2ª VERIFICAÇÃO DE APRENDIZAGEM

Leia atentamente as instruções gerais:

- No Eclipse limpe todos os projetos existentes. Crie um novo **projeto chamado br.2va.mpoo.edu.NomeSobrenome**, o qual deverá ter **a pasta de pacotes** "sistemaMercadinho" contendo todas as classes das questões.
- A prova é prática e deverão ser devolvidas tanto a prova impressa quanto os códigos-fonte implementados em Java.

Nesta 2ª V.A. você estará sendo avaliado quando aos assuntos: (i) Composição; (ii) Interface; (iii) Classes e métodos abstratos; (iv) Tratamento de Exceção; e (v) Thread.
A Nota máxima desta prova é de 10 pontos e tem peso 1,0.

No Eclipse limpe todos os projetos existentes.

- Crie um novo **projeto chamado**
  br.2va.mpoo.edu.NomeSobrenome
  Este deverá ter **a pasta de pacotes** "sistemaMercadinho".
- Ao finalizar a prova **compacte o projeto** contendo toda a codificação do projeto (arquivos texto **(.java)** e *bytecodes* ***(.class)***) e envio-o no AVA :
  [2ª Verificação de Aprendizagem] em Semana 15

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

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

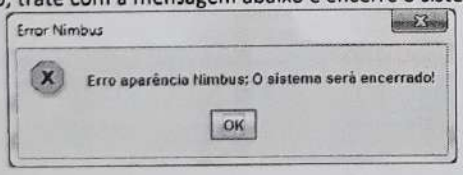
1) Implemente em Java:

Componentes gráficos e Tratamento de Eventos: Bônus (1,5 pontos)
Tratamento de Exceção: (0,5 ponto)

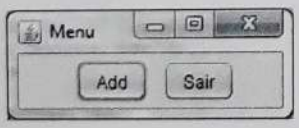
a) Uma janela (JFrame) personalizada. Para sua aparência utilize:

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

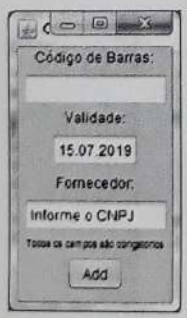
Utilize o bloco `try - catch` na personalização da aparência nimbus. Em caso de exceção, trate com a mensagem abaixo e encerre o sistema:



b) Uma aplicação que ao ser executada inicia a TelaMenu:



A opções Add do Menu permite a abertura da tela de Cadastro:



- A Tela cadastro exibe os campos para cadastro de um Produto.
- O fechamento da tela Cadastro é realizada pela opção fechar do JFrame, mas que NÃO encerram o sistema. Apenas a tela Menu encerra a aplicação.

c) Utilize a codificação abaixo para o campo de validade do produto:

```
JFormattedTextField validadeField = new JFormattedTextField(new
     SimpleDateFormat("dd.MM. yyyy"));
       validadeField.setValue(new java.util.Date());
```

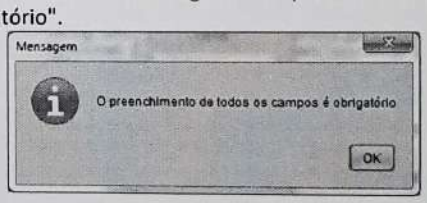
d) O botão Adicionar é responsável por cadastrar um novo produto. Ao cadastrar um novo produto, as informações ser exibidas em JOptionPane. Já o botão Sair é responsável por encerrar o sistema, bem como a opção fechar do JFrame da TelaMenu.

Pontuação: Tratamento de exceção: 0,5 ponto

e) Implemente o método validarDados( ) no controlador de maneira que todos os campos da TelaCadastro sejam obrigatórios:

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

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 através de JOptionPane a mensagem: "O preenchimento de todos os campos é obrigatório".



Esse método será utilizado pelo botão Add da TelaCadastro.

Composição: 1,0 ponto

f) Como se percebe na Tela Cadastro, os campos são relacionados a um Produto. Faça a devida abstração para criação de seus atributos, tipos e construtor. Todos os atributos deverão ser privados.

g) Fornecedor possui uma relação de dependência forte com produto, em que produto é o todo e fornecedor é a parte, conforme ilustrado no diagrama de classes. A criação dos objetos deve refletir essa criação.

h) Faça a devida utilização de polimorfismo para a sobrecarga de construtores de maneira a garantir a composição.

Composição: 1,0 ponto

i) Realize uma relação 1..N entre fornecedor e endereço, de maneira que um fornecedor possa ter diversos endereços.

Tratamento de exceção: 1,5 ponto

j) Os produtos do sistema estão armazenados no atributo estático *produtos* da classe BaseDados. Faça o devido uso de ArrayList. Nesse sistema cada produto é diferenciado pelo código de barras (único entre os produtos).
O ArrayList deverá ser manipulado pelos métodos CRUD definidos no diagrama de classes:

- Se um produto já cadastrado tentar ser inserido novamente deve-se exibir uma caixa de diálogo com mensagem "Produto já cadastrado!", caso contrário "Produto cadastrado com sucesso!". A verificação deverá ser realizada no método `public static boolean addProduto(Produto produto){}` responsável por adicionar um produto no ArrayList.
- Trate o erro de cadastro de produto por Exceção:
  `public static boolean addProduto(Produto produto) throws ProdutoException{}`
- A verificação da existência de um produto é dada por
  `boolean existeProduto(Produto produto) {}`.
- Observe a busca por produtos vencidos, ou seja, sua validade expirou (vide letra k))
- A consulta dos dados de um produto deve utilizar o método toString().

Interface: 1,0 ponto

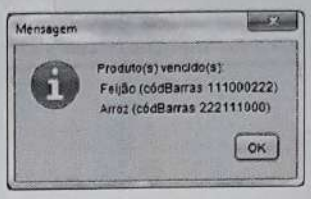
k) A classe BaseDados realiza a interface Verificador. Segue abaixo a definição do método.

```
public static boolean verificarValidade(Produto
produto){
      Date dateAtual=new
Date(System.currentTimeMillis());
      Calendar calB = Calendar.getInstance();
      calB.setTime(dateAtual);
      dateAtual = calB.getTime();
      if(produto.getValidade().before(dateAtual))
         return true;
      return false;
}
```

Thread: 1,5 pontos

l) App é uma especialização de Thread responsável exibir a cada 10 segundos os "produtos" cadastrados em uma caixa de diálogo.

m) A Thread também é responsável por verificar se um produto passou do prazo de validade. Se um produto estiver fora da validade, o sistema deverá emitir a mensagem:



Objetos e Chamadas de Métodos: 0,5 ponto

n) Deve-se ter em App:

- A criação de instâncias MVC
- Execução da thread
- Deverá ter pelo menos dois produtos já cadastrados:
  - 100 unidades de feijão, validade de 10/07/2019 e código de barras 111000222;
  - 50 unidades de arroz, validade de 14/07/2019 e código de barras 222111000.
  - 1000 unidades de ovos, validade de 20/04/2023 e código de barras 333222111.

BOA PROVA!

DIAGRAMA DE CLASSES