UNIVERSIDADE FEDERAL DE UBERLÂNDIA

CIÊNCIA DA COMPUTAÇÃO

Princípios e Padrões de Projetos – Prof. Marcelo Maia

Aluno: Daniel Borges Gonçalves Matrícula: 12311BCC005

API JDBC

Sobre o trabalho

O presente trabalho tem como objetivo o desenvolvimento de uma aplicação Java para o controle de estoque de produtos, utilizando a API JDBC e o banco de dados Apache Derby em modo embarcado. A aplicação permitirá a realização de operações essenciais de gerenciamento de produtos, como inserção, consulta, atualização e remoção de registros, implementando um sistema CRUD (Create, Read, Update, Delete).

A funcionalidade de atualização será aplicada na forma de operações de compra e venda, permitindo a modificação dinâmica da quantidade de cada produto em estoque. Para tornar o sistema acessível e intuitivo, a interface gráfica foi desenvolvida utilizando a biblioteca Swing, permitindo que o usuário interaja com as funcionalidades de forma visual e prática.

O banco de dados será responsável por armazenar as informações dos produtos, que incluirão atributos como ID único, nome, descrição, preço e quantidade disponível. A comunicação entre a aplicação e o banco será realizada através de comandos SQL executados via JDBC, garantindo a integridade e a persistência dos dados.

Para a execução do projeto, é necessário rodá-lo em um ambiente de desenvolvimento compatível, como o IntelliJ IDEA, garantindo que a configuração do banco de dados e da conexão JDBC esteja corretamente ajustada.

Com isso, este projeto visa não apenas a implementação de um sistema funcional para controle de estoque, mas também o aprofundamento no uso do JDBC, na estruturação de aplicações Java com interface gráfica em Swing e na integração com banco de dados embarcados.

O código completo do programa pode ser encontrado no github:

Danielbgoncalves/java-derby-db-application-: Essa aplicação Java usa o Apache Derby como banco de dados embarcado de uma aplicação de compras e vendas de produtos

Estrutura do projeto

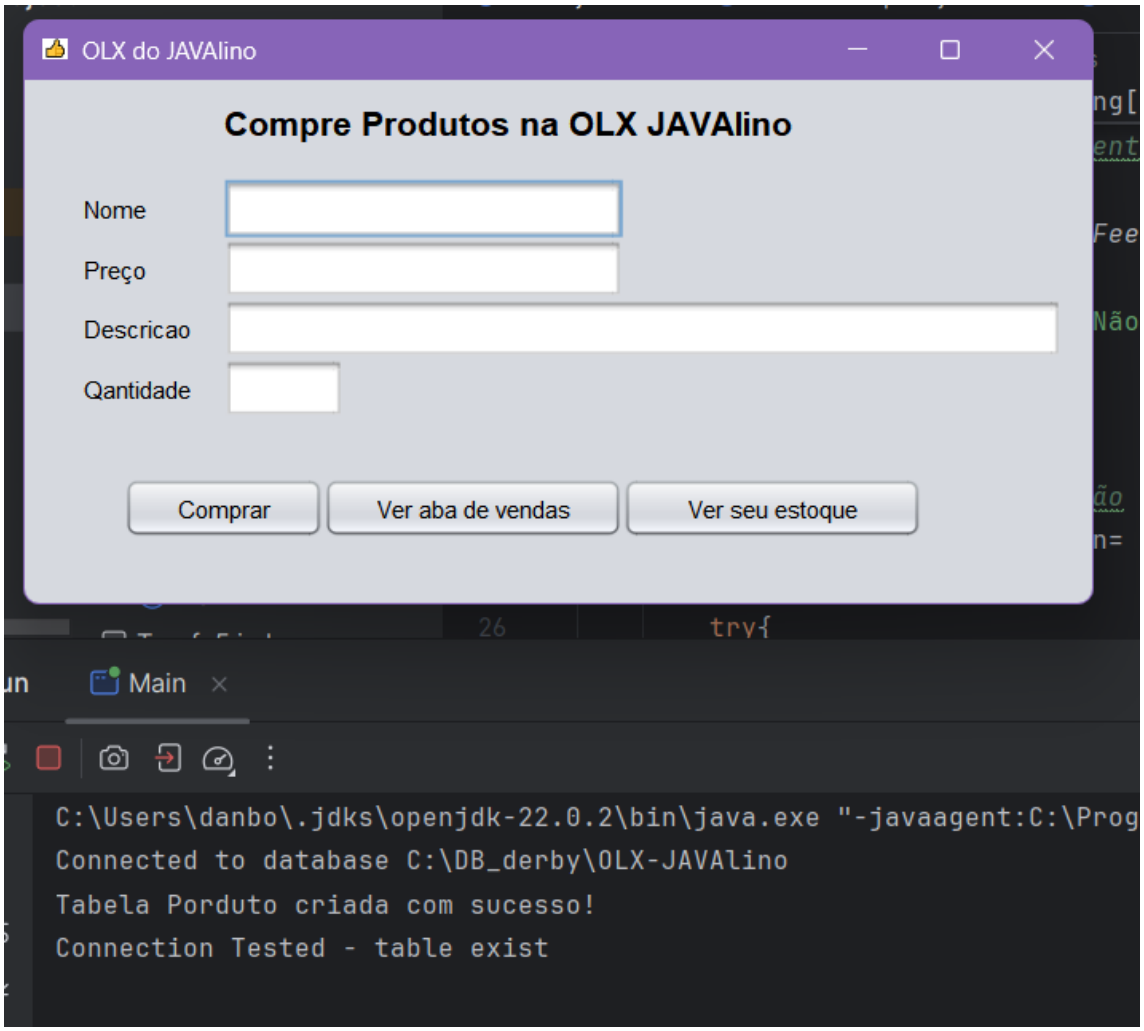
O projeto está organizado de forma modular, separando as responsabilidades em diferentes pacotes:

- **dao (Data Access Object):** Responsável pela interação com o banco de dados.
    - ProdutoDAO: Classe que gerencia as operações CRUD para os produtos.
- **db (Database):** Contém a classe de conexão com o banco de dados.
    - DBConnection: Classe que gerencia a conexão JDBC com o banco Derby.
- **model (Modelagem de Dados):** Define as entidades do sistema.
    - Produto: Representa um produto no estoque, contendo atributos como ID, nome, descrição, preço e quantidade.
- **ui (User Interface):** Contém as classes responsáveis pela interface gráfica.
    - PainelComprar: Tela para registrar compras e aumentar o estoque.
    - PainelVender: Tela para registrar vendas e reduzir o estoque.
    - PainelEstoque: Tela para visualizar os produtos cadastrados.
- **Outros arquivos:**
    - Main: Classe principal que inicia a aplicação.
    - dialogMessage: Classe utilitária para exibir mensagens na interface, os pop ups de aviso de conclusão ou de erros.

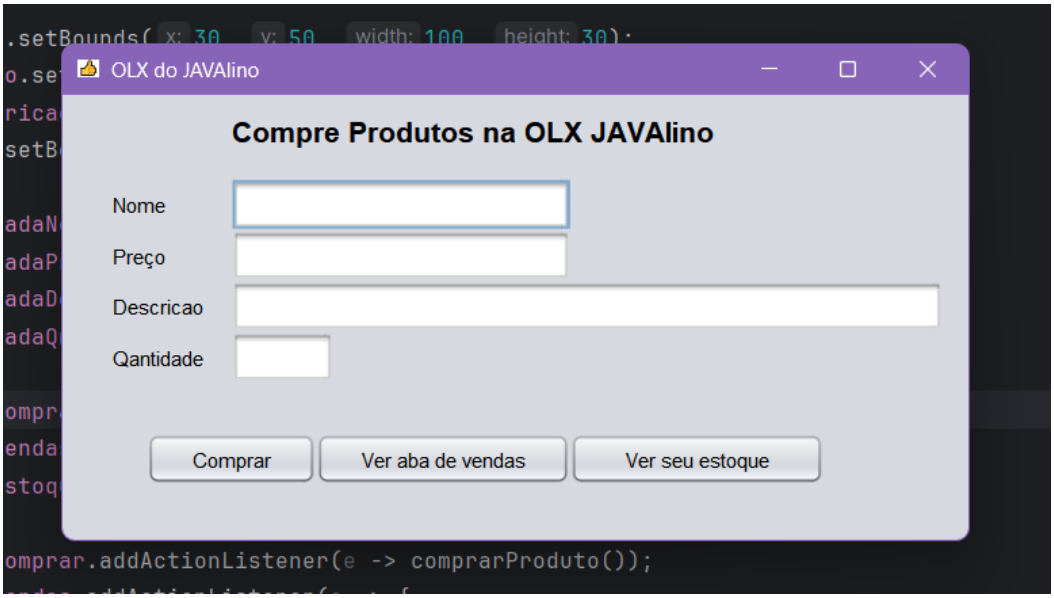
Telas de execução:

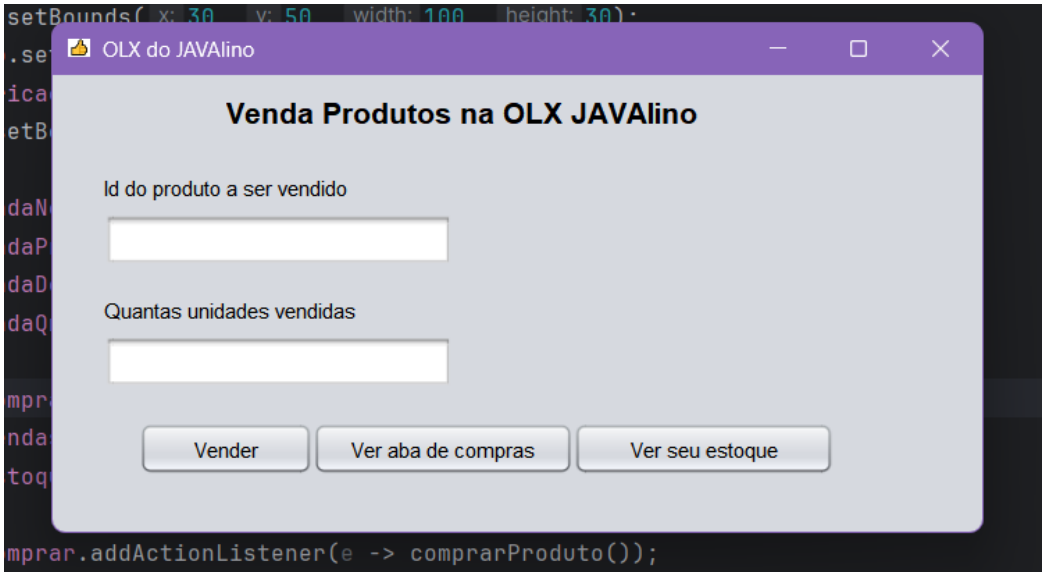
A sequência das telas segue desde a criação do banco de dados até sua aplicação. Primeiro há a conexão com o Apache Derby, criação da tabela Produtos e em seguida ela pode ser usada na aplicação.

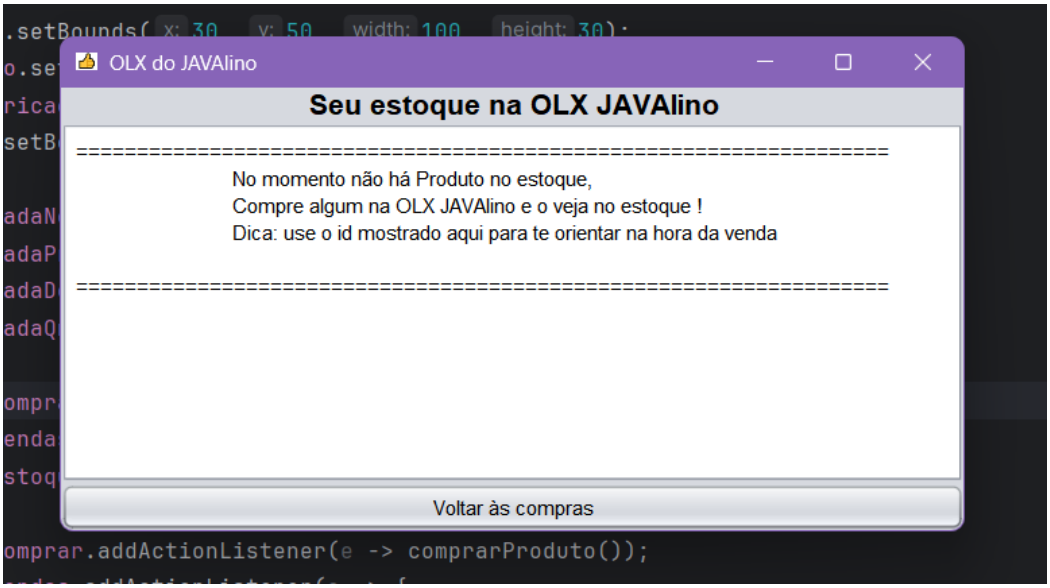
- Quando a tabela não existe no caminho especificado ela é criada. Após ser criada a conexão é testada: tentamos fazer um update na tabela em uma tupla inexistente, não há de fato atualização alguma, contudo isso serve para testar se a tabela está respondendo aos comandos. Como é possível verificar no print a tabela foi criada e testada com sucesso.



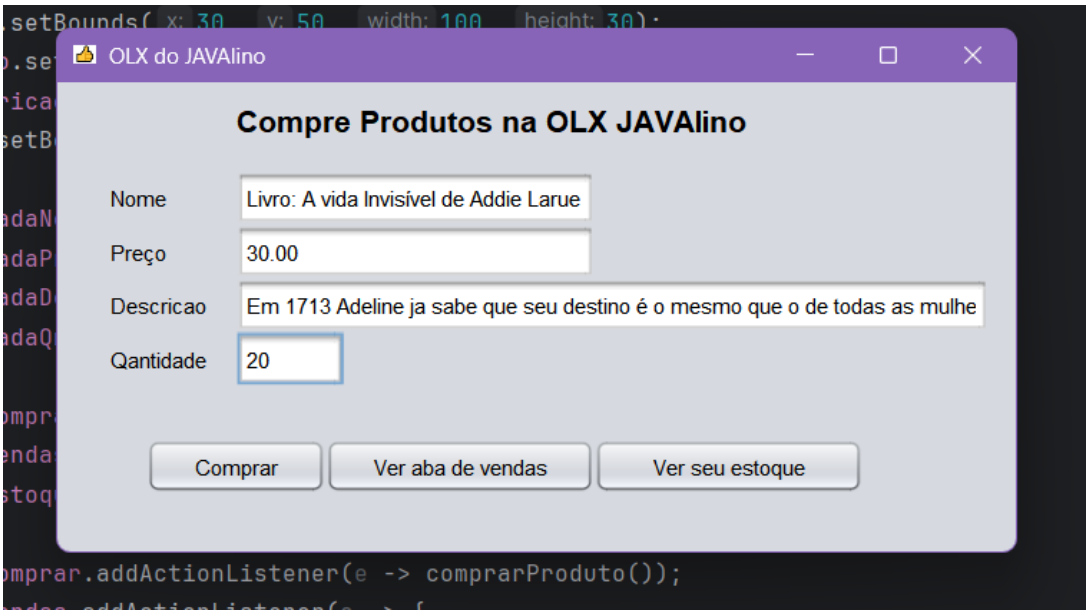
- Como os painéis são:

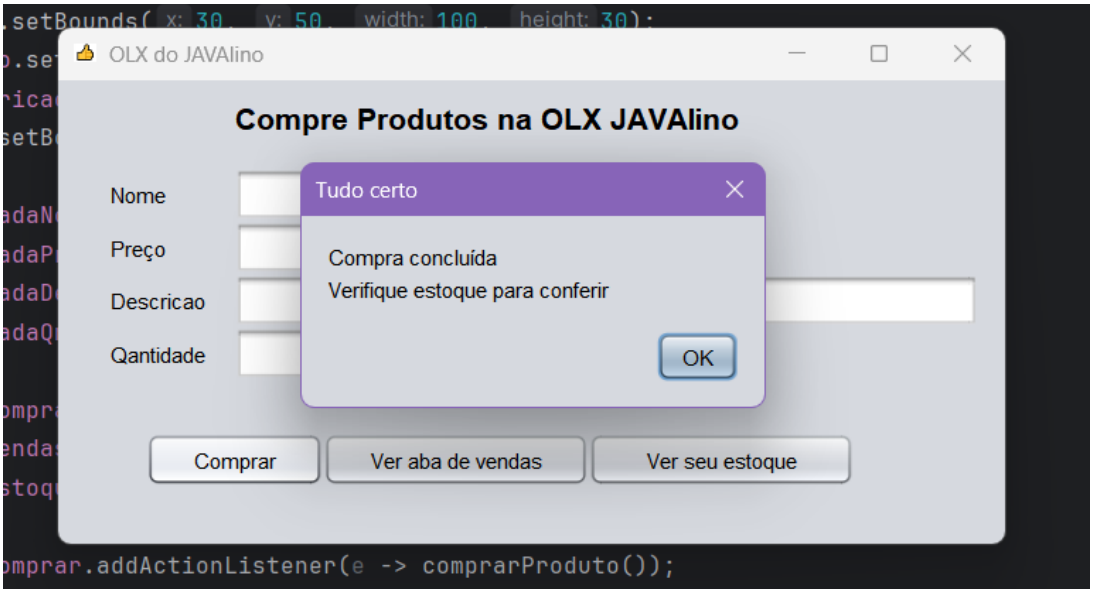




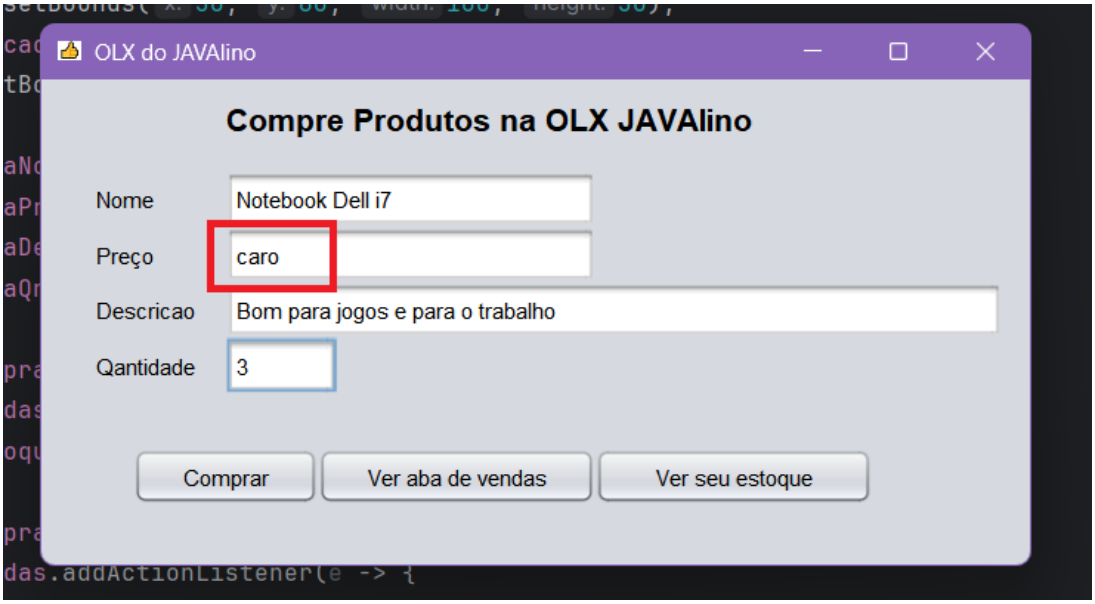


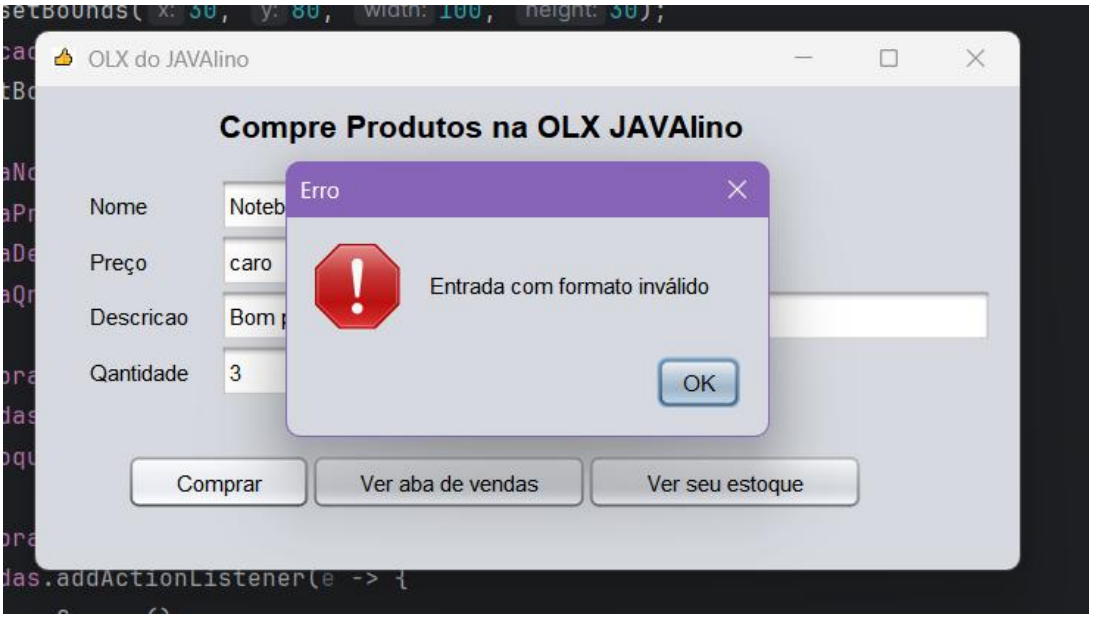
- Primeira compra





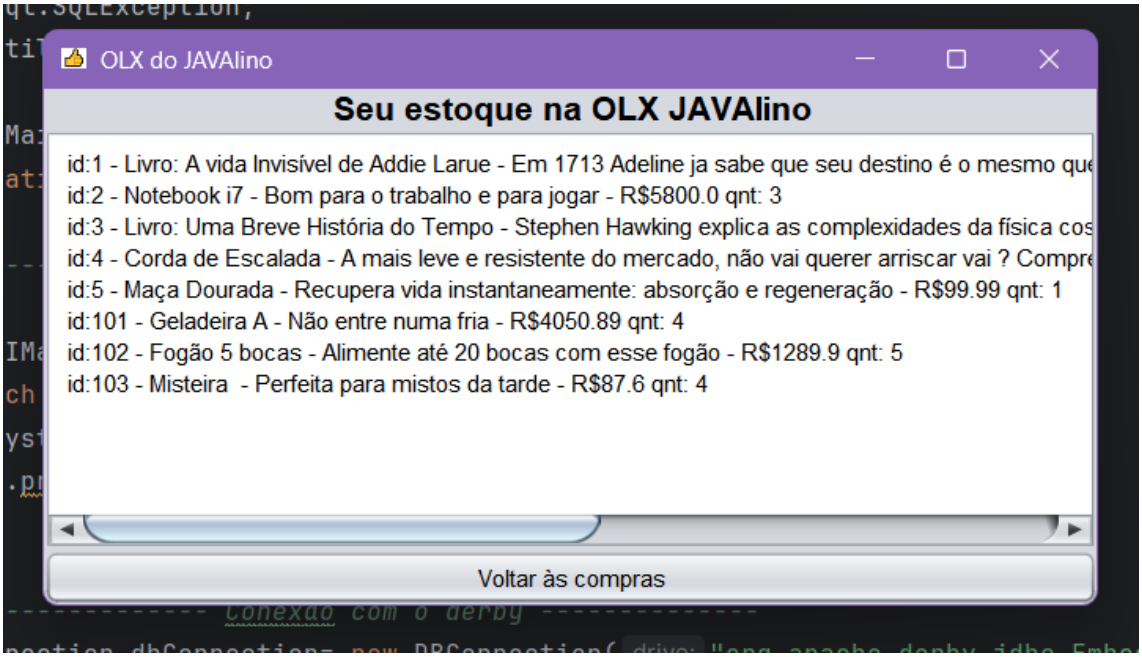
- Se um campo qualquer não tiver o formato correto haverá um pop up avisando, assim a venda é canelada





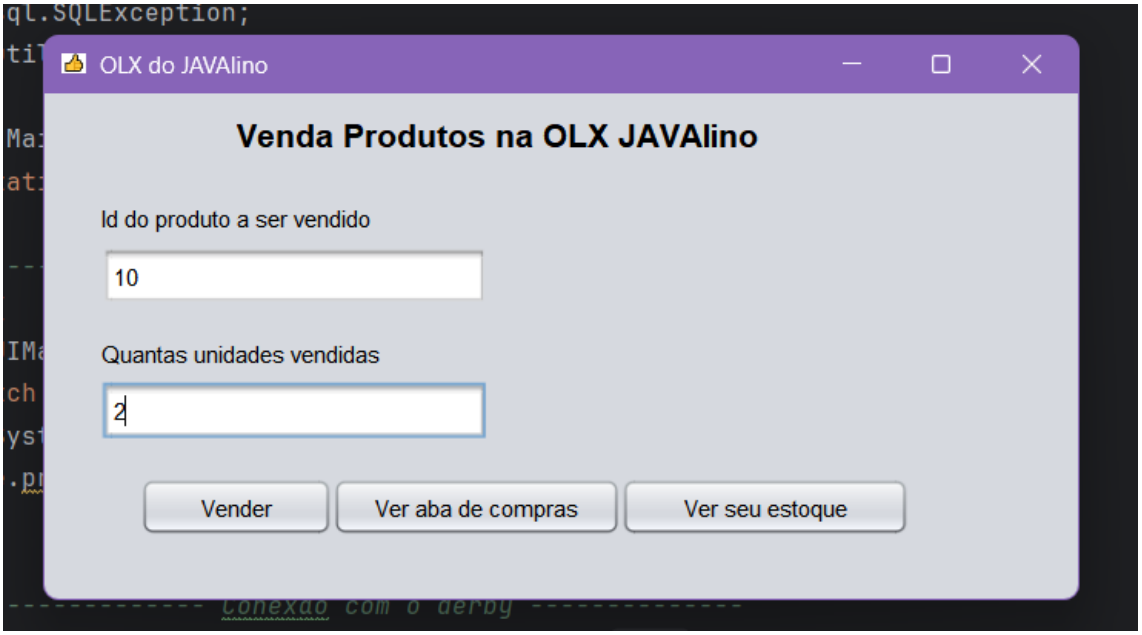
- Quando a aplicação é fechada e reaberta os dados são carregados do banco de dados. As novas comprar recebem id uma centena maior que da abertura anterior. Desse modo, quando abrimos pela segunda vez o primeiro id será 100.

- Depois de algumas adições nosso estoque ficou assim:

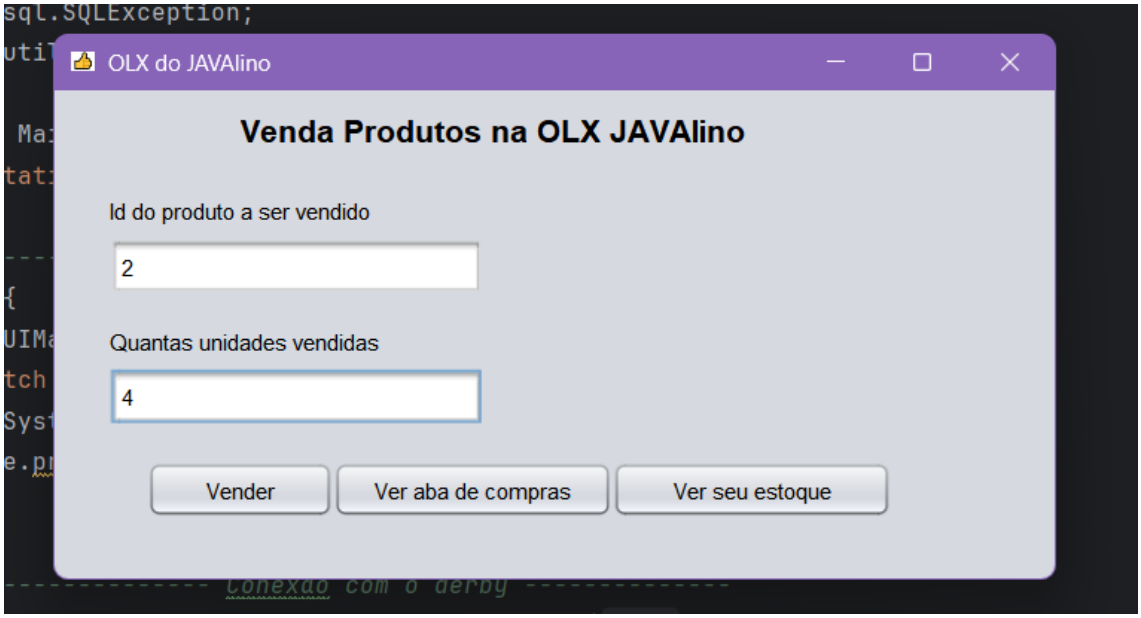


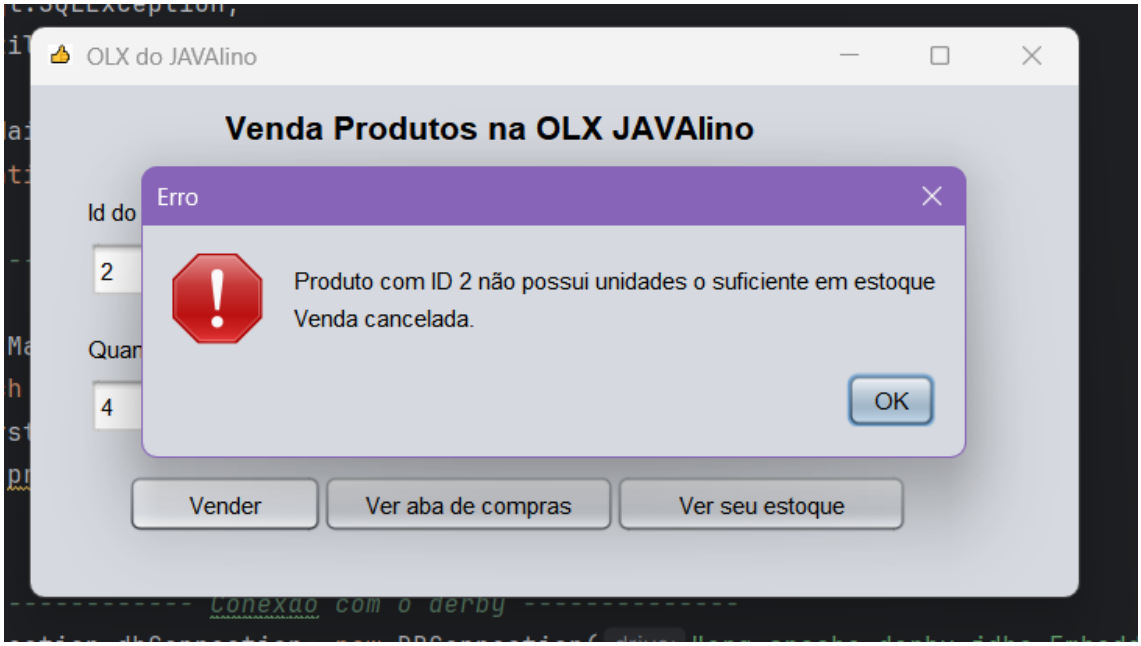
- Agora a venda de produtos

- Vamos tentar vender duas unidades de produto com id 10, o que não existe no estoque:

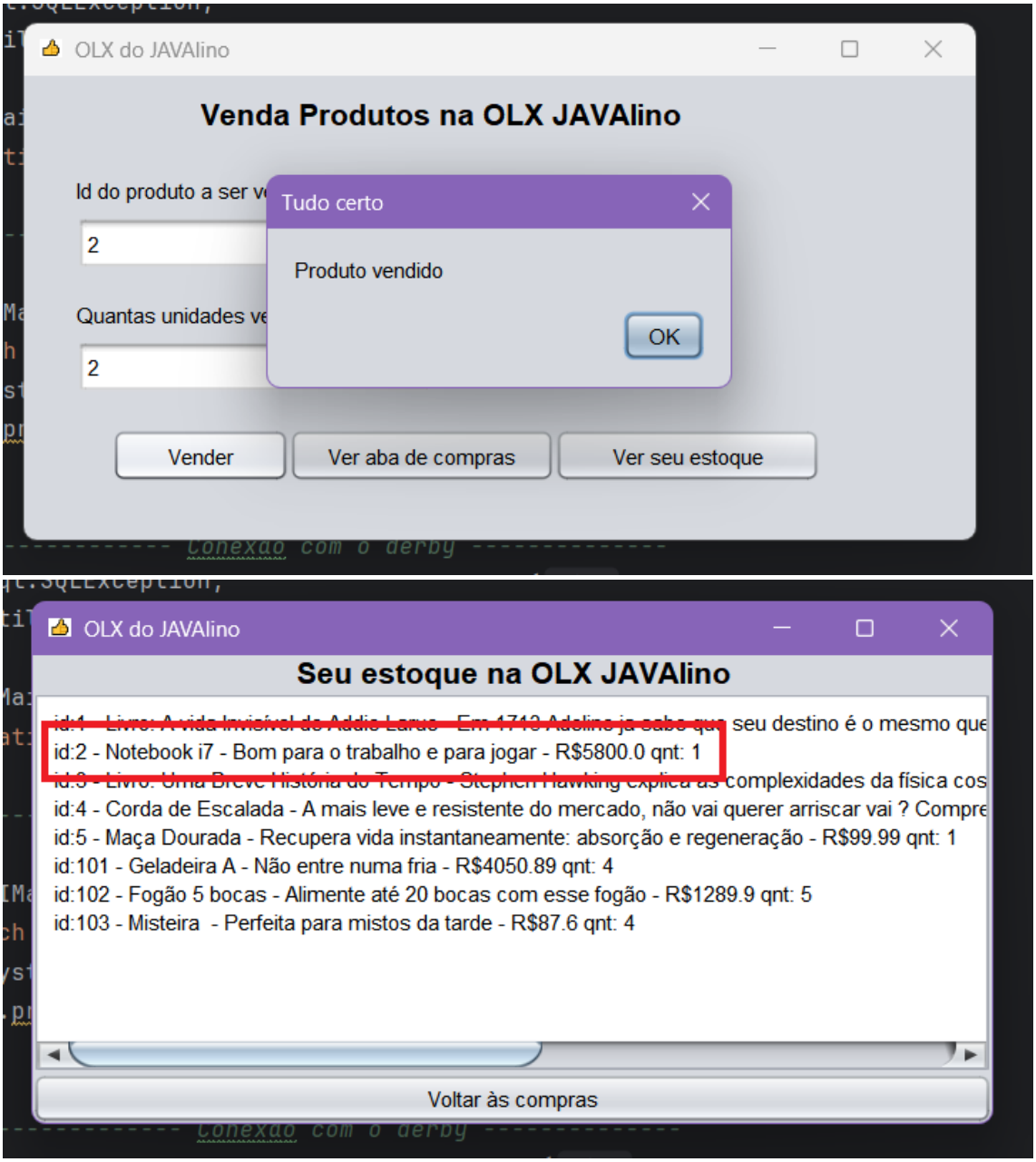


- Vamos vender quatro notebooks i7 de id 2. Existem 3 unidades dele, um erro deve ocorrer:





- Agora vamos vender apenas 2 então. Note que que a quantidade do produto diminuiu de acordo:



- Vamos vender a última unidade dele. Como era o último o sistema logo deleta a tupla desse roduto da tabela: