Estácio

RPG0018 - Por que não paralelizar

202303832338 - Rafael Rosendo Tagliaferro

Campos Interlagos

Objetivos da prática

Criar servidores Java com base em Sockets.

1.Criar clientes síncronos para servidores com base em Sockets.
2.Criar clientes assíncronos para servidores com base em Sockets.
3.Utilizar Threads para implementação de processos paralelos.
4.No final do exercício, o aluno terá criado um servidor Java baseado em Socket, com acesso ao banco de dados via JPA, além de utilizar os recursos nativos do Java para implementação de clientes síncronos e assíncronos. As Threads serão usadas tanto no servidor, para viabilizar múltiplos clientes paralelos, quanto no cliente, para implementar a resposta assíncrona.

1º Procedimento | Criando o Servidor e Cliente de Teste

Análise e Conclusão:

Como funcionam as classes Socket e ServerSocket?

1.Qual a importância das portas para a conexão com servidores?
R: Importante para diferenciar varios serviços, pois um unico serviço esta conectado a uma porta de comunicação.

2.Para que servem as classes de entrada e saída ObjectInputStream e ObjectOutputStream, e por que os objetos transmitidos devem ser serializáveis?
R: InputStream serve para enviar fluxo de bytes nas portas e outputstream serve para ler a entrada de dados feita pelo input, os objetos precisam ser serializaveis para garantir o estado do objeto e a conversão dele em objeto

3.Por que, mesmo utilizando as classes de entidades JPA no cliente, foi possível garantir o isolamento do acesso ao banco de dados?

R: Por que o isolamento do banco é feito atraves da camada server.

2º Procedimento | Servidor Completo e Cliente Assíncrono

1. **Como as Threads podem ser utilizadas para o tratamento assíncrono das respostas enviadas pelo servidor?**

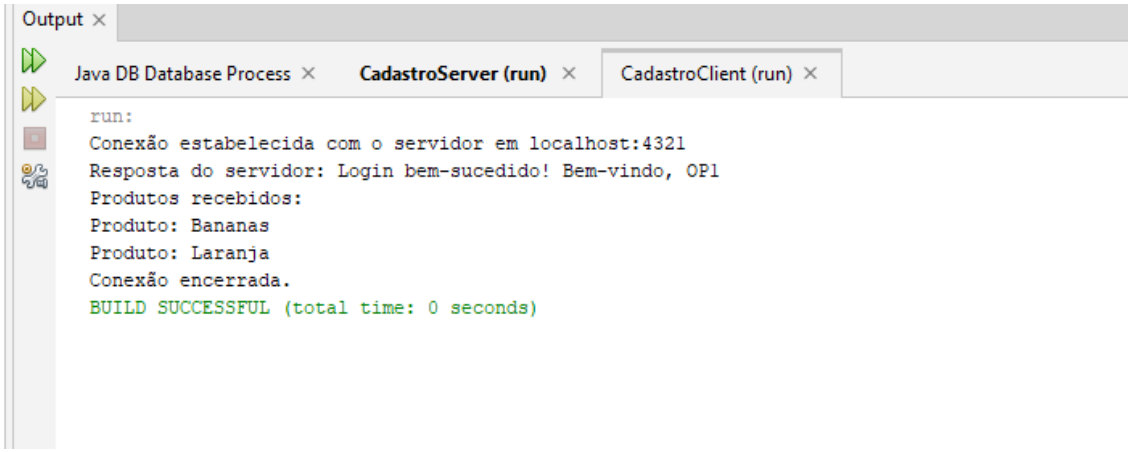
1.Para que serve o método invokeLater, da classe SwingUtilities?
R: Para efetuar a atualização dos graficos dentro da thread, sem ele não teria atualização da interface para o objeto grafico.

