CEFET/RJ
Departamento Acadêmico de Informática (DEPIN)
Arquitetura e Padrões de Software - 1º trabalho prático - 2018/2
Prof. Eduardo Bezerra

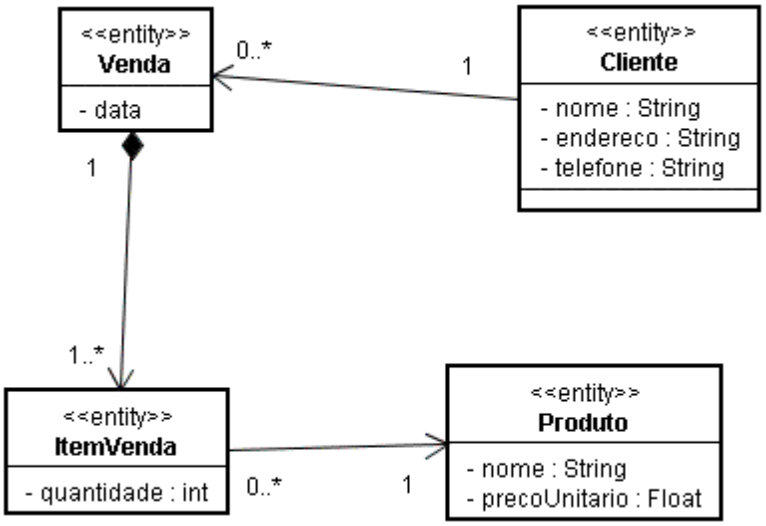
Tarefas

(1) Arquitetura de software e operação de sistema - A correta definição da arquitetura é fundamental em sistemas de software complexos, com efeitos na qualidade do produto de software gerado.

a) Descreva cada uma das camadas típicas da arquitetura de um sistema de informação. Nessa descrição, apresente o **propósito** de cada camada. A seguir, descreva de que forma ocorre a comunicação entre camadas em uma aplicação implementada em Java.

b) Explique em detalhes o conceito de **operação de sistema**. Na definição da arquitetura de um sistema de informação, em que contexto esse conceito se insere?

(2) Princípios de projeto OO – encapsulamento - Considere o modelo de classes a seguir para uma aplicação de uma loja virtual que comercializa produtos via Internet.



a) Forneça (em linguagem Java) a implementação das classes do diagrama acima, considerando atributos e associações apresentadas no diagrama de classes fornecido.

b) Considere que é necessário implementar na classe Venda uma operação cujo propósito é retornar o total de uma venda. Forneça duas implementações dessa operação, uma aderente e outra não, ao princípio do encapsulamento. Nas duas implementações que você fornecer, a assinatura da operação deve ser necessariamente **public BigDecimal getTotal()**. Forneça justificativas para as duas implementações, i.e., explique porque uma implementação viola e a outra não viola o princípio do encapsulamento.

(3) Operações de sistema, camada do serviço - Ainda no contexto do diagrama de classes do item anterior, considere a adição de uma nova classe no diagrama, Pagamento. Essa classe representa o registro de pagamento de uma parcela para uma venda. Uma venda pode estar associada a vários (zero ou mais) pagamentos; cada pagamento está associado a uma venda apenas. Considere ainda a existência de um caso de uso para permitir o registro de um pagamento para uma venda criada previamente. Um cenário típico desse caso de uso é o seguinte:

> *O usuário fornece o código de uma venda, assim como o valor do pagamento para abater do total dessa venda. Além do valor, deve ser também registrada a data em que o pagamento foi realizado (embora o usuário não forneça diretamente esta data). Após o registro do pagamento, o sistema apresenta o saldo (i.e., quanto falta pagar do total da venda).*

a) Forneça a assinatura (nome, parâmetros, tipo de retorno) da operação de sistema correspondente ao registro de pagamento de uma venda. Essa operação deve constar em alguma classe da **camada de serviço** dessa aplicação de loja virtual.
b) Quais são as adições (em termos de classes, associações, operações ou atributos) que devem acontecer no modelo de classes original que se fazem necessárias por conta da existência da operação de sistema do item a? Justifique sua resposta.

(4) - Camada do domínio - Ainda no contexto da aplicação de vendas, considere que uma venda está quitada quando o seu valor total já foi pago. Nesse sentido, considere os seguintes métodos na classe Venda:

a) **public Boolean estaQuitada()**, cujo propósito é verificar se uma venda está quitada ou não. Esse método deve retornar true se a venda está quitada, e false em caso contrário.
b) **public void registrarPagamento(BigDecimal valorParcela)**, que recebe um único parâmetro do tipo BigDecimal correspondente ao valor de uma parcela dessa venda. O objetivo desse método é registrar o pagamento de uma parcela da venda.

Forneça a implementação dos métodos **estaQuitada** e **registrarPagamento**. Descreva, se for o caso, que atributos novos devem ser definidos na classe Venda para serem usados por conta desses novos métodos.

(5) - Operações de sistema - Considere uma aplicação para uma loja virtual para venda de produtos pela Web. Essa aplicação deve possuir um módulo para permitir ao usuário definir as entregas dos produtos de uma venda selecionada. Nessa funcionalidade, uma venda é selecionada. Então, os produtos dessa venda podem ser divididos pelo usuário em uma ou mais entregas. Para cada entrega, o usuário deve definir a data em que ocorrerá.

Considerando o contexto apresentado acima, forneça a descrição de cada um das operações que devem constar na **camada de serviço** dessa aplicação.

(6) - Invariantes de uma classe - Considere uma classe denominada Triangulo, que representa a forma geométrica de mesmo nome. Sabendo que os lados de um triângulo são representados nesta classe pelos atributos l1, l2 e l3, então algumas invariantes da classe Triangulo são:

l1 > 0
l2 > 0
l3 > 0
l1 + l2 > l3
l2 + l3 > l1
l3 + l1 > l2

Implemente a classe Triangulo em Java. Crie nessa classe um construtor que contém 3 parâmetros, cujos valores devem ser usados para iniciar o estado do objeto correspondente. Crie os métodos de modificação dos atributos l1, l2 e l3. Sua implementação deve ser tal que a invariante da classe não seja violada. Crie também um construtor que mantenha a invariante da classe.

(7)– DDD, Value Object – Em tarefas cotidianas, é comum utilizar um par de datas para registrar a duração de um determinado evento, como um congresso ou uma viagem. Considere uma classe que representa o conceito de um período de tempo delimitado por duas datas. Nessa parte do trabalho, você deve implementar essa classe como um **Value Object**. Uma parte da definição dessa classe é apresentada abaixo.

```
public final class PeriodoDatas {
   private final Date inicio;
   private final Date termino;
...
   public FaixaDatas prolongar(int numDias) {
     ...
  }
...
}
```

Os atributos inicio e termino devem armazenar o estado de cada objeto dessa classe. Ambos são do tipo java.util.Date.

Implemente um construtor com a seguinte assinatura:

PeriodoDatas(String de, String ate)

Um exemplo de chamada desse construtor é

periodo = new PeriodoDatas("08/08/2018", "13/08/2018")

Repare que os argumentos do construtor são do tipo String, mas os atributos são do tipo java.util.Date. Durante a implementação desse construtor, atente para o seguinte:

a) Você deve verificar a validade dos argumentos. Se ao menos um deles não for válido (no sentido de não corresponder a uma data válida), o construtor deve disparar a exceção IllegalArgumentException.
b) Você deve verificar se a data de início é igual ou anterior à data de término. Se não for esse o caso, o construtor também deve disparar IllegalArgumentException.
c) Os usuários dessa classe não devem saber que internamente é utilizada a classe java.util.Date. Isso implica que métodos de PeriodoDatas criados para consultar os valores de início e fim do período devem necessariamente retornar String.

Implemente também o método prolongar. O objetivo desse método é retornar um novo objeto da classe PeriodoDatas que corresponda a um prolongamento do período de datas representado pelo objeto this. Por exemplo, considerando o objeto faixa acima, podemos ter:

```
periodoProlongado = periodo.prolongar(30)
```

O resultado seria a criação de um objeto PeriodoDatas com datas de início e termino iguais a "08/08/2018", "12/09/2018", respectivamente.

O que deve ser entregue

Você deve entregar um relatório em formato PDF detalhando a solução de cada tarefa. Além disso, você deve entregar todo em qualquer código fonte (arquivo .java) gerado em alguma tarefa do trabalho. **ATENÇÃO!** Crie uma pasta raiz denominada **t1**. Essa pasta deve conter uma subpasta para cada tarefa que demande a criação de algum código. Para submeter seu trabalho compacte a pasta t1 e o seu relatório (em PDF) e submeta o arquivo compactado pelo Moodle. O nome do arquivo compactado deve seguir o padrão T1_SEU_NOME_COMPLETO.zip. Esse trabalho é individual.