UAST Unidade Acadêmica de Serra Talhada - PE — UFRPE

Bacharelado em Sistemas de Informação

MPOO

Disciplina: Modelagem e Programação Orientada a Objetos (MPOO)

Profº: Richarlyson D'Emery **Data:** 04 / 09 / 2023

Aluno: _____

Nota:_____

2ª VERIFICAÇÃO DE APRENDIZAGEM

Instruções gerais:

- Utilize um diretório para salvar as implementações. Salve as implementações a cada modificação, caso aconteça alguma falha de energia o trabalho será preservado. Lembre-se que uma vez removido o arquivo do Eclipse, seu conteúdo será perdido.
- *A Nota máxima desta prova é de 10,0 pontos*. A Pontuação da Prova está distribuída entre os conceitos vistos na disciplina MPOO CCMP5012: (i) Projeto Eclipse e organização, (ii) Manipulação de Exceção, (iii) interface, (iv) composição, (v) Thread, (vi) Componentes gráficos, (vii) Eventos e tratamentos e (viii) Design Pattern.

---

1) No Eclipse limpe todos os projetos existentes.
   - Importe o **projeto** disponibilizado no SIGAA:
     br.2va.mpoo.edu.NomeSobrenome
     Este possui **uma pasta de pacotes chamada MPOOStore** que deverá conter todos os arquivos necessários para as respectivas questões.
   - Ao finalizar a prova **compacte o projeto** contendo toda a codificação do projeto (arquivos texto, *bytecodes e* imagens) e envio-o no SIGAA:
     [2ª Verificação de Aprendizagem] em Semana 15
     O SIGAA aceitará submissões até às 18h

2) O proprietário de MPOOStore precisa de um sistema para distribuir cupons de descontos aos seus clientes, o qual será disponibilizado em um totem recém adquirido. O sistema é descrito nas questões abaixo, modelado no APÊNDICE A e deverá ser implemente em Java.

***Design* Pattern (0,5 ponto)**

a) Utilize o padrão de arquitetura de projeto Model-View-Controller (MVC).

b) Algumas definições do modelo do diagrama de classes do Apêndice A já são disponibilizadas. É descrição:
   - Respeite as definições do diagrama.
   - BaseDados é uma classe que contém uma estrutura de dados para as pessoas do sistema. **APENAS** devem ser implementados os métodos apresentados no diagrama. Toda lógica necessária deve estar presentes exclusivamente nesses métodos;
   - A codificação deve aproveitar comportamentos já definidos, evitando a duplicidade de programação;
   - Há apenas uma base no sistema;
   - As pessoas do sistema podem ser pessoas do tipo Cliente;
   - Um cliente é identificado pelo seu CPF e email;
   - Um cliente só poderá ser cadastrado uma única vez;
   - Os clientes cadastrados podem ganhar um cupom de desconto que poderá ser usado em futuras compras;

Composição (1,0 ponto)

   - É relação de composição entre Cliente (todo) e Cupom Desconto (parte)

Interface (1,0 ponto)

   - Um cupom para ser utilizado deverá ser validado. A validação é comportamento da Interface ValidarCupomInterface, de maneira que:

`se codigoCupom é igual a "MPOOSTOREOFF" então é verdadeiro`

   - Uma Pessoa poderá ter o CPF validado pela interface ValidadorCPFInterface. Se inválido, a pessoa não poderá fazer parte do sistema. O método de validação está presente na codificação disponibilizada.

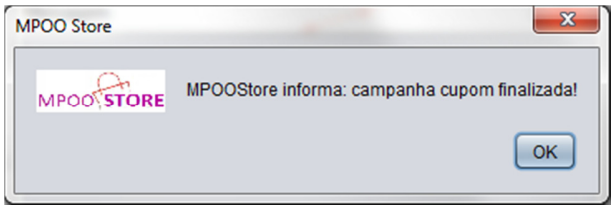
Thread (1,5 pontos)

c) O sistema possui dois robôs. TempoOferta mantém uma oferta ativa, enquanto GeradorCupom gera um novo valor de Cupom.
O tempo de oferta será de 10 minutos (600000 ms), após esse tempo um cliente não poderá mais resgatar um cupom.
A geração de cupom é dada a cada 5 segundos, com valores que variam de R$ 0 a R$ 100,00:

```
new Random().nextInt(100);
```

Mas atenção:
Não se deve confundir uma geração automática realizada por um robô *started* em uma aplicação com a possibilidade de definir valores diretamente em Cupom de Desconto, um cupom deve utilizar o valor de valorCupomAtual (atributo de GerenciadorCupom).

Depois de finalizado o tempo de oferta o sistema deverá informar que será encerrado.