2.Como os objetos são enviados e recebidos pelo Socket Java?
R: São enviados e recebidos atraves de sockets e objetos binarios, atraves de serialização das mensagens

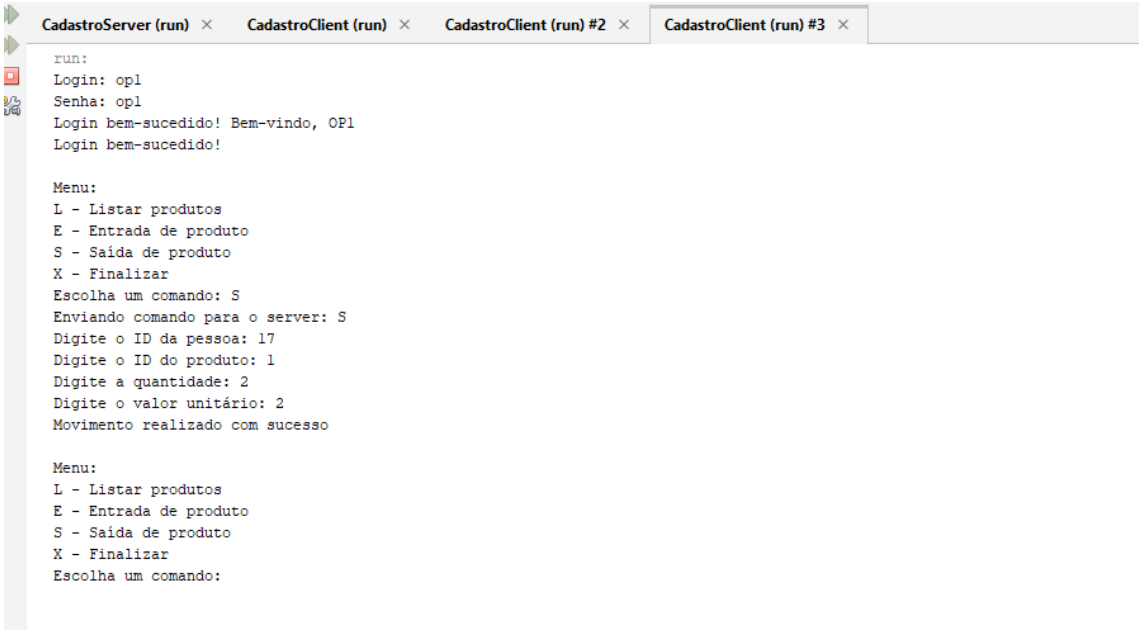
3.Compare a utilização de comportamento assíncrono ou síncrono nos clientes com Socket Java, ressaltando as características relacionadas ao bloqueio do processamento.
R: No comportamento assincrono, a resposta não vem imediatamente e na sincrona, equanto a resposta não chegar, a thread fica aguardando a sua execução.

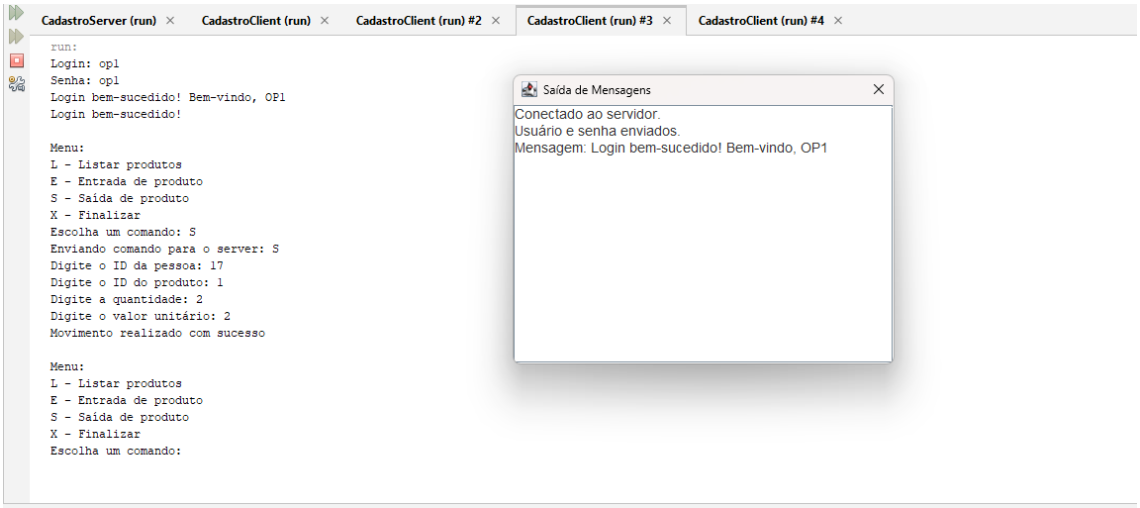
Estácio

Print – 1º Procedimento :



