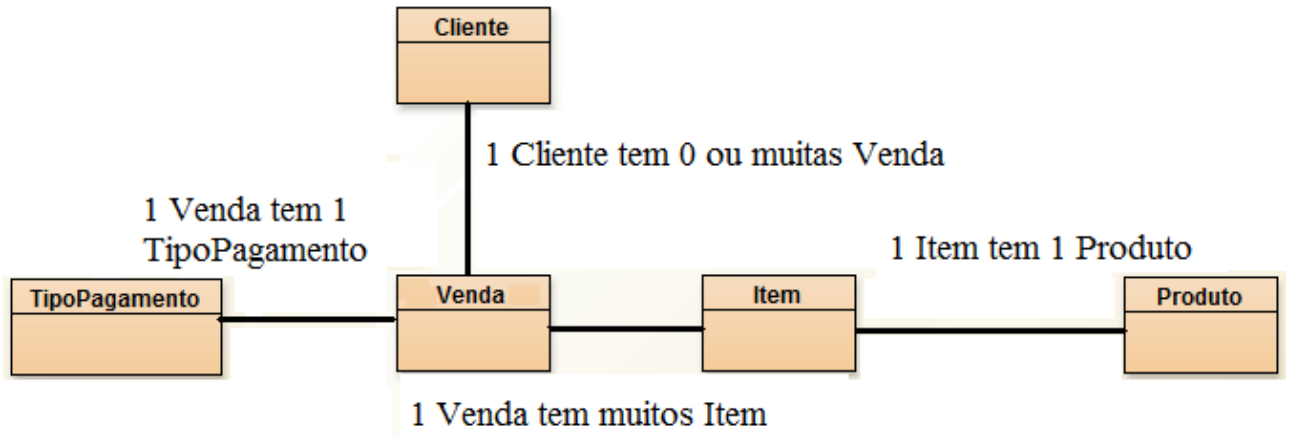
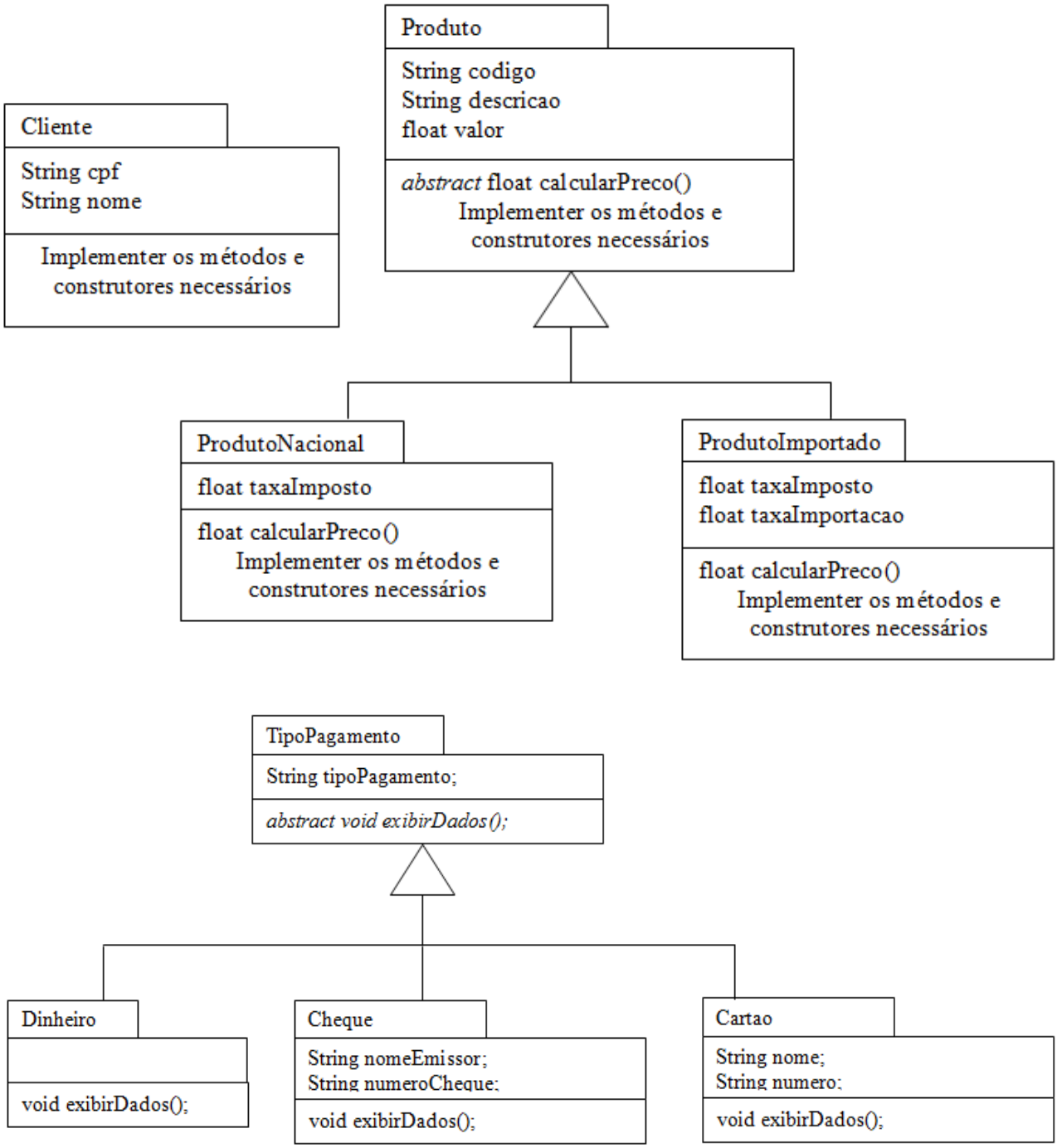
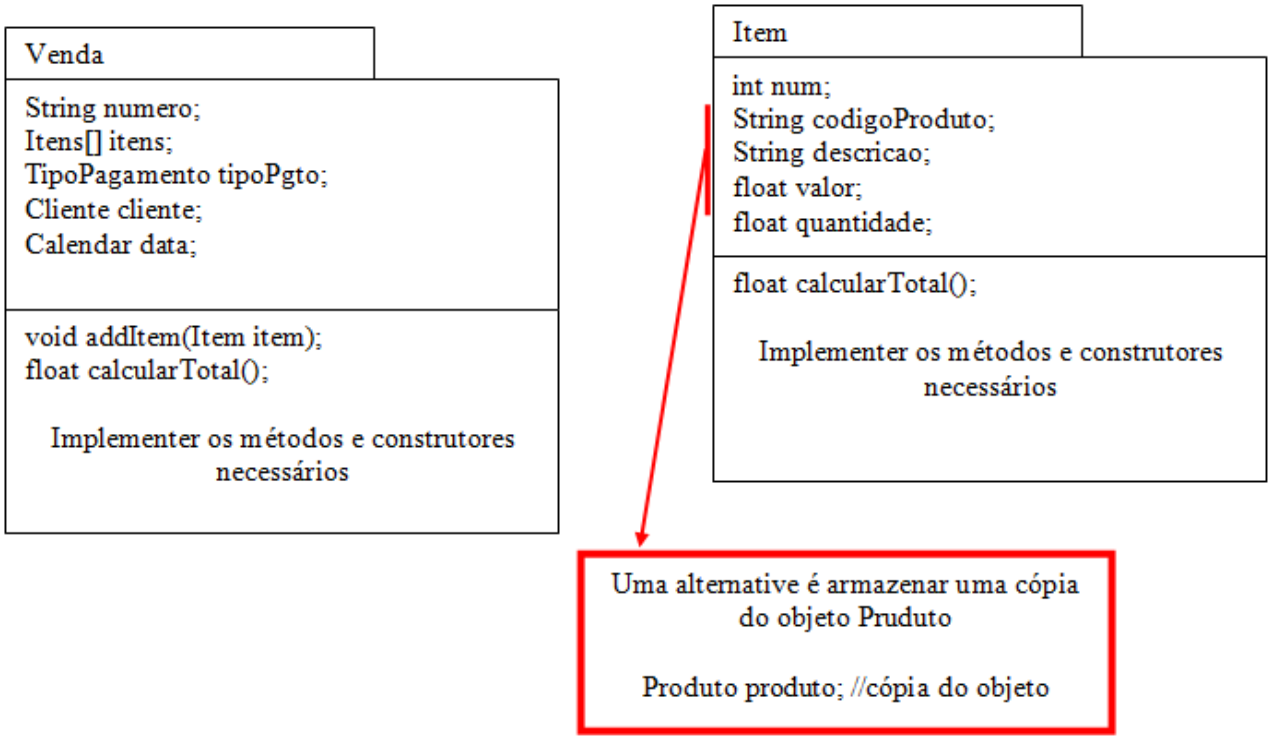


Diagrama de Classes

Os diagramas apresentados a seguir visam identificar o relacionamento de herança entre as classes. Além disso, eles descrevem os atributos e os métodos obrigatórios. Note que os métodos construtores, *getters* e *setters* foram omitidos. Portanto, você deve adicionar os atributos, os construtores e os métodos conforme a sua necessidade para desenvolver o sistema.


Produto
String codigo
String descricao
float valor
abstract float calcularPreco()
Implementer os métodos e construtores necessários
Cliente
String cpf
String nome
Implementer os métodos e construtores necessários
ProdutoNacional
float taxaImposto
float calcularPreco()
Implementer os métodos e construtores necessários
ProdutoImportado
float taxaImposto
float taxaImportacao
float calcularPreco()
Implementer os métodos e construtores necessários
TipoPagamento
String tipoPagamento;
abstract void exibirDados();
Dinheiro
void exibirDados();
Cheque
String nomeEmissor;
String numeroCheque;
void exibirDados();
Cartao
String nome;
String numero;
void exibirDados();



INTERFACE COM O USUÁRIO

Menu Principal
1 – Cadastrar
2 – Registrar Compra
3 – Relatórios
4 – Salvar Dados
5 – Carregar Dados
6 – Sair

Item de Menu Cadastrar
1 – Cliente
2 – Produto
3 – Sair (voltar)

Item de Menu Registrar Compra
Inicia o cadastro de uma compra, obtendo as informações do usuário

Item de Menu Relatórios
Deve conter todos os relatórios disponíveis no sistema, os quais foram anteriormente descritos.

Item de Menu Salvar Dados
Salva os dados em um arquivo (persistência de objetos – será apresentado em aula)

Item de Menu <u>Carregar Dados</u>
Carrega dados de um arquivo

Item de Menu <u>Sair</u>
Finaliza o sistema

OBSERVAÇÕES

- Por questões de simplicidade, o programa deve ser desenvolvido de acordo com a modelagem apresentada neste trabalho. No entanto, algumas modificações poderiam ser realizadas para melhorar a organização dos dados e das operações.

- O java possui uma classe chamada Calendar para manipular datas. Ela permite criar uma instância (um objeto) com a data e a hora do sistema. Veja o exemplo abaixo.

```
///Classe Calendar
Calendar d = Calendar.getInstance(); //data e hora do sistema
System.out.println(d.get(Calendar.DAY_OF_MONTH)); //obtem dia
System.out.println(d.get(Calendar.MONTH)); //obtem mes
System.out.println(d.get(Calendar.YEAR)); //obtem ano
d.set(2000,06,22); //maneira de setar uma data completa
System.out.println(d.get(Calendar.DAY_OF_MONTH));
System.out.println(d.get(Calendar.MONTH));
System.out.println(d.get(Calendar.YEAR));
d.set(Calendar.YEAR, 1990); //maneira de setar um campo especifico
System.out.println(d.get(Calendar.DAY_OF_MONTH));
System.out.println(d.get(Calendar.MONTH));
System.out.println(d.get(Calendar.YEAR));
```

- Lembre-se que em java as variáveis são ponteiros que referenciam os objetos. Então, para criar uma cópia você teria que criar um novo objeto e atribuir os valores do original para o novo objeto criado. Uma alternativa seria fazer isso por meio de um método de cópia, que retorna um objeto do mesmo tipo, ou então por meio de um construtor (new Produto(produto)), que criaria um objeto com os mesmos atributos daquele passado como parâmetro. Uma possibilidade é utilizar um método chamado clone(), o qual é herdado da classe base Object. Muitas classes disponíveis no java já disponibilizam esse método, que retorna uma cópia do objeto que invoca esse método. Por exemplo, se houver um

atributo data do tipo Calendar, e quiser fazer uma cópia dele, basta fazer data.clone(), fazendo *casting*. No seu caso, você deve sobrescrever o método clone nas classes que deseja permitir a cópia, como descrito abaixo.

```
public class Produto {
    private String codigo;
    private String descricao;
    private float valor;
    @Override
    public Object clone() {
        Produto produto = new Produto(); //cria uma nova instancia de Produto
        ///faz uma copia de todos os atributos
        produto.setCodigo(this.codigo); //pode usar os metodos ou os atributos
        produto.descricao = this.descricao; //Tem acesso direto, pois é da mesma classe
        produto.valor = this.valor;
        return produto; //retorna o novo objeto, que possui os mesmos valores de atributo
        //se houvesse algum atributo que fosse um objeto, seria necessario fazer uma copia
        // Por exemplo, se houvesse um atributo data do tipo MinhaData, faríamos
        //  produto.data = (MinhaData) this.data.clone();
        // O clone do objeto referenciado pela variável data deveria ser
        // implementado (sobrescrito) na classe MinhaData
        // Quando utilizamos alguma classe do java, ela já tem uma versão do método clone.
        //Por exemplo, a classe Calendar do java   produto.data = (Calendar) this.data.clone();
    }
```

```
Produto p1 = new Produto("111","d1111",111);
Produto p2 = (Produto) p1.clone();
System.out.println(p1);
System.out.println(p2);
p1.setCodigo("0000");
p2.setCodigo("99999");
System.out.println(p1);
System.out.println(p2);
```

SAÍDA

```
111 d1111 111.0
111 d1111 111.0
0000 d1111 111.0
99999 d1111 111.0
```