Centro de Engenharia Elétrica e Informática – CEEI
Unidade Acadêmica de Sistemas e Computação – UASC
Disciplina: Laboratório de Programação 2

Laboratório 05

Como usar esse guia:

- Leia atentamente cada etapa
- Referências bibliográficas incluem:
  - material de referência desenvolvido por professores de p2/lp2 em semestres anteriores (ONLINE)
  - o livro Use a cabeça, Java (LIVRO-UseCabecaJava)
  - o livro Java para Iniciantes (Livro-JavaIniciantes)
- Quadros com dicas tem leitura opcional, use-os conforme achar necessário
- Preste atenção nos trechos marcados como importante (ou com uma exclamação)!

Sumário

Acompanhe o seu aprendizado

Conteúdo sendo exercitado

- Padrões GRASP: ênfase no uso de expert da informação, creator, baixo acoplamento, alta coesão e controladores
- Uso de interfaces: Comparable<> e Comparator<>

- Polimorfismo com interfaces
- Uso de interface com composição como uma forma de ter menor acoplamento e polimorfismo (padrão de projeto strategy)
- Uso de testes de aceitação com easy accept

Objetivos de aprendizagem

- Temos dois objetivos principais nesse lab: 1) ter a experiência de um lab bem parecido com o projeto em termos de estruturação (um conjunto de casos de uso, testes de aceitação, sem interface com usuário, mais funcionalidades); 2) explorar o reuso de software com composição e interfaces, identificando vantagens e desvantagens de cada estratégia.

Perguntas que você deveria saber responder após este lab

- O polimorfismo é uma consequência do reuso com interfaces. Como usamos o polimorfismo no código?
- Como usar o padrão GRASP Controller?
- Considerando as implementações de coleções de Java, em quais situações cada coleção é mais apropriada para ser usada?
- Qual a relação da interface Comparable<> com os mecanismo de ordenação de coleções?
- Qual a relação da interface Comparator<> com os mecanismo de ordenação de coleções?

Para se aprofundar mais...

- Referências bibliográficas incluem:
    - material de referência desenvolvido por professores de p2/lp2 em semestres anteriores (ONLINE)
    - o livro Use a cabeça, Java (LIVRO-UseCabecaJava)
    - o livro Java para Iniciantes (Livro-JavaIniciantes)
- Projetando com interfaces
- Um pouco mais sobre GRASP

Introdução

Neste laboratório começaremos a trabalhar através de uma especificação com casos de uso seguindo um modelo típico usado em desenvolvimento de software. Uma especificação determina os requisitos dos usuários para o software. Cada funcionalidade do sistema a ser implementado é descrita em um caso de uso, ou seja, um cenário que descreve como o sistema é usado.

Ao contrário dos outros laboratórios, você não deve criar a classe main/menu.
Neste laboratório você está construindo a base (*backend*) do sistema. Sua obrigação é oferecer classes com métodos bem definidos para que qualquer

desenvolvedor possa pegar o que você desenvolveu para implementar a interface gráfica/textual/web com o usuário (*frontend*).

Para facilitar a vida do desenvolvedor de *frontend*, seu sistema deve ter uma classe Facade contendo todos os métodos e operações que o desenvolvedor responsável por implementar a interface gráfica ou textual precisa. Exemplo de uma classe Main que poderia ser implementada utilizando a classe Facade que seu sistema irá oferecer:

```
public class Main {
    public static void main(String[] args) {
         Facade facade = new Facade();
         facade.inicializa();
         facade.adicionaCliente(“João Neto","SPLAB","joaoneto@ccc.ufcg.edu.br”);
         // … etc
    }
}
```

Um outro ponto importante desse laboratório é o uso de testes de aceitação. Este tipo de teste está bem próximo do usuário do sistema, seu objetivo é verificar se o software está pronto para ser usado, oferecendo as funcionalidades definidas para o mesmo. **Veja esta apresentação** preparada pelos monitores e atualizada por por Raphael Agra para o RAE, **tem um passo a passo de como usar os testes de aceitação com o EasyAccept neste laboratório**, além de referências interessantes sobre o assunto. Além disso, atente para os testes de aceitação a serem usados nesse lab, cujos arquivos estão aqui.

Dica de implementação!

O que é Facade/Façade? Facade = Fachada. É uma classe que abstrai todo um sistema (ou um conjunto de subsistemas) para facilitar o uso dessas partes abstraídas.

A estratégia de criar uma única classe de entrada (como a classe Facade) é bastante comum e facilita a vida do desenvolvedor. A Facade pode conter um ou mais controladores do sistema. Exemplo:

```
public class Facade {
    // ...
    public void salva() {
        pagamentoController.salva();
        cadastroController.salva();
    }
    // ...
}
```

Idealmente a Facade deve trabalhar apenas com os tipos mais básicos do sistema. Isso diminui o acoplamento do sistema. Dessa forma, faça métodos que usem apenas os tipos básicos nos parâmetros e no retorno (Strings, int, double, etc.).

Uma fachada não é o mesmo que um controlador. Uma facade deve oferecer apenas métodos simples e de uso geral do usuário. Um controlador pode ter mais métodos e é responsável por fazer tratamento da lógica de negócio. Em geral a facade apenas delega ações para um ou mais controladores que o compõe.

Sistema para Auto-Gestão de comércio de Alimentos - SAGA

A comercialização de lanches acontece de maneira auto-gerida em alguns laboratórios de graduação da nossa universidade. Atualmente, vários fornecedores ofertam os seus produtos alimentícios nos laboratórios, mas não permanecem no local para receber o dinheiro da compra. Na verdade, eles deixam um caderninho, onde cada cliente mantém uma página (a chamada **conta**) para anotar as suas compras. Ao retirar um produto, o cliente anota a data, quantidade e o nome do produto na página de sua **conta**, no caderninho do fornecedor. Periodicamente, os clientes realizam o pagamento de suas **contas**. Essa dinâmica de comercialização, apesar de funcionar em termos gerais, dificulta o controle dos processos de venda de produtos e pagamento. Diante disto, surgiu a necessidade de informatizar este processo.

A atividade proposta por este Lab é implementar um sistema de comércio eletrônico para dar suporte à comercialização de lanches nos laboratórios. As necessidades de fornecedores e clientes motivou o surgimento do SAGA - Sistema para Auto-Gestão de comércio de Alimentos.

Neste sistema, é possível cadastrar os clientes, fornecedores dos lanches e os produtos oferecidos por eles. Através do sistema, é possível manter o registro das contas e compras dos clientes junto aos seus fornecedores. Além disso, do ponto de vista do administrator, será possível fazer consultas sobre clientes, fornecedores, produtos e compras de um determinado cliente.

Este Lab está estruturado no formato de Casos de Uso. Especificamente, abordaremos os seguintes casos de uso: (1) CRUD de Clientes; (2) CRUD de Fornecedores; (3) CRUD de Produtos de fornecedores; (4). Ordenação da listagem de clientes, fornecedores e produtos; (5) Cadastro de compra em uma conta; e (5) Listagem de compras de um cliente. Estes casos de uso constituem um subconjunto das possíveis funcionalidades do sistema SAGA.

O que é CRUD? É um acrônimo que na língua inglesa que significa: Create, Read, Update e Delete. Estas são as operações básicas de qualquer sistema de informação que precisa manter e manipular registros das entidades do domínio.

US1. CRUD de Clientes

User story: Como administrador do sistema, quero adicionar, recuperar, editar ou apagar clientes.

O sistema SAGA mantém o cadastro dos clientes com informações suficientes para identificá-los e localizá-los na universidade. Os clientes são identificados, unicamente, por

seu CPF e possuem outras informações adicionais como nome, email e localização (referente ao nome do local onde trabalham: LSD, LSI, SPLab, etc). Sempre que for feita uma operação sobre um cliente, este deve ser recuperado através do seu CPF. Não é possível cadastrar um cliente já existente.