Prints – 2 Procedimento :





Códigos

cadastroclient.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package cadastroclient;

import javax.swing.*;
import java.io.*;
import java.net.Socket;
import java.util.List;
import model.Produtos;

public class ThreadClient extends Thread {
    private ObjectInputStream entrada;
```

```
private ObjectOutputStream saida;
private SaidaFrame saidaFrame;
private final String usuario = "op1"; // Usuário fixo
private final String senha = "op1";   // Senha fixa

public ThreadClient(SaidaFrame saidaFrame) {
    this.saidaFrame = saidaFrame;
}

@Override
public void run() {
    try {
        // Conecta ao servidor
        Socket socket = new Socket("localhost", 4321);
        saidaFrame.texto.append("Conectado ao servidor.\n");

        // Inicializa canais de entrada e saída
        saida = new ObjectOutputStream(socket.getOutputStream());
        entrada = new ObjectInputStream(socket.getInputStream());

        // Envia usuário e senha ao servidor
        saida.writeObject(usuario);
        saida.writeObject(senha);
        saida.flush();
        saidaFrame.texto.append("Usuário e senha enviados.\n");
```

```
// Loop contínuo para leitura do servidor
while (true) {
    Object obj = entrada.readObject();

    // Atualiza o JTextArea com a mensagem recebida
    SwingUtilities.invokeLater(() -> {
        if (obj instanceof String) {
            saidaFrame.texto.append("Mensagem: " + obj + "\n");
        } else if (obj instanceof List) {
            List<?> listaProdutos = (List<?>) obj;
            for (Object item : listaProdutos) {
                if (item instanceof Produtos) {
                    Produtos produto = (Produtos) item;
                    saidaFrame.texto.append(
                        "Produto: " + produto.getNome() +
                        ", Quantidade: " + produto.getQuantidade() + "\n"
                    );
                }
            }
        } else {
            saidaFrame.texto.append("Objeto desconhecido recebido.\n");
        }

        // Atualiza o caret para o final do texto
```

```
saidaFrame.texto.setCaretPosition(saidaFrame.texto.getDocument().getLength
());
                });
            }
        } catch (IOException | ClassNotFoundException e) {
            SwingUtilities.invokeLater(() -> {
                saidaFrame.texto.append("Erro: " + e.getMessage() + "\n");
            });
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        // Cria o frame e inicializa a thread cliente
        JFrame parentFrame = new JFrame("Cliente");
        SaidaFrame saidaFrame = new SaidaFrame(parentFrame);
        saidaFrame.setVisible(true);

        ThreadClient threadClient = new ThreadClient(saidaFrame);
        threadClient.start();
    }
}
```

CadastroClientV2.java

```
/*
```

```
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package cadastroclient;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.math.BigDecimal;
import java.net.Socket;
import java.util.List;
import model.Produtos;

/**
 *
 * @author rafae
 */
public class CadastroClientV2 {
    private static Socket socket;
    private static ObjectOutputStream out;
    private static ObjectInputStream in;
```

```
    private static BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));

    public static void main(String[] args) {
        try {
            // Conectar ao servidor na porta 4321
            socket = new Socket("localhost", 4321);
            out = new ObjectOutputStream(socket.getOutputStream());
            in = new ObjectInputStream(socket.getInputStream());

            // Login e senha
            System.out.print("Login: ");
            String login = reader.readLine();
            System.out.print("Senha: ");
            String senha = reader.readLine();

            // Enviar login e senha para o servidor
            out.writeObject(login);
            out.writeObject(senha);

            // Lê a resposta do servidor (validação do login)
            String resposta = (String) in.readObject();
            System.out.println(resposta);
            if (resposta.startsWith("Login bem-sucedido")) {
                System.out.println("Login bem-sucedido!");
```

```
// Menu - inicio
while (true) {
    exibirMenu();
    String comando = reader.readLine().toUpperCase();
    if ("L".equals(comando)) {
        out.writeObject("L"); // Listar produtos
        out.flush();
        Object obj = in.readObject();
        if (obj instanceof List) {
            List<?> produtos = (List<?>) obj;

            System.out.println("Produtos recebidos:");
            for (Object item : produtos) {
                if (item instanceof Produtos) {
                    Produtos produto = (Produtos) item;
                    System.out.println("Produto: " + produto.getNome());
                }
            }
        } else {
            System.out.println("Erro ao receber lista de produtos.");
        }
    } else if ("E".equals(comando) || "S".equals(comando)) {
        processarMovimento(comando);
        Object obj = in.readObject();
```

```
                System.out.println(obj);
            } else if ("X".equals(comando)) {
                System.out.println("Finalizando o cliente.");
                break;
            } else {
                System.out.println("Comando inválido. Tente novamente.");
            }
        }
    } else {
        System.out.println("Login ou senha inválidos.");
    }

} catch (IOException | ClassNotFoundException e) {
    System.err.println("Erro ao conectar ao servidor: " + e.getMessage());
    e.printStackTrace();
} finally {
    try {
        socket.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
}

private static void exibirMenu() {
```

```
    System.out.println("\nMenu:");
    System.out.println("L - Listar produtos");
    System.out.println("E - Entrada de produto");
    System.out.println("S - Saída de produto");
    System.out.println("X - Finalizar");
    System.out.print("Escolha um comando: ");
}

private static void processarMovimento(String tipo) throws IOException {
    // Enviar tipo de movimento (E ou S)
    System.out.println("Enviando comando para o server: " + tipo);
    out.writeObject(tipo);

    // Obter ID da pessoa
    System.out.print("Digite o ID da pessoa: ");
    Integer pessoaId = Integer.parseInt(reader.readLine());
    out.writeInt(pessoaId);

    // Obter ID do produto
    System.out.print("Digite o ID do produto: ");
    Integer produtoId = Integer.parseInt(reader.readLine());
    out.writeInt(produtoId);

    // Obter quantidade
    System.out.print("Digite a quantidade: ");
```

```
        Integer quantidade = Integer.parseInt(reader.readLine());
        out.writeInt(quantidade);

        // Obter valor unitário
        System.out.print("Digite o valor unitário: ");
        BigDecimal valorUnitario = new BigDecimal(reader.readLine());
        out.writeObject(valorUnitario);

        //;out.flush();
    }

}
```

SaidaFrame.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package cadastroclient;

import java.awt.BorderLayout;
import java.awt.Font;
import javax.swing.JDialog;
```

```
import javax.swing.JFrame;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;

/**
 *
 * @author rafae
 */
public class SaidaFrame extends JDialog {
    public JTextArea texto;

    public SaidaFrame(JFrame parent) {
        super(parent, "Saída de Mensagens", false);

        setBounds(100, 100, 400, 300);

        texto = new JTextArea();
        texto.setEditable(false);
        texto.setFont(new Font("Arial", Font.PLAIN, 14));
        texto.setLineWrap(true);
        texto.setWrapStyleWord(true);

        JScrollPane scrollPane = new JScrollPane(texto);

        getContentPane().add(scrollPane, BorderLayout.CENTER);
```

```
        setDefaultCloseOperation(DISPOSE_ON_CLOSE);
    }

    public void adicionarMensagem(String mensagem) {
        texto.append(mensagem + "\n");
        texto.setCaretPosition(texto.getDocument().getLength() - 1);
        texto.update(texto.getGraphics());
    }

    // Método para exibir a janela
    public void exibir() {
        setVisible(true);
    }
}
```

Threadclient.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package cadastroclient;

import javax.swing.*;
```

```
import java.io.*;
import java.net.Socket;
import java.util.List;
import model.Produtos;

public class ThreadClient extends Thread {
    private ObjectInputStream entrada;
    private ObjectOutputStream saida;
    private SaidaFrame saidaFrame;
    private final String usuario = "op1"; // Usuário fixo
    private final String senha = "op1";   // Senha fixa

    public ThreadClient(SaidaFrame saidaFrame) {
        this.saidaFrame = saidaFrame;
    }

    @Override
    public void run() {
        try {
            // Conecta ao servidor
            Socket socket = new Socket("localhost", 4321);
            saidaFrame.texto.append("Conectado ao servidor.\n");

            // Inicializa canais de entrada e saída
            saida = new ObjectOutputStream(socket.getOutputStream());
```

```
entrada = new ObjectInputStream(socket.getInputStream());

// Envia usuário e senha ao servidor
saida.writeObject(usuario);
saida.writeObject(senha);
saida.flush();
saidaFrame.texto.append("Usuário e senha enviados.\n");

// Loop contínuo para leitura do servidor
while (true) {
    Object obj = entrada.readObject();

    // Atualiza o JTextArea com a mensagem recebida
    SwingUtilities.invokeLater(() -> {
        if (obj instanceof String) {
            saidaFrame.texto.append("Mensagem: " + obj + "\n");
        } else if (obj instanceof List) {
            List<?> listaProdutos = (List<?>) obj;
            for (Object item : listaProdutos) {
                if (item instanceof Produtos) {
                    Produtos produto = (Produtos) item;
                    saidaFrame.texto.append(
                        "Produto: " + produto.getNome() +
                        ", Quantidade: " + produto.getQuantidade() + "\n"
                    );
```

```
                    }
                }
            } else {
                saidaFrame.texto.append("Objeto desconhecido recebido.\n");
            }

            // Atualiza o caret para o final do texto
saidaFrame.texto.setCaretPosition(saidaFrame.texto.getDocument().getLength
());
            });
        }
    } catch (IOException | ClassNotFoundException e) {
        SwingUtilities.invokeLater(() -> {
            saidaFrame.texto.append("Erro: " + e.getMessage() + "\n");
        });
        e.printStackTrace();
    }
}

public static void main(String[] args) {
    // Cria o frame e inicializa a thread cliente
    JFrame parentFrame = new JFrame("Cliente");
    SaidaFrame saidaFrame = new SaidaFrame(parentFrame);
    saidaFrame.setVisible(true);
```

```
        ThreadClient threadClient = new ThreadClient(saidaFrame);
        threadClient.start();
    }
}
```

CadastroServer.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template
 */
package cadastroserver;

import controller.MovimentosJpaController;
import controller.PessoaJpaController;
import controller.ProdutosJpaController;
import controller.UsuariosJpaController;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

/**
 *
 * @author rafae
```

```
 */
public class CadastroServer {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        int port = 4321;

        EntityManagerFactory emf =
Persistence.createEntityManagerFactory("CadastroServerPU");

        ProdutosJpaController ctrl = new ProdutosJpaController(emf);
        UsuariosJpaController ctrlUsu = new UsuariosJpaController(emf);
        MovimentosJpaController ctrlMov = new MovimentosJpaController(emf);
        PessoaJpaController ctrlPessoa = new PessoaJpaController(emf);

        try (ServerSocket serverSocket = new ServerSocket(port)) {
            System.out.println("Servidor iniciado na porta " + port);

            while (true) {
                System.out.println("Aguardando conexões...");
                Socket clientSocket = serverSocket.accept();
                System.out.println("Cliente conectado: " +
clientSocket.getInetAddress());
```

```
            CadastroThread thread = new CadastroThread(ctrl, ctrlUsu, ctrlMov,
ctrl, ctrlPessoa, clientSocket);
            thread.start();


            System.out.println("Thread criada para o cliente " +
clientSocket.getInetAddress());
          }
       } catch (IOException e) {
          System.err.println("Erro no servidor: " + e.getMessage());
       } finally {
          emf.close();
          System.out.println("EntityManagerFactory fechado.");
       }
    }

}
```

CadastroThread.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit
this template
 */
package cadastroserver;

import controller.MovimentosJpaController;
```

```
import controller.PessoaJpaController;
import java.io.*;
import java.net.*;
import java.util.List;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

import controller.ProdutosJpaController;
import controller.UsuariosJpaController;
import java.math.BigDecimal;
import model.Movimentos;
import model.Pessoa;
import model.Usuarios;
import model.Produtos;

/**
 *
 * @author rafae
 */
public class CadastroThread extends Thread {
    private final ProdutosJpaController ctrl;
    private final UsuariosJpaController ctrlUsu;
    private final MovimentosJpaController ctrlMