MC322 – Programação Orientada a Objetos
Laboratório 04 – 1s2021
TESTE PRÁTICO 1

Leonardo Montecchi (Professor) Junior Cupe Casquina Fillipe dos Santos Silva
Natan Rodrigues de Oliveira Caio Henrique Pardal Leonardo Yoshida

Para perguntas ou dúvidas usem o Discord (`https://discord.gg/KmZNzXNVhM`)

1 Submissão

Data de entrega

- **25/04/2021** até às **23h59**.

Submissão

- Ao criar o projeto Java no Eclipse, selecionar a versão **JavaSE-11** no JRE
- IMPORTANTE: Nomear o projeto na forma **RA_Lab04** e o pacote base na forma **com.unicamp.mc322.lab04**. Substitua RA com o seu *Registro Acadêmico* (matrícula).
- Submeta o trabalho no link de entrega na página do Classroom da disciplina, em formato de arquivo compactado (zip) com o nome **RA_Lab04.zip**. Substitua RA com o seu *Registro Acadêmico* (matrícula). Entregas feitas de outras formas não serão consideradas.
- O arquivo compactado deve conter o projeto inteiro ("File / Export" no Eclipse, ou crie o arquivo manualmente).

Critérios de avaliação

- **Este laboratório VALE nota.**

  Faz parte da avaliação saber quais classes devem ser criadas e em quais classes cada método deve ser implementado. Em particular, o código será avaliado de acordo com os seguintes aspectos:

  1. Definição de Classes (30%).
  2. Aplicação de princípios de Programação Orientada a Objetos (POO) (30%).
     - Ex: *encapsulamento* de atributos, *responsabilidades* de classes.
  3. Funcionalidades implementadas (40%).

2 Exercício

2.1 Especificação do Sistema

O Pidão é uma solução de gerenciamento de pedidos para restaurantes, visando ser uma opção acessível para os clientes pedirem comida em casa ou no trabalho. O aplicativo é configurado pelo gerente do restaurante e implementa as seguintes funcionalidades.

- O aplicativo é configurado com os detalhes do restaurante: nome, CNPJ, e posição no mapa. Por simplicidade considere que a posição é um par de coordenadas (X, Y).

- O aplicativo permite o cadastro de usuários, que é feito informando nome, CPF, e endereço de entrega. Por simplicidade considere que o endereço é armazenado em forma de posição (X, Y).
- O aplicativo permite adicionar e remover itens ao cardápio do restaurante. Cada item contém nome, preço e um identificador alfanumérico de 5 dígitos. O identificador deve ser único.
- O aplicativo permite imprimir o cardápio completo do restaurante.
- O aplicativo permite definir (e remover) um desconto temporário para itens específicos do cardápio, que pode ser em porcentagem do preço base ou em valor fixo.
- O aplicativo permite registrar pedidos dos usuários. Cada pedido pode incluir um ou mais itens do cardápio do restaurante.
- Um pedido começa no estado "novo" e passa em seguida pelos estados "em preparação", "saiu para entrega" e "entregue". Um pedido pode ser cancelado apenas se estiver nos estados "novo" ou "em preparação". O aplicativo deve permitir alterar o estado dos pedidos de acordo com o fluxo acima.
- O valor total do pedido é a soma dos preços atuais de cada item (isto é, considerando o preço com desconto se estiver ativo), menos eventuais descontos adicionais, se houverem. O cálculo do valor do pedido é feito quando o pedido entra no estado "em preparação".
- Se for a primeira compra do cliente, deverá ser oferecido um desconto de 20% ao valor total do pedido. A cada dez compras do cliente o sistema oferecerá um desconto de 100% do valor do pedido, com valor limitado a 60 R$. Nos demais casos, se o preço total do pedido for maior que 100 R$, o sistema deverá aplicar um desconto de 10% ao valor total do pedido.
- O sistema é capaz de imprimir um resumo de todos os pedidos a qualquer momento.

2.2 Exemplo de fluxo de Execução

Considere o seguinte cenário como um exemplo de fluxo de execução. Esse código deve ser considerado como uma das possíveis execuções do programa. Porém, ele também define um contrato do que deve ser implementado: deve ser possível rodar esse trecho de código na sua solução.

```
public class Runner {

  public static void main(String[] args) {
    Pidao pidaoApp = new Pidao("MARAMBAR", "123.456.789-10", 10, 2);

    User user1 = pidaoApp.cadastrarUsuario("Marcos Paulo", "123.789.643-11", 1, 2);
    User user2 = pidaoApp.cadastrarUsuario("Pereira", "123.789.643-11", 8, 4);

    Lanche cuscuz = new Lanche("CCZ00", "Cuscuz com ovo", 10.00);
    Lanche macaxeira = new Lanche("MXCOS", "Macaxeira com costela no bafo", 15.00);
    Lanche coxinhaFrango = new Lanche("CXFRA", "Coxinha de frango", 8.00);

    pidaoApp.adicionarAoCardapio(cuscuz);
    pidaoApp.adicionarAoCardapio(macaxeira);
    pidaoApp.adicionarAoCardapio(coxinhaFrango);

    pidaoApp.aplicarDesconto("CCZ00", 10, TipoDesconto.PORCENTAGEM);

    Pedido p1 = new Pedido(user1);
    p1.addItem(cuscuz);
    p1.addItem(macaxeira);
```

```
22      pidaoApp.fazerPedido(p1);
23
24      Pedido p2 = new Pedido(user2);
25      p2.addItem(coxinhaFrango);
26      p2.addItem(coxinhaFrango);
27      pidaoApp.fazerPedido(p2);
28
29      Pedido p3 = new Pedido(user2);
30      p3.addItem(coxinhaFrango);
31      p3.addItem(coxinhaFrango);
32      pidaoApp.fazerPedido(p3);
33
34      pidaoApp.imprimirCardapio();
35
36      pidaoApp.imprimirResumoPedidos();
37    }
38
39  }
```

Neste exemplo, temos dois clientes, Marcos e Pereira, e três itens no cardápio. Temos no total três pedidos, dois feitos por Pereira e um feito por Marcos. A saída esperada baseada no exemplo de fluxo de execução descrito anteriormente é a seguinte.

```
1  Restaurante MARAMBAR
2  (CNPJ: 123.456.789-10)
3
4  Cardapio de hoje:
5  [CCZ00] Cuscuz com ovo R$ 9.0 (PROMO  O! Pre o normal: R$ 10.0)
6  [MXCOS] Macaxeira com costela no bafo R$ 15.0
7  [CXFRA] Coxinha de frango R$ 8.0
8
9  Existem 3 pedidos:
10 ==========================================
11 Usu rio: Marcos Paulo (123.789.643-11)
12  - CCZ00
13  - MXCOS
14 Valor Total: R$ 19.2
15 ==========================================
16 Usu rio: Pereira (123.789.643-11)
17  - CXFRA
18  - CXFRA
19 Valor Total: R$ 12.8
20 ==========================================
21 Usu rio: Pereira (123.789.643-11)
22  - CXFRA
23  - CXFRA
24 Valor Total: R$ 16.0
25 ==========================================
```

Vale a pena ressaltar que **os critérios de avaliação não são baseados apenas na correta produção da saida acima.** Os critérios de avaliação estão descritos no início deste documento.

3 Boas Práticas

- Nomes de classes devem começar com letra maiúscula;
- Nomes de variáveis e métodos devem começar com letra minúscula;
- Nomes de classes devem ser substantivos;

- Nomes de métodos devem ser verbos ou começar com verbo;
- Nomes compostos por mais de uma palavra devem ser escritos na forma *CamelCase*. Isto é, com a primeira letra de cada palavra maiúscula. Por exemplo: "get something" deve ser escrito como `getSomething` (método), e uma classe que representa *something* poderia ser declarada como `Something`.