A Facade do sistema deve ter métodos para:

- Cadastrar clientes;
  Em uma adição bem sucedida, o CPF deve ser retornado. Quando não é bem sucedida, uma exceção deve ser lançada e a mensagem de erro mostrada para o usuário. Após isso, o programa **não** deverá continuar.
- Retornar a representação textual de um Cliente;
  EXEMPLO <nome, local e email>:
  `João Neto - SPLAB - joaoneto@ccc.ufcg.edu.br`
- Retornar a representação textual de todos os clientes cadastrados no sistema (use o separador | para separar os clientes retornados);
  EXEMPLO:
  `Ana Silva - Embedded - anasilva@ccc.ufcg.edu.br | João Neto - SPLAB - joaoneto@ccc.ufcg.edu.br`

- Editar cadastro de um cliente (devido a unicidade de CPF, este campo não pode ser editado);
- Remover cliente do cadastro;

US2. CRUD de Fornecedores

User story: Como administrador do sistema, quero adicionar, recuperar, editar ou apagar fornecedores.

O sistema SAGA também mantém o cadastro dos fornecedores dos lanches. Os fornecedores são identificados unicamente pelo seu **nome** e não podem ser cadastrados no sistema em duplicidade. Sempre que for feita uma operação sobre um fornecedor, este deve ser recuperado através desta chave (**nome**). Além do nome, outras informações necessárias para o cadastro do fornecedor são: email e telefone. Cada fornecedor mantém a sua lista dos produtos que comercializa. Informações mais detalhadas sobre os produtos, serão apresentadas no US3.

A Facade do sistema deve ter métodos para:

- Cadastrar fornecedores;
  Em uma adição bem sucedida, o **nome** do fornecedor deve ser retornado. Quando não é bem sucedida, uma exceção deve ser lançada e a mensagem de erro mostrada para o usuário. Após isso, o programa **não** deverá continuar.
- Retornar a representação textual de um Fornecedor;
  EXEMPLO <nome, email e telefone>:
  `Dona Inês - dines@gmail.com - 83 9999-5050`
- Retornar a representação textual de todos os fornecedores cadastrados no sistema (use o separador | para separar os fornecedores retornados);
  EXEMPLO:
  `Dona Inês - dines@gmail.com - 83 9999-5050 | Josenilda -`

`nilda@computacao.ufcg.edu.br - 83 98736-5050 | Ron Weasley - rweasley@splab.ufcg.edu.br - 83 99936-5050`

- Editar cadastro do fornecedor (Devido a unicidade de nome, este campo não pode ser editado);
- Remover fornecedor do cadastro;

US3. CRUD dos Produtos dos Fornecedores

User story: Como administrador do sistema, quero adicionar, recuperar, editar ou apagar produtos de fornecedores.

Os produtos comercializados são de responsabilidade dos fornecedores. Assim, todo o controle dos produtos está sempre associado a um fornecedor específico. O sistema SAGA permite o cadastro de produtos para os fornecedores já existentes no sistema. Inicialmente, trataremos nesta US de produtos simples. Entretanto, nos próximos casos de uso os produtos podem tornar-se entidades mais complexas: com especializações e comportamentos distintos.
Produtos devem ter preço, nome e descrição. Não deve ser possível cadastrar o mesmo produto para o mesmo fornecedor. Os produtos são identificados por seu nome e descrição.

A Facade do sistema deve ter métodos para (em todas as funcionalidades abaixo o fornecedor é identificado pelo seu **nome** e o produto pelo **nome e descrição**):

- Cadastrar produto para um fornecedor;
- Consultar um produto de um fornecedor deve retornar a representação textual:
  <nome, descrição e preço>
  EXEMPLO:
  `Tapioca simples - Tapioca com manteiga - R$3,00`
- Consultar todos os produtos para **um** dado fornecedor cadastrado no sistema deve retornar a representação textual:
  (use o separador | para separar os dados retornados);
  EXEMPLO:
  `Dona Inês - Bolo - Bolo de chocolate - R$3,00 | Dona Inês - Tapioca simples - Tapioca com manteiga - R$3,00 | Dona Inês - Tapioca completa - Tapioca com côco, queijo e manteiga - R$3,50`
- Consultar todos os produtos de **todos** os fornecedores cadastrado no sistema, deve retornar a representação textual:
  (use o separador | para separar os fornecedores e produtos retornados);
  EXEMPLO:
  `Dona Inês - Tapioca simples - Tapioca com manteiga - R$3,00 | Dona Inês - Tapioca completa - Tapioca com coco, queijo e manteiga - R$3,50 | Dona Inês - Bolo - Bolo de chocolate - R$3,00 | Josenilda - Mousse - Mousse de Limão - R$4,00 | Josenilda - Salada - Salada de frutas com leite condensado - R$4,50 | Josenilda - Biscoito doce - Maizena - R$3,00`
- Editar produto (Só é possível alterar o **preço**. Para outras alterações sugere-se excluir o produto e inseri-lo posteriormente);
- Remover produto do fornecedor;

US4. Ordenação das listagens

User story: Como administrador do sistema, quero visualizar os Clientes, Fornecedores e Produtos de Fornecedores em ordem alfabética.

Espera-se que as listagens das entidades do sistema sejam em ordem alfabética. De modo que:

- Ao retornar a representação textual de todos os clientes cadastrados no sistema, estes devem ser apresentados em ordem alfabética, separados por "|";
  EXEMPLO:
  `Ana Silva - Embedded - anasilva@ccc.ufcg.edu.br | João Neto - SPLAB - joaoneto@ccc.ufcg.edu.br`
- Ao retornar a representação textual de todos os fornecedores cadastrados no sistema, estes devem ser apresentados em ordem alfabética, separados por "|":
  EXEMPLO:
  `Dona Inês - dines@gmail.com - 83 9999-5050 | Josenilda - nilda@computacao.ufcg.edu.br - 83 98736-5050 | Ron Weasley - rweasley@splab.ufcg.edu.br - 83 99936-5050]`
- Ao retornar a representação textual dos produtos de **todos** os fornecedores cadastrado no sistema, estes devem ser apresentados em ordem alfabética do fornecedor;
  EXEMPLO:
  `Dona Inês - Tapioca simples - Tapioca com manteiga - R$3,00 | Dona Inês - Tapioca completa - Tapioca com coco, queijo e manteiga - R$3,50 | Dona Inês - Bolo - Bolo de chocolate - R$3,00 | Josenilda - Mousse - Mousse de Limão - R$4,00 | Josenilda - Salada - Salada de frutas com leite condensado - R$4,50 | Josenilda - Biscoito doce - Maizena - R$3,00`

US5. Cadastrar uma compra em uma Conta (realizar compra)

User story: Como administrador do sistema, quero adicionar a compra de um produto na conta que um cliente mantém para um dado fornecedor.

Os clientes mantém Contas para os fornecedores dos quais eles compram. Quando o cliente compra um produto ele anota na Conta do fornecedor a sua Compra (data, identificação do produto e preço). Caso seja a primeira compra naquele fornecedor, a Conta do cliente para o dado fornecedor deve ser criada. A Conta mantém o débito do cliente para o fornecedor. O débito é o somatório do preço de todas as compras, realizadas, ou seja, que estão anotados naquela conta.
OBS: Para que a exclusão de um produto não gere inconsistência na Conta do Cliente, a Compra deve guardar as informações do Produto e não uma referência a ele.

A Facade deve ter métodos para:

- Cadastrar uma compra de produto de um fornecedor. É a operação de compra da facade. Isto significa incluir uma compra em uma Conta. Deve ser informado o cliente (identificado pelo CPF), o fornecedor (identificado por nome), data e as informações do produto (nome, descrição). Para a compra de mais de um item do mesmo produto deve ser criada uma nova compra;
  EXEMPLO: <cpf, fornecedor, data, nome_prod, desc_prod>

- Totalizar Conta de um fornecedor para um dado cliente. É a operação de *getDebito,* da facade. Deve ser informado o cliente (identificado pelo CPF), o fornecedor (identificado por nome). É retornado o valor decimal que representa a totalização das compras realizadas pelo cliente com o fornecedor.
  EXEMPLO: <cpf, fornecedor>

- Retornar a representação textual da conta do cliente (identificado pelo CPF) com um fornecedor (identificado pelo nome).
  EXEMPLO: <nome do cliente, fornecedor>
  "Cliente: João Neto | Dona Inês | Lanche FIT - 01-11-2018 | Salada - 29/10/2018"
  Para facilitar o entendimento, mostramos como seria esta saída em um front-end em linha de comando, por exemplo:

```
Cliente: João Neto
Dona Inês
Lanche FIT - 01-11-2018
Salada - 29/10/2018
```

  Lembre que você não vai fazer front-end!

- Retornar a representação textual de todas as contas de um cliente (identificado pelo CPF).
  EXEMPLO: <nome do cliente>
  "Cliente: João Neto | Dona Inês | Lanche FIT - 01-11-2018 | Salada - 29/10/2018 | Josenilda | Biscoito salgado - 01-11-2018"
  Para facilitar o entendimento, mostramos como seria esta saída em um front-end em linha de comando, por exemplo:

```
Cliente: João Neto
Dona Inês
Lanche FIT - 01-11-2018
Salada - 29/10/2018
Josenilda
Biscoito salgado - 01-11-2018
```

  Lembre que você não vai fazer front-end!

US6. Listar Compras de um Cliente

User story: Como administrador do sistema, quero listar as compras realizadas no SAGA de formas diversas.

É importante que sistemas de cadastro de entidades sejam capazes de retornar seus elementos em determinada ordem. Para o sistema SAGA, não seria diferente. Vários sistemas trabalham com a emissão de relatórios que servem como entrada para atividades de análise de dados.
O sistema tem acesso de forma global a todos os clientes, suas contas e por consequência às compras por eles realizadas.
Assim, pode ser realizada a listagem de todas as compras cadastradas de acordo com os critérios: Cliente, Fornecedor e Data.

- **Cliente:** Devem ser listadas todas as compras realizadas em cada conta de fornecedor (*ordenados em ordem alfabética*) dos clientes cadastrados. Após a ordenação pelo nome do cliente, as compras devem ser ordenadas em ordem alfabética das strings formadas pela concatenação de (<fornecedor>, <desc_prod>, <data_compra>);
- **Fornecedor:** Devem ser listadas todas as compras para cada fornecedor cadastrado (*ordenados em ordem alfabética*). Após a ordenação pelo nome do fornecedor, as compras devem ser ordenadas em ordem alfabética das strings formadas pela concatenação de (<cliente>, <desc_prod>, <data_compra>);
- **Data:** Devem ser listadas todas as compras para cada data (*ordenados em crescente*). Após a ordenação pela data, as compras devem ser ordenadas em ordem alfabética das strings formadas pela concatenação de (<cliente>, <fornecedor>, <desc_prod>);

A Facade deve ter métodos para:

- Definir critério de ordenação.
    - void ordenaPor(String criterio)
- Retornar a listagem das compras de acordo com o critério de ordenação definido anteriormente.
    - String listarCompras()

Para facilitar o entendimento, mostramos como seria a saída de uma **ordenação por cliente** em um front/end em linha de comando.

EXEMPLO:

"João Neto, Dona Inês, Lanche FIT, 01/11/2018 | João Neto, Dona Inês, Salada, 29/10/2018 | João Neto, Josenilda, Biscoito salgado, 01/11/2018 | João Neto, Josenilda, Biscoito doce, 28/10/2018 | Kleberson, Josenilda, Biscoito doce, 15/05/2018 | Zana, Dona Inês, Lanche FIT, 10/06/2018"

```
João Neto, Dona Inês, Lanche FIT, 01/11/2018
João Neto, Dona Inês, Salada, 29/10/2018
João Neto, Josenilda, Biscoito salgado, 01/11/2018
João Neto, Josenilda, Biscoito doce, 28/10/2018
Kleberson, Josenilda, Biscoito doce, 15/05/2018
Zana, Dona Inês, Lanche FIT, 10/06/2018
```

Lembre que você não vai fazer front/end!

Para facilitar o entendimento, mostramos como seria a saída de uma **ordenação por fornecedor** em um front/end em linha de comando.

EXEMPLO:

"Dona Inês, João Neto, Lanche FIT, 01/11/2018 | Dona Inês, João Neto, Salada, 29/10/2018 | Dona Inês, Zana, João Neto, Lanche FIT, 10/06/2018 | Josenilda, João Neto, Biscoito salgado, 01/11/2018 | Josenilda, João Neto, Biscoito doce, 28/10/2018 | Josenilda, Kleberson, Biscoito doce, 15/05/2018"

```
Dona Inês, João Neto, Lanche FIT, 01/11/2018
Dona Inês, João Neto, Salada, 29/10/2018
Dona Inês, Zana, João Neto, Lanche FIT, 10/06/2018
Josenilda, João Neto, Biscoito salgado, 01/11/2018
Josenilda, João Neto, Biscoito doce, 28/10/2018
Josenilda, Kleberson, Biscoito doce, 15/05/2018
```

Lembre que você não vai fazer front/end (gui/main/menu)!

Para facilitar o entendimento, mostramos como seria a saída de uma **ordenação por data** em um front/end em linha de comando.

EXEMPLO:

"15/05/2018, Kleberson, Josenilda, Biscoito doce | 10/06/2018, Zana, Dona Inês, Lanche FIT | 28/10/2018, João Neto, Josenilda, Biscoito doce | 29/10/2018, João Neto, Dona Inês, Salada | 01/11/2018, João Neto, Dona Inês, Lanche FIT | 01/11/2018, João Neto, Josenilda, Biscoito salgado"

```
15/05/2018, Kleberson, Josenilda, Biscoito doce
10/06/2018, Zana, Dona Inês, Lanche FIT
28/10/2018, João Neto, Josenilda, Biscoito doce
29/10/2018, João Neto, Dona Inês, Salada
01/11/2018, João Neto, Dona Inês, Lanche FIT
01/11/2018, João Neto, Josenilda, Biscoito salgado
```

Lembre que você não vai fazer front/end (gui/main/menu)!

Entrega

Faça o sistema SAGA com as funcionalidades descritas nas US1, US2, US3, US4, US5 e US6. É importante que o código esteja devidamente documentado.

Ainda, você deve entregar um programa com testes de unidade para as classes com lógica testável. Tente fazer um programa que possa ser devidamente testado e que explore as condições limite e as regras de negócio apresentadas.

A facade do seu sistema deve ser capaz de testar cada uma das US já especificadas com os testes de aceitação definidos pela equipe aqui.

Faça um **zip** da pasta do seu projeto. Coloque o nome do projeto para:

- LAB5_SEU_NOME e o nome do zip para LAB5_SEU_NOME.zip. Exemplo de projeto: LAB5_LIVIA_SAMPAIO.zip. Este zip deve ser submetido pelo Canvas.

Seu programa será avaliado pela corretude e, principalmente, pelo DESIGN do sistema. É importante:

- Usar nomes adequados de variáveis, classes, métodos e parâmetros.

- Fazer um design simples, legível e que funciona. É importante saber, apenas olhando o nome das classes e o nome dos métodos existentes, identificar quem faz o que no código.