UAST
Unidade Acadêmica de Serra Talhada - PE
UFRPE

MPOO

site: https://sites.google.com/site/profricodemery/mpoo

Disciplina: Modelagem e Programação Orientada a Objetos (MPOO)

Profº: Richarlyson D'Emery **Data:** 17 / 04 / 2023

Aluno: Marco Antonio de Sousa Santos

Nota:_____

3ª V.A. - TEÓRICO-PRÁTICA (CODIFICAÇÃO)

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.

Nesta 3ª V.A. você estará sendo avaliado quando aos assuntos da disciplina de MPOO. *A Nota máxima desta prova é de 10 pontos e tem peso 1,0.*

Criação e Configuração de Projeto (0,25 ponto)

1) No Eclipse limpe todos os projetos existentes.
   - Crie um novo **projeto chamado**
     br.3va.mpoo.edu.NomeSobrenome
     Este deverá ter **uma pasta de pacotes chamada mpooSystem** contendo todos os arquivos necessários para as respectivas questões.
   - Ao finalizar a prova **compacte o projeto** contendo toda a codificação do projeto (arquivos texto (**.java**), *bytecodes* ***(.class)*** e imagens) e envio-o no AVA :
     - [3ª Verificação de Aprendizagem] em Semana 16

2) O sistema descrito abaixo é modelado no APÊNDICE A e deverá ser implemente em Java.

Design Pattern e complementos (0,5 ponto)

a) Utilize o padrão de arquitetura de projeto: Model-View-Controller (MVC). Utilize os pacotes: model, view, controller e app.

b) Importe os arquivos disponibilizados validarCPF.txt, logo.png e icone.png

Classe, atributo, Método (construtor e concreto) e Encapsulamento (1,0 ponto)
Herança (0,5 ponto)
interface (0,5 ponto
Sobreposição: (0,25 pontos)
Agregação e ArrayList, métodos e utilização (0,5 ponto)
Atributos e métodos static (0,5 ponto)
Polimorfismo de Objetos (0,5 ponto)
Composição (0,5 ponto)

c) Implemente o model do sistema. É descrição:
   - Respeite as definições do diagrama: classes, encapsulamentos, static e final, herança, agregação, realização e dependências;
   - Há herança entre Usuario e Pessoa.
   - Todos os dados de pessoas do sistema devem ser persistidos e utilizados a partir de BaseDados;
   - Há apenas uma base no sistema;
   - Faça o devido polimorfismo de objeto;
   - A BaseDados consiste de um ArrayList (vide letra j));

Uma pessoa para ser adicionada a Basedados deve verificar se o cpf é válido. Para isso utilize a interface ValidadorCPF. O método de validação de CPF `validarCPF(String cpf)` é disponibilizado no AVA.
   - Toda mensagem deve utilizar as constantes definidas;
   - Uso de toString para exibir dados não sigilosos. Apenas senha é um dado sigiloso.
   - Há relação de um para muitos entre pessoa e telefone, de maneira que uma pessoa possa ter diversos telefones.
   - O usuário poderá ser do tipo SuperUsuario, com tempo de login superior ao de um Usuário (vide letra m).
   - Os métodos `listPessoas()` e `listUsuarios()` de BaseDados retornam nomes e logins, respectivamente.

Sobrecarga de métodos (0,5 ponto)

d) Observe as diversas possibilidades de criação de um Usuário:
   - A senha é passada por parâmetro; ou
   - Um usuário criado possui a senha padrão: "123"

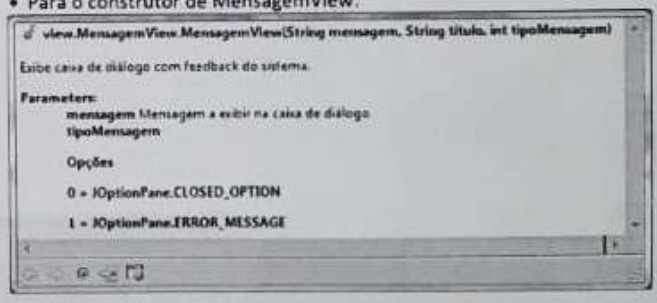
   Ob.: Ao criar um usuário passe como parâmetro para lastLogin o comando:
   `LocalDateTime.now()`

Tratamento de Exceção: 0,5 ponto

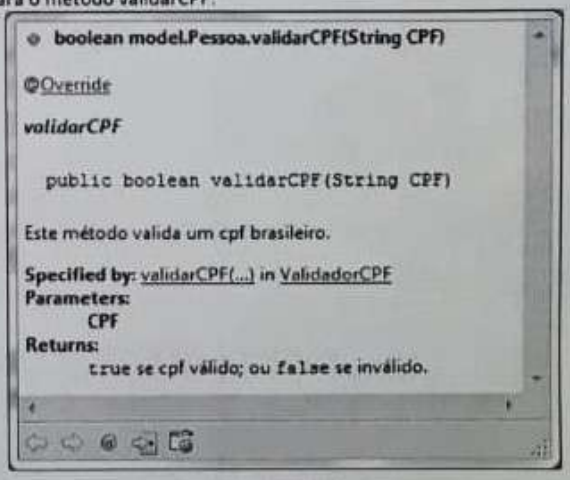
e) Os usuários do sistema estão armazenados no atributo estático *pessoas* da classe BaseDados. Faça o devido uso de ArrayList.
   O ArrayList deverá ser manipulado pelos métodos CRUD definidos no diagrama de classes.
   - Se uma pessoa for ser cadastrada com cpf inválido, levanta-se a exceção CPFException vinda de Pessoa.
     Faça o devido uso de `try - catch` nas chamadas dos métodos e `throw` e `throws` para o tratamento da Exceção
   - O tratamento consiste em exibir a caixa de diálogo da MSG01 (letra k)

Comments e métodos (0,25 ponto)

f) Adicione "doc comments":
   - Para o construtor de MensagemView:



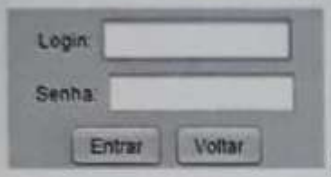
   - Para o método validarCPF:



Componentes gráficos (1,5 pontos)
Tratamento de Exceção (0,25 ponto)

g) São telas do sistema:

(250x170) (Fig. 1)



(200x100) (Fig. 2)

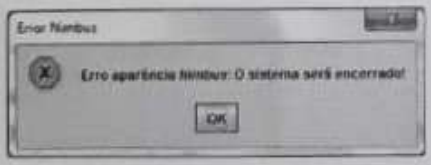
(155x380) (Fig. 3)

Que possui a seguinte descrição:

- *Look and feel:*

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

Utilize o bloco `try - catch` na personalização da aparência nimbus. Em caso de exceção, trate com a mensagem abaixo e encerre o sistema:



- Componentes: `JFrame`, `JLabel`, `JTextField`, `JFormattedTextField`, `JPasswordField`, `JRadioButton`, `JButton` e `JPanel`.
- Layout: FlowLayout.
- Figs. 1 a 3 são uma única instância, que devem ser visualizadas pelo usuário conforme sua utilização.
  - LoginView (Fig. 2) será exibida quando se clica na opção Logar de MenuView;
  - CadastrarView (Fig. 3) será exibida quando a opção cadastrar de MenuView for acionada;
  - Só deverá ter uma tela ativa. Os botões "Voltar" voltam para MenuView (Fig. 1).

Eventos e Tratamentos (3,5 pontos)

h) A opção Sair da Tela Menu encerra o sistema.

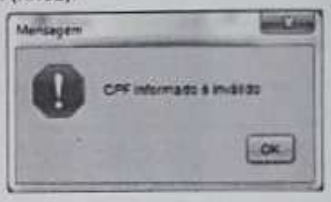
i) Os eventos dos botões devem ser implementados, em conformidade ao APÊNDICE A, da seguinte forma:
- Para cada tela há um controlador;
- MenuController o tratamento será realizado em classe interna anônima;
- LoginController possui tratamento em classe interna privada; e
- CadastroController o tratamento será realizado na própria classe.

Regras de negócio (1,0 ponto)

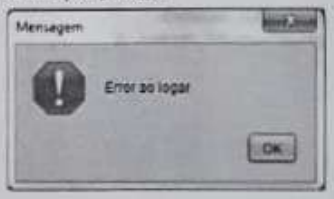
j) Os usuários fazem parte de uma base de dados. Faça a devida manipulação de ArrayList quanto a criação, consulta e autenticação de usuários.
São regras de Negócios:
- RN01: uma pessoa é identificado pelo seu cpf (chave primária);
- RN02: uma pessoa só é instanciada se tiver um cpf válido;
- RN03: a autenticação de um usuário é dada pelo seu login e senha;
- RN04: uma pessoa só poderá ser cadastrada uma única vez;
- RN05: para entrar no sistema um usuário deverá informar seu login e senha;
- RN06: a codificação deve aproveitar comportamentos já definidos, evitando a duplicidade de programação;
- RN07: um usuário só adicionado à base se for uma pessoa válida; e
- RN08: usuários logados sãoarmazenados em usuariosLogados de Log.

Mensagem e MVC (0,5 ponto)

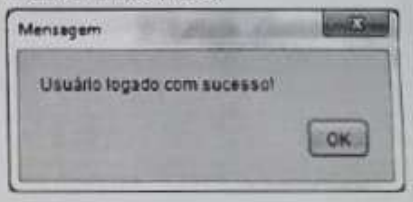
k) Exiba as mensagens abaixo em uma caixa de diálogos:
- MSG01: falha de cpf (RN02):



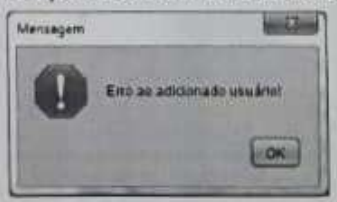
- MSG02: falha de autenticação (RN03):



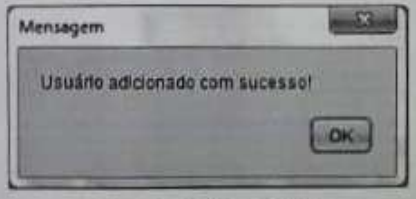
- MSG03: sucesso de autenticação (RN03):



- MSG04: sucesso de inserção de usuário na base de dados (RN07):



- MSG05: falha de inserção de usuário na base de dados (RN07):



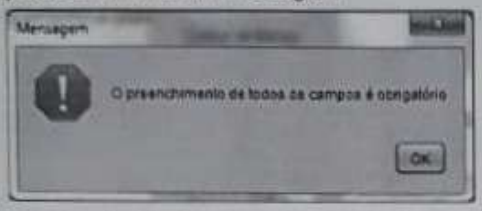
Tratamento de Exceção: 0,5 ponto

l) Implemente o método validarDados( ) no controlador de maneira que todos os campos de CadastroView sejam obrigatórios:

```
if (condição)
        throw new Exception();
```

*condição = comparação lógica de campos vazios

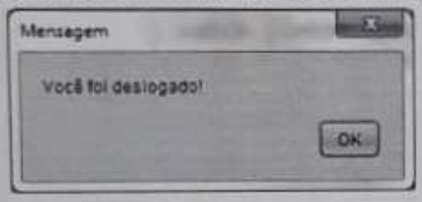
Utilize o bloco try - catch na chamada do método e throw e throws para o tratamento da Exceção. No tratamento da validação deve ser fornecida a mensagem:



Esse método será utilizado pelo botão cadastrar de CadastroView.

Thread: 1,0 ponto

m) Log é uma especialização de Thread responsável por deslogar os usuários logados. O tempo em que o usuário poderá estar logado é definido na constante TEMPO_LOG.
O método deslogar é disponibilizado no AVA.
Sempre que um usuário é deslogado, exibe-se a mensagem:



Manipulação das definições: pontuação distribuída nas conceituações

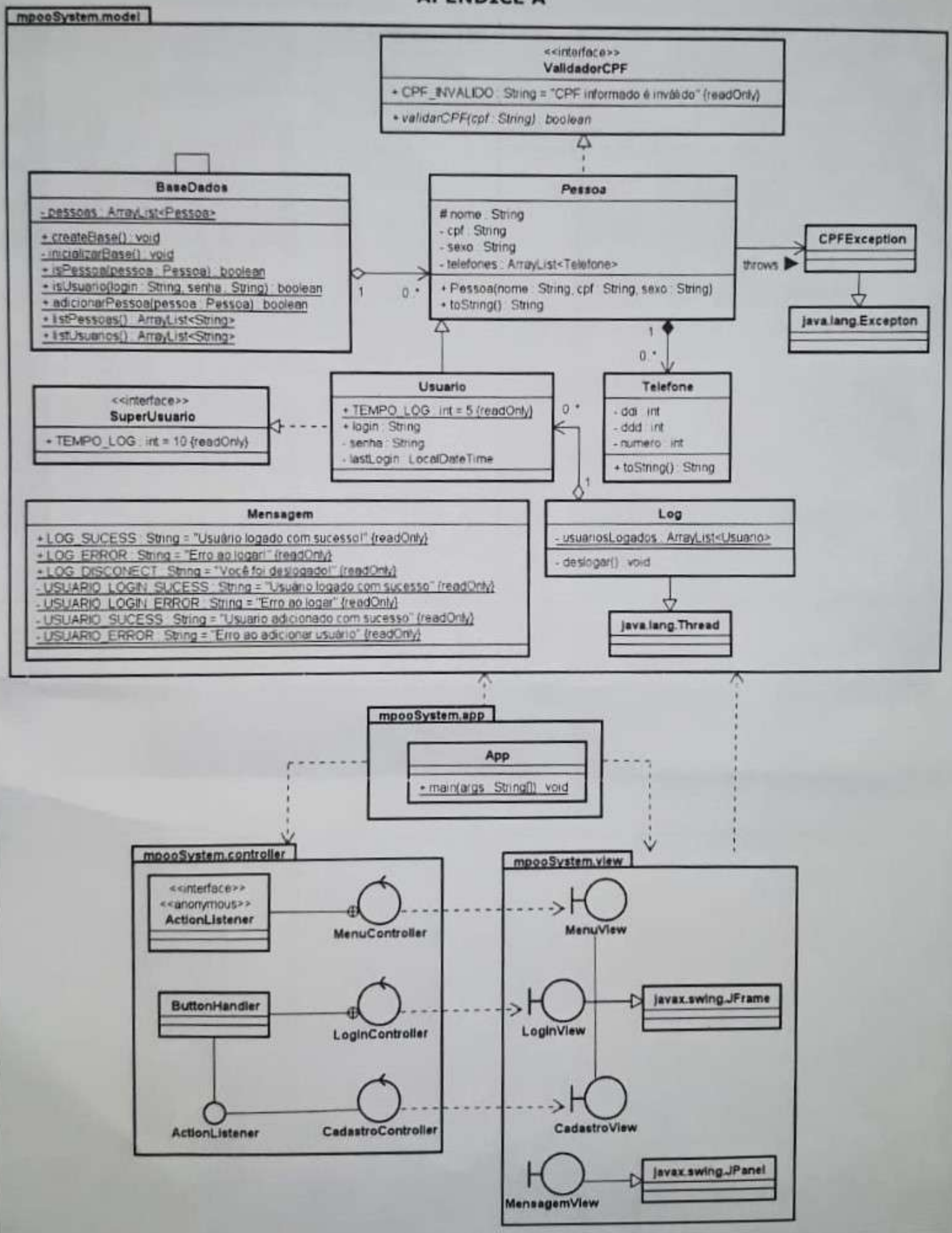
n) Deve-se ter em App:
- A criação de instâncias MVC;
- A criação e execução de thread;

Sem a utilização de telas e exibição em console:

- A criação dos usuários:
  - nome: "seu nome", cpf: "seu cpf", sexo: "seu sexo", login: "seunome", senha: admin e telefone: 55 87 99999-0000;
  - nome: Maria Silva, cpf: 833.533.163-34, sexo: feminino, login: mariasilva, senha: mAriA e telefone: 55 87 99999-11111;
  - nome: João Silva, cpf: 833.533.163-34, sexo: masculino, login: joaoSilva, senha: joaoSilva e telefone: 55 87 99999-2222;
  - usuário: nome: José Santos, cpf: 123.456.789-00, sexo: masculino, login: joseSantos, senha: joseSantos; e telefone 55 81 9999-3333;
- A criação do superUsuario:
  - nome: Godofredo Silva, cpf: 358.251.830-27, sexo: masculino, login: "admin", senha: "admin" e telefone: 55 87 99999-4444.
- A exibição dos dados dos usuários cadastrados por
  - toString; e
  - métodos list da base.

```
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();
    }
}
```

APÊNDICE A


mpooSystem.model
<<interface>>
ValidadorCPF
+ CPF_INVALIDO : String = "CPF informado é inválido" {readOnly}
+ validarCPF(cpf : String) : boolean
BaseDados
- pessoas : ArrayList<Pessoa>
+ createBase() : void
- inicializarBase() : void
+ isPessoa(pessoa : Pessoa) : boolean
+ isUsuario(login : String, senha : String) : boolean
+ adicionarPessoa(pessoa : Pessoa) : boolean
+ listPessoas() : ArrayList<String>
+ listUsuarios() : ArrayList<String>
1
0 *
Pessoa
nome : String
- cpf : String
- sexo : String
- telefones : ArrayList<Telefone>
+ Pessoa(nome : String, cpf : String, sexo : String)
+ toString() : String
throws
CPFException
java.lang.Excepton
1
0 *
<<interface>>
SuperUsuario
+ TEMPO_LOG : int = 10 {readOnly}
Usuario
+ TEMPO_LOG : int = 5 {readOnly}
+ login : String
- senha : String
- lastLogin : LocalDateTime
0 *
1
Telefone
- ddi : int
- ddd : int
- numero : int
+ toString() : String
Mensagem
+ LOG_SUCESS : String = "Usuário logado com sucesso!" {readOnly}
+ LOG_ERROR : String = "Erro ao logar!" {readOnly}
+ LOG_DISCONECT : String = "Você foi deslogado!" {readOnly}
- USUARIO_LOGIN_SUCESS : String = "Usuário logado com sucesso" {readOnly}
- USUARIO_LOGIN_ERROR : String = "Erro ao logar" {readOnly}
- USUARIO_SUCESS : String = "Usuario adicionado com sucesso" {readOnly}
- USUARIO_ERROR : String = "Erro ao adicionar usuário" {readOnly}
Log
- usuariosLogados : ArrayList<Usuario>
- deslogar() : void
java.lang.Thread
mpooSystem.app
App
+ main(args : String[]) : void
mpooSystem.controller
<<interface>>
<<anonymous>>
ActionListener
MenuController
ButtonHandler
LoginController
ActionListener
CadastroController
mpooSystem.view
MenuView
LoginView
javax.swing.JFrame
CadastroView
MensagemView
javax.swing.JPanel