Introdução

O trabalho desenvolvido foi um programa criado com o intuito de auxiliar uma pessoa fictícia com um problema bem comum: a necessidade de um sistema para controle das vendas e do estoque de um pequeno mercado. Essa pessoa (Seu José) costumava utilizar uma planilha do Excel para controlar a entrada e saída dos produtos, porém conforme a tabela foi enchendo, ele percebeu que consultar essa tabela estava demorando, e que a situação ia se tornar um caos se continuasse como estava. O programa consiste em uma seleção para achar a planilha no Excel para usá-la tanto para armazenar dados, quanto para controlar os dados já existentes sobre os produtos. Após a planilha ser selecionada, o usuário terá uma tela na qual ele pode pesquisar os produtos pelo código de barras (esse código teoricamente seria recebido através de leitor(es) que ficariam no(s) caixa(s), e passariam os códigos diretamente para o sistema, depois o usuário (caixa), pode selecionar a quantidade do produto que teve seu código buscado para acrescentar o valor daquela determinada quantidade de produtos para o carrinho de compra. O carrinho consiste em uma tabela na qual é possível ver os produtos, a quantidade deles que está sendo inclusa (esse valor é modificável) e o preço de cada item, bem como o preço total por todos os itens que estão sendo comprados. Além disso há um botão de finalizar compra que zera o carrinho e assume que a compra foi realizada.

Implementação

O programa funciona com base em uma tela principal, 4 campos de texto (que representam respectivamente o código, o nome, a quantidade e o preço parcial do produto sendo pesquisado no campo de código).

O programa possui no momento 6 classes, sendo 5 delas já funcionais, e elas são:

- JanelaPrincipal.java - Jframe principal do programa, onde ficam todos os outros componentes Swing existentes;
- CompraGUI.java - painel principal onde é realizado o gerenciamento do produto até a compra, é a tela que aparece no programa quando ele é iniciado;
- FakeBD.java - classe que faz a leitura (e eventual alteração) da planilha no Excel, bem como faz a pesquisa dos produtos na planilha usando seu código como parâmetro de pesquisa.
- ModeloTabelaBuy.java - classe que modela tanto a organização quanto a formatação dos dados para a tabela que aparece no painel CompraGUI. Essa tabela é utilizada como carrinho de compras e é uma das partes mais importantes do programa, logo se essa classe faz sua modelagem, é de extrema importância pois a tabela depende dessa classe para funciona.
- Produto.java - Modelo dos objetos do tipo Produto, é a classe que permite ao programa funcionar.

As variáveis principais de cada classe são:

-Janela Principal:

- JScrollPane scrollB: barra de rolagem das telas;
- JPanel trocaInfo: painel que é auxilia na troca de informações
- CardLayout panels: CardLayout do programa, permite a troca entre telas.

-CompraGUI: (não vou listar os botões nem campos de texto pois vou explorar a funcionalidade deles depois)

- Produto produtoS: objeto que recebe os valores do produto sendo pesquisado para compra;
- Produto prodSold: objeto que recebe os valores do produto que foi vendido;
- ModeloTabelaBuy cart: objeto resultante do modelo da tabela;
- Double precoComp: preço da compra que é usado no programa para definir o valor total da compra em cada compra.
- Int qUser : quantidade do produto que está sendo pesquisado que o usuário deseja comprar.

-FakeBD:

- Vector<Produto> produtos: vetor que recebe todos os objetos do tipo de produto que serão criados pelo programa para receber os dados da tabela no Excel.

-ModeloTabelaBuy:

- Vector<Produto> buyCart: vetor de Objetos do tipo Produto que recebe apenas os produtos que estão sendo comprados;
- CompraGUI panel: é instanciado aqui com o intuito de usar um de seus métodos nessa classe.

-Produto:

- Int codigo: referente ao código dos produtos;
- String nome: referente ao nome dos produtos;
- Double preco: referente ao preço dos produtos;
- Int quantidade: referente a quantidade de cada produto.

Os métodos principais de cada classe são:

-Janela Principal:

- Private void MyInitComponents: método que inicializa os componentes da tela de forma personalizada, criando e permitindo o funcionamento de elementos como as barras de rolagem lateral, e instanciando o modelo CardBoardLayout que será usado quando houver a possibilidade de alternar entre as telas de compra e de estoque;

- Trasition static void: método que faz as trocas entre os painéis do programa. como no momento a tela de estoque não está sendo usada, este método ainda não está sendo usado também;
- Private void formWindowClosing: método que atualiza os dados na planilha do Excel quando o programa é encerrado.

-CompraGUI:

- MyInitComponents: instancia a tabela principal do programa (tableCar) para seguir o modelo da classe ModeloTabelaBuy;
- CodigoTxtKeyTyped: após um código ser digitado e a tecla enter ser digitada, esse método checa se o código digitado é numérico (dispara um erro caso não seja ou caso o código seja nulo), e se o valor for numérico ele pesquisa o código na tabela, (se não achar nenhum produto com esse código, o programa envia uma mensagem apontando que não há produto com esse código) e caso ele ache algum produto com o mesmo código, ele exibe seus outros dados nas caixas de texto inferiores na tela(esses dados são carregados do objeto produtoS, que é instanciado durante o funcionamento deste método);
- QuantTxtKeyboardReleased: após o último método citado ser utilizado e os dados de um produto serem exibidos nas caixas de texto, esse método permite a alteração da quantidade do produto em questão, para que seja acrescentado mais de um ao carrinho. Isso também altera seu preço parcial, já que o preço unitário do produto será multiplicado pela nova quantidade digitada. Dispara uma mensagem de erro caso o usuário tente colocar um caractér não numérico. Esse método usa as variáveis do objeto produtoS também;
- AddProdButtonActionPerformed: método referente ao botão de adicionar produtos sendo pesquisados ao carrinho. Primeiro este método checa se o produto cujos dados estão sendo exibidos é válido, caso seja ele instancia um objeto chamado produtoSold para receber os dados do produto que vai ser adicionado ao carrinho, incluindo a quantidade que pode ter sido ou não alterada pelo usuário. Depois, caso a adição do produto dê certo, este método também reseta os campos para a pesquisa de um novo produto.(Caso não haja uma quantidade suficiente do produto que o usuário está tentando adicionar ao carrinho, o programa dispara uma mensagem de erro e avisa que a quantidade é insuficiente, e reseta novamente os campos para que o usuário pesquise novamente. Caso o valor da quantidade não seja um valor numérico, também há uma mensagem de erro neste método);
- F5: método que atualiza a tabela principal na tela, alterando seus valores para corresponder à quaisquer alterações feitas antes da execução deste método (este método é chamado dentro de outros métodos para que mudanças geradas por mudar a quantidade de um produto ou removê-lo sejam visíveis ao usuário imediatamente);
- RemoveButtonActionPerformed: Remove um produto selecionado do carrinho (que corresponde a uma linha na tabela), utilizando o método removeProd que recebe o dado de qual linha está selecionada, removendo-a, assim

mudando o valor total da compra, subtraindo-o do valor do produto que foi removido. (esse método tem uma caixa de verificcação para o usuário ter certeza de que quer remover aquele produto, e exige a senha do gerente (12345) para que a remoção aconteça, e caso não haja nenhum produto selecionado, o método dispara um erro pedindo que o usuário selecione algum);

- VerifyQuant: método que verifica se todos os produtos existem em quantidade suficiente para realizar a venda que está sendo solicitada. (este método compara as quantidades de todos os produtos no carrinho com as quantidades dos objetos Produto dentro do vetor que recebe todos os produtos, e gera um valor booleano falso para caso qualquer um dos produtos esteja numa quantidade insuficiente além de disparar uma mensagem de erro, e verdadeiro apenas para caso todos estejam nas quantidade necessárias,);

-FakeBD:

- LoadArquivo: método que cria um arquivo do tipo File para receber o local(path) da planilha do excel no disco rígido, e depois lê seu conteúdo linha por linha (utilizando de um bufferedReader), armazenando-o num vetor que conterá todos os produtos da planilha. Caso o path informado esteja errado, o programa dispara uma mensagem de erro indicando que o arquivo especificado não existe ou foi corrompido;
- SearchCode: método que roda o vetor produtos, comparado um código informado pelo usuário (deve ser um valor numérico) com os códigos de todos os objetos do tipo Produto armazenados no vetor. Caso não encontre nenhum produto com aquele código, ele dispara uma mensagem de erro avisando que não há nenhum produto com o código pesquisado. Caso funcione, como este método está sendo chamado dentro de outro, ele permite que os dados do produto sejam armazenados num objeto auxiliar (produtoS) para serem exibidos na tela;
- UpdateFile: método que manda as mudanças nas quantidades dos produtos no programa de volta para a planilha do excel, alterando as linhas (através de um BufferedReader) no índice da linha certo onde se encontram os produtos que foram “comprados”;

-ModeloTabelaBuy:

- GetRowCount: retorna o número de produtos que há no carrinho (número de linhas que há na tabela);
- GetValueAt: recebe os índices de linha e coluna na tabela, e retorna o dado presente naquele local especificado na tabela (possui um tipo de retorno para cada possível índice de coluna, já que cada coluna armazena um tipo de dado diferente;
- AddProd: adiciona um produto ao carrinho (é o método principal utilizado pelo botão ‘adicionar produto’);

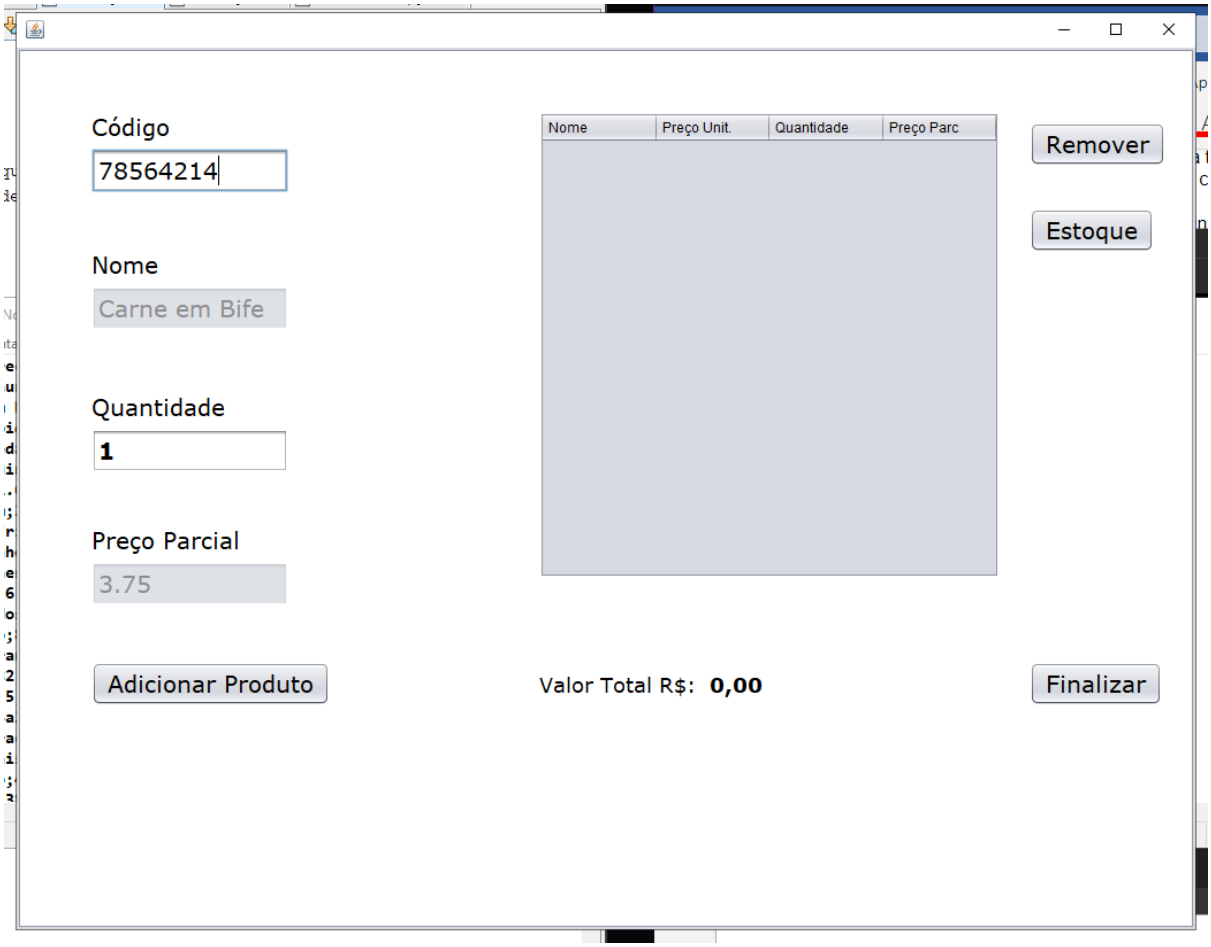
- RemoveProd: recebe o índice de uma linha na tabela e remove o produto correspondente àquele índice no carrinho. (é o método principal utilizado pelo botão ‘remover’);
- IsCellEditable: método que habilita a edição apenas da quantidade na tabela(carrinho);
- SetValueAt: método que recebe o valor que será incluso na tabela, e os índices de linha e coluna, e altera o valor que estava anteriormente no local indicado pelos índices para o novo valor fornecido. Exige a senha do gerente quando para alterar a quantidade de qualquer produto já presente no carrinho;
- CalculaPrecoParcial: método que faz propriamente a conta que gera o valor parcial de qualquer produto, essa conta é: valor do produto * quantidade especificada pelo usuário, retornando o resultado da conta como double;
- ResetCart: método que limpa a tabela (ou reseta o carrinho) para uma nova compra, chamado pelo botão finalizar;

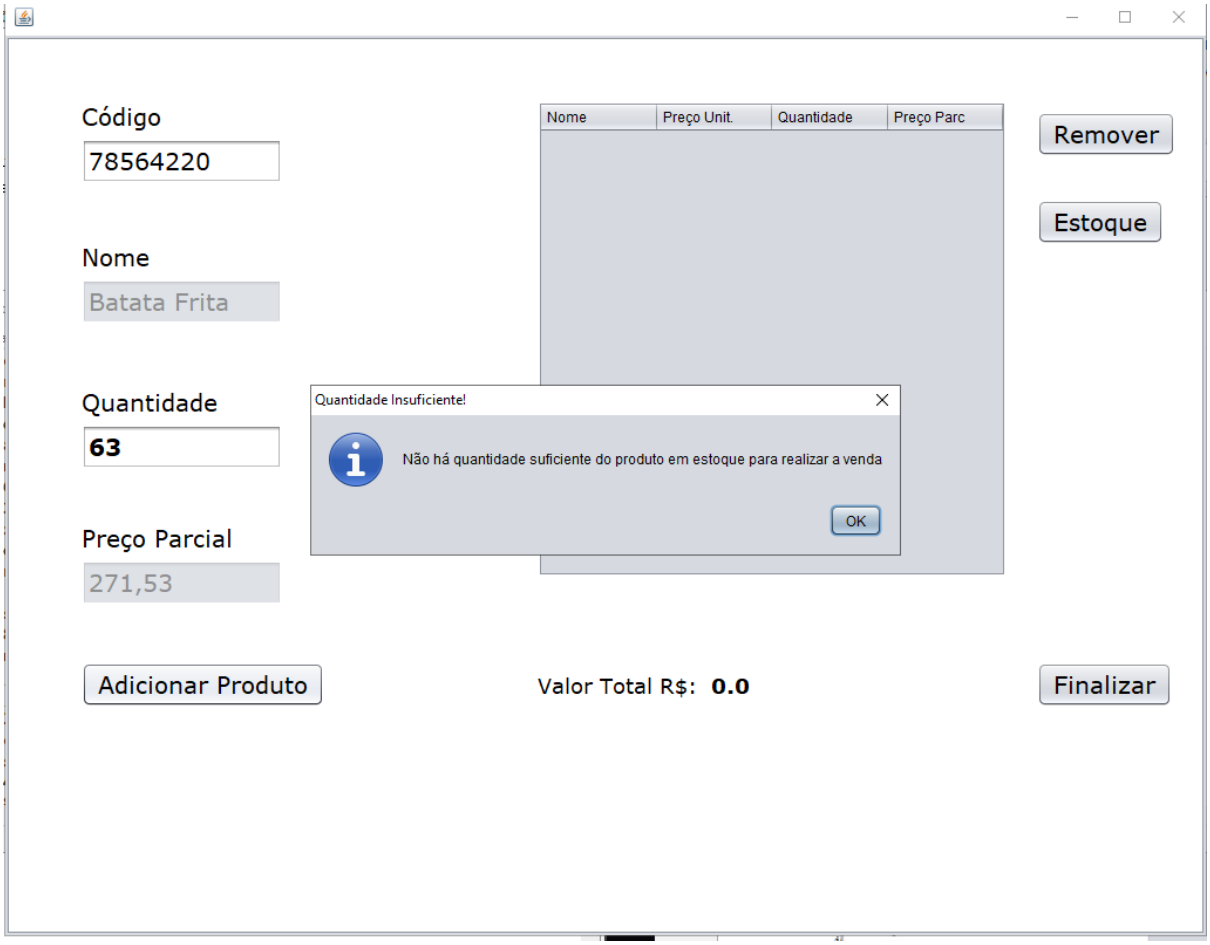
-Produto:

- ToString: método sobrescrito que agora gera uma string com os dados de cada produto já no formato que será incluído na planilha. (É chamado dentro do método UpdateFile, para formatar imediatamente os dados de cada produto para serem atualizados na planilha)

Prints do programa funcionando


Código
Nome
Quantidade
1
Preço Parcial
Adicionar Produto
Nome
Preço Unit.
Quantidade
Preço Parc
Remover
Estoque
Valor Total R$: 0.0
Finalizar


Código
78564214
Nome
Carne em Bife
Quantidade
1
Preço Parcial
3.75
Adicionar Produto
Nome
Preço Unit.
Quantidade
Preço Parc
Remover
Estoque
Valor Total R$: 0,00
Finalizar



Conclusão

Através desse trabalho consegui recuperar boa parte das práticas de programação, e sinto que retomei um pouco do ritmo necessário para continuar bem durante o ano na matéria de Tópicos em Desenvolvimento de Software. Minhas maiores dificuldades foram na fase de testes e na produção desta documentação.

Bibliografia

Videoaulas disponibilizadas pelo professor Saulo.

Material disponibilizado pelo professor Saulo.