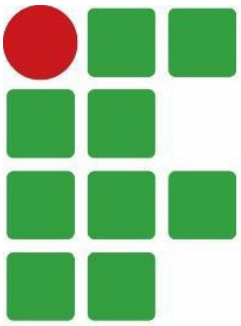


INSTITUTO FEDERAL
Minas Gerais
Campus Ouro Branco

Professor: Saulo Henrique Cabral da Silva
Alunos: Arthur Lana e Guilherme Monteiro
Curso: Técnico Integrado em Informática – 3º ano
Disciplina: Tópicos em Desenvolvimento de Software
Github:https://github.com/GuilhermeMonteiroKVT/RevisaoMercearia_ArthurLana_GuilhermeMonteiro
https://github.com/arthurlana/TemDS-TrabalhoPropostoRevisao_Arthur_Guilherme

Obs.: Para garantir o aprendizado de ambos os estudantes que compõem a dupla, cada um desenvolveu um algoritmo em sua máquina. Porém, a documentação foi feita em conjunto.

Introdução

O programa desenvolvido consiste em um sistema de gerenciamento de vendas de uma mercearia e foi desenvolvido por meio da utilização da plataforma “NetBeans IDE 8.2”. O Trabalho Prático 2 tem como objetivo possibilitar a revisão de alguns conceitos em um primeiro contato com a nova metodologia remota de ensino.

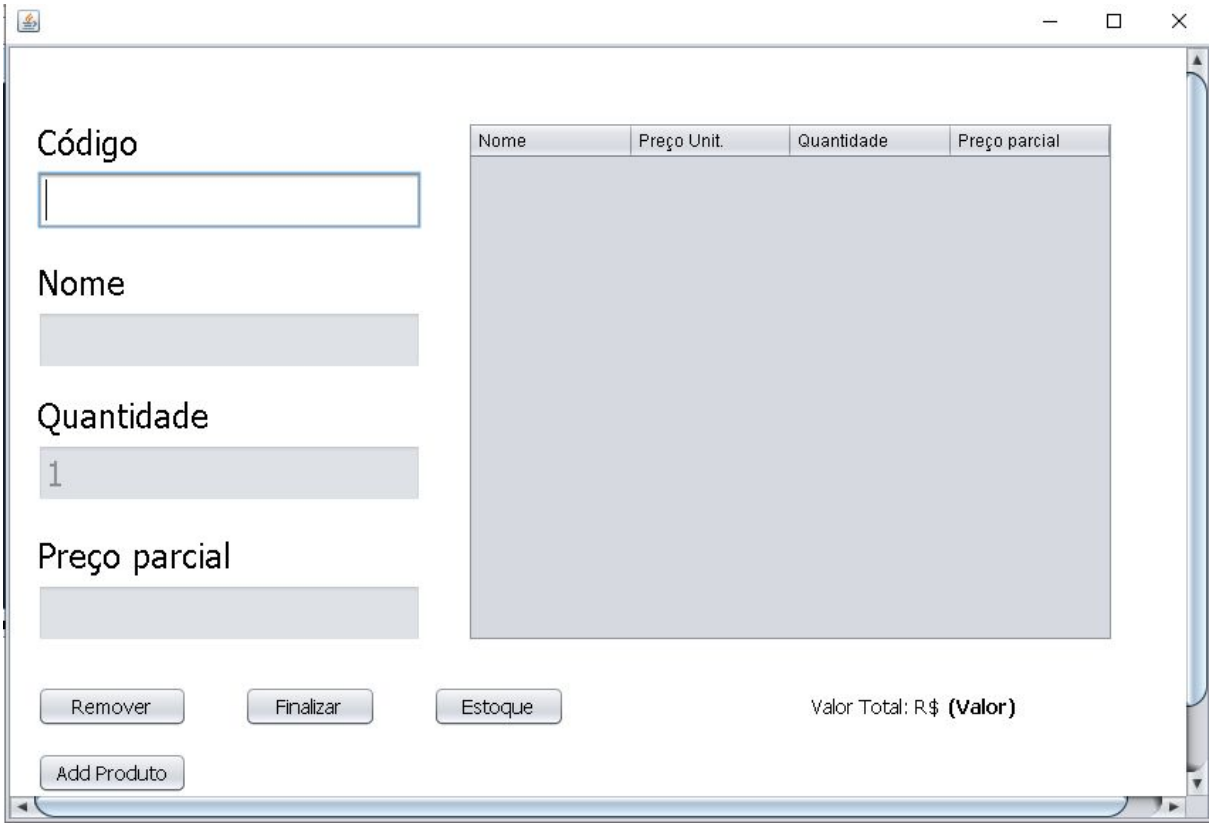
Implementação

Para realizar a implementação, utilizamos um JFrame, dois JPanels e três classes Java.

- O JFrame “JanelaPrincipal” consiste em frame para exibir os dois painéis desenvolvidos ao longo do desenvolvimento do programa. Essa exibição funciona por meio da utilização de CardLayout’s e barras de rolagem (JScrollPane), o que facilita a construção das interfaces e permite uma execução mais otimizada.

  Métodos: efetuaTransicao(): Efetua a transição dos painéis.
  formWindowClosing(java.awt.event.WindowEvent evt): Método com a função de acionar o método atualizaArquivo da classe “FakeBancoDados” para atualizar o arquivo csv quando o frame for fechado.

- O primeiro JPanel, “CompraGUI”, resume-se em uma interface que permite a visualização dos elementos do produto, a adição e remoção de produtos em uma tabela que representa um “carrinho de compras”, a finalização da compra e da venda e o acesso ao segundo painel.



Métodos:removerbutonActionPerformed(java.awt.event.ActionEvent evt): Método que configura a remoção de itens selecionados na tabela pelo uso de um botão.
codtxtKeyTyped(java.awt.event.KeyEvent evt): Define que itens no campo de texto do código, só serão lidos depois que a tecla enter(“\n”) for pressionada, e avalia se o código informado é um número, se ele existe e posteriormente busca o produto baseado no código informado e preenche os outros campos do produto com suas demais informações, como seu preço.
quantidadetxtKeyReleased(java.awt.event.KeyEvent evt): Responsável por ler o campo de texto da quantidade informada pelo operador do caixa e calcula o preço parcial do produto.
addprodutobutonActionPerformed(java.awt.event.ActionEvent evt): Adiciona um novo produto a tabela e limpa os campos de texto para que possa ser adicionado outros produtos e avisa caso no estoque não possua a quantidade desejada pelo cliente.
finalizarbutonActionPerformed(java.awt.event.ActionEvent evt): Finaliza a compra limpando o carrinho e verifica se o estoque possui as quantidades do produtos no carrinho e zera o valor total da compra.

- “EstoqueGUI” configura-se como o segundo painel desse sistema de gerenciamento da mercearia, porém ainda não compõe uma exigência do Trabalho Prático 2 e, portanto, não foi implementado.

- A primeira classe, “Produto”, permite o instanciamento de objetos que armazenam os dados dos produtos da mercearia: código, nome, preço unitário e a quantidade existente no estoque.
  Métodos: Getters e Setters (manipulação de dados).
  toString (importante para modificação da planilha do estoque).

- A segunda classe, “FakeBancoDeDados”, possibilita o acesso, a leitura e o armazenamento dos dados do arquivo.csv e, posteriormente, a sua modificação mediante a finalização de vendas, que reduzem a quantidade de produtos existentes no estoque.
  Métodos: cargaArquivo(): acesso, leitura e armazenamento da planilha.

```java
public class FakeBancoDeDados {
    private static Vector<Produto> produtos;
    //leitura das informações da planilha do excel
    private static void cargaArquivo(){
        //ajuste na criação do vetor de produtos estático
        if(produtos == null){
            produtos = new Vector<>();
        }else{
            produtos.clear();
        }
        File arquivoCsv = new File("C:\\Users\\Yasminn\\Documents"
                + "\\1 - Main\\EAD IFMG\\Tópicos em Desenvolvimento de Software\\arquivos.csv");
        try{
            //estruturas de leitura do arquivo
            FileReader marcaLeitura = new FileReader (arquivoCsv);
            BufferedReader buffLeitura = new BufferedReader(marcaLeitura);
            //******ler cada linha******
            //lendo a primeira linha descartando o cabeçalho
            buffLeitura.readLine();
            String linha = buffLeitura.readLine();
            while(linha != null){
                //lendo as linhas seguintes até o final do arquivo
                String infos[] = linha.split(";");
                int cod = Integer.parseInt(infos[0]);
                String nome = infos[1];
                double preco = Double.parseDouble(infos[2]);
                int qnt = Integer.parseInt(infos[3]);
                //adição dos produtos no vetor dinâmico
                produtos.add(new Produto(cod, nome, preco, qnt));
```

```
                produtos.add(new Produto(cod, nome, preco, qnt));
                //lendo a próxima linha
                linha = buffLeitura.readLine();
            }
            //Liberando o arquivo para outros processos.
            buffLeitura.close();
        }catch(FileNotFoundException ex){
            System.err.println("Arquivo especificado não existe.");
        }catch(IOException e){
            System.err.println("Arquivo corrompido");
        }
    }
    public static Produto consultaProdutoCod(int cod){...14 linhas }
    public static void atualizaArquivo(){...14 linhas }
}
```

consultaProdutoCod(int cod): consulta um produto a partir do seu código.

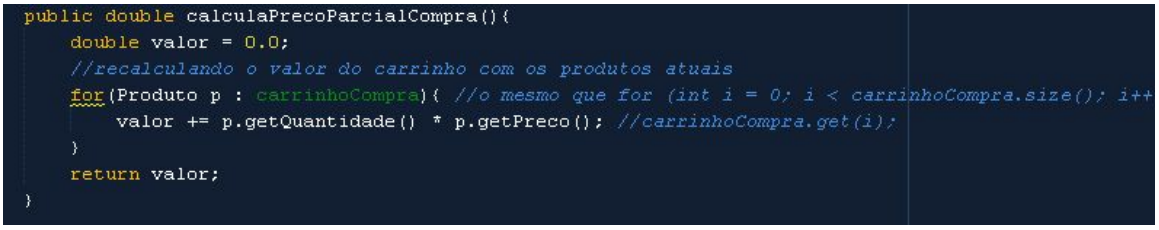
```
public static Produto consultaProdutoCod(int cod){
    //o arquivo é carregado se ainda não tiver carregado anteriormente
    if(produtos == null){
        cargaArquivo();
    }

    for(Produto prodI:produtos){ //o mesmo que for(int i = 0; i < produtos.size(); i++);
        if(prodI.getCodigo() == cod){  //if(produtos.get(i).getCodigo() == cod)
            return prodI;
        }
    }
    //não existe produto com o código especificado
    return null;
}
```

atualizaArquivo(): atualiza a planilha do estoque (arquivo.csv).

```
public static void atualizaArquivo(){
    File arquivo = new File("C:\\Users\\Yasminn\\Documents\\"
            + "1 - Main\\EAD IFMG\\Tópicos em Desenvolvimento de Software\\arquivos.csv");
    try {
        FileWriter escritor = new FileWriter(arquivo);
        BufferedWriter buffEscrita = new BufferedWriter(escritor);
        for(int i = 0; i < produtos.size();i++){
            buffEscrita.write(produtos.get(i)+"\n"); //a chamada do toString é desnecessária, mas pode ser feita.
        }
        buffEscrita.flush();
        buffEscrita.close();
    } catch (IOException ex) {
        System.err.println("Dispositivo com falha");
    }
}
```

- A terceira classe, “ModeloTabelaCompra” é responsável por formatar a tabela, definir seu cabeçalho e número de linhas e colunas, permitir a manipulação do carrinho (tabela do painel “CompraGUI”), adicionar, remover e modificar as informações de itens na tabela, possibilitando a soma dos valores parciais de cada lote de produtos e a exibição desse valor total na interface por meio de um jLabel.
  Métodos: addNovoProduto(Produto vendido): adiciona um produto no Carrinho.
  removeProdutoCarrinho(int indice): remove um produto do Carrinho

calculaPrecoParcialCompra(): recalcula o valor do carrinho com os atuais produtos presentes nele.



```
public double calculaPrecoParcialCompra(){
    double valor = 0.0;
    //recalculando o valor do carrinho com os produtos atuais
    for(Produto p : carrinhoCompra){ //o mesmo que for (int i = 0; i < carrinhoCompra.size(); i++
        valor += p.getQuantidade() * p.getPreco(); //carrinhoCompra.get(i);
    }
    return valor;
}
```

limpaCarrinho(): Responsável pela limpeza dos produtos no carrinho.
setValueAt(): Requisita senha para alteração de dados na tabela e a atualiza.

Conclusão

O desenvolvimento do algoritmo e da documentação permitiram, principalmente, a revisão de alguns conceitos, exercitação de manipulação de tabelas e eventos e o aprendizado de maneiras para acessar e modificar arquivos armazenados no computador, que nunca havíamos aprendido em anos anteriores do Curso Integrado de Informática. Com relação aos problemas encontrados, todos foram resolvidos ao longo das aulas, já que a proposta do Trabalho Prático 2 era implementar o código em sua própria máquina juntamente com o professor, o que já reduzira a possibilidade de erros, mas decerto é visível o aumento da dificuldade entre das últimas implementações.

Referências Bibliográficas

Aulas assíncronas publicadas na plataforma AVA “Moodle”.