Métodos e Manipulação de Exceção (1,5 pontos)

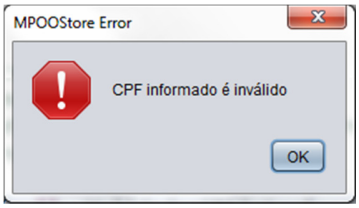
d) Faça a devida manipulação de Exception quando um cliente, CPF ou cupom é inválido.

Em chamadas de métodos deve-se utilizar `try-catch`

Utilize as cláusulas `throws-throw`:

   - Em BaseDados modifique `isCliente(Cliente cliente)` de maneira a levantar a exceção ClienteException quando não for um cliente válido;
   - No construtor de CupomDesconto quando se está tentando criar um cupom de código inválido (`CupomException`) ou atribuir um cupom válido a um cliente que não faz parte da base (`ClienteException`);
   - Ao tentar instanciar uma pessoa com cpf inválido (`CPFException`).

Personalize mensagens para notificar o usuário do sistema quando uma exceção ocorrer, por exemplo:



(Bônus) Relançamento de Exceção e seu Tratamento (0,5 ponto)

Reflita o Relançamento de Exceção para a não violar o MVC, de maneira que o model não exiba tratamento de erro com apresentação de view, consequentemente, passando a responsabilidade para os pacotes app e/ou controller.

Componentes Gráficos (2,0 pontos)

É descrição das GUI's do sistema MPOOStore:

e) Possui uma tela a ser exibida no Totem para interação do usuário:

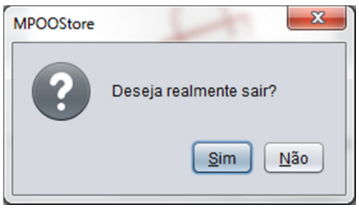
(AberturaView: 300x200)

(300x350)

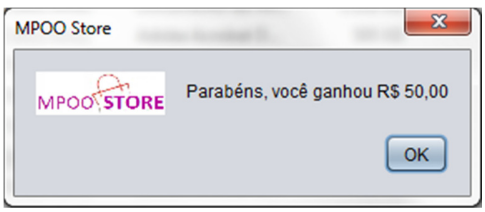
ou

Se um cliente não cadastrado tentar ganhar um cupom ele receberá o aviso da impossibilidade de ganhar um cupom, caso contrário deverá confirmar seus dados.

A tela GanharCupomView não encerra o sistema, logo se fechada (ou tecla ESC) o sistema deverá voltar a exibir AberturaView (esta última se fechada deverá encerrar o sistema após confirmação do usuário). Há apenas uma janela ativa e uma única instância ativa para cada tela.



Quando um cliente confirma seus dados e ganha um cupom é exibida a mensagem:



Obs.: O valor do cupom depende do valor que o cliente ganhou gerado pelo robô GeradorCupom e disposto em GerenciadorCupom.

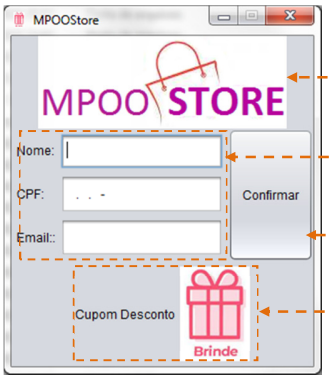
Utilize Look and feel:

```
UIManager.setLookAndFeel("javax.swing.plaf.nimbus.NimbusLookAndFeel")
```

(Bônus) SpringLayout (0,25 ponto)

É organização das telas:

- AberturaView: FlowLayout
- GanharCupomView:



Dicas:

JLabel em BorderLayout.*NORTH* do JFrame

Componentes em SpringLayout de JPanel disposto no JFrame em BorderLayout.*EAST*

JButton do JFrame em BorderLayout.*EAST*

Componentes e em JPanel disposto no JFrame em BorderLayout.*SOUTH*

São componentes:

JFrame, JPanel, JLabel, ImageIcon, JTextField, JFormattedTextField, JRadioButton, JButton e JOptionPane

Dica:

```
JFormattedTextField cpfField;
try {
     cpfField = new JFormattedTextField(
                                  new MaskFormatter("###.###.###-##")
                                        );
     cpfField.setColumns(10);
    } catch (ParseException e) {}
```

Eventos e Tratamentos (2,0 pontos)
(Bônus) Design Pattern: Adapter (0,25 ponto)

f) Os tratamentos dos eventos das telas (mpooStore.view) devem estar nos respectivos controladores (mpooStore.controller) conforme diagrama de classes do Apêndice A:

- Para AberturaView tem-se AberturaController, sendo:
  - Tratamento de evento por classe interna privada:
    - RadioHandler: Para saber se é cliente de MPOOStore;
    - ButtonHandler: Quando o cliente aciona o botão PegarCupom;
    - KeyHandler: Para tratar o encerramento do sistema por ESC.
- Para GanharCupomView tem-se GanharCupomController, sendo:
  - Tratamento de evento realizado por método sobrescrito por classe (classe realiza a interface):
    - botão Confirmar: Confirma o cupom de desconto para um cliente.

(Bônus) Eventos e Tratamentos (0,25 ponto) e Design Pattern: Adapter (0,25 ponto)

Observe que na tela GanharCupomView não há botão para voltar para a tela anterior:

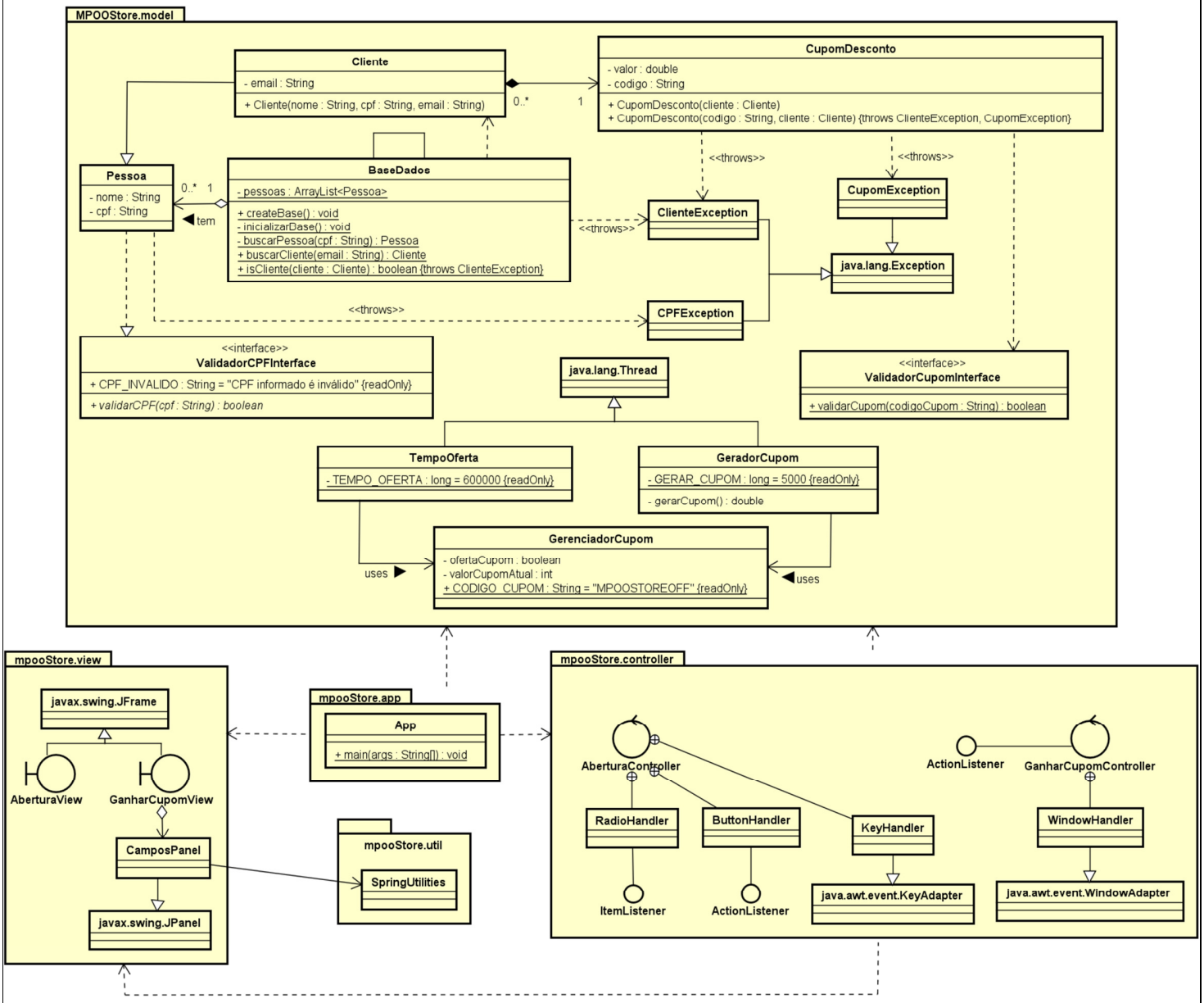
- Tratamento de evento por classe interna privada:
  - WindowHandler: Para saber se a tela foi fechada.

Manipulação das definição

g) Deve-se ter em App:

- A criação dos robôs (Threads)
- A criação das instâncias MVC

APÊNDICE A




```java
public class Window{
    JButton button = new JButton();
    public static void main(String[] args) {
        Window window = new Window();
        window.button.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                JOptionPane.showMessageDialog(null, "Boa Prova!");
            }
        });
        window.button.doClick();
    }
}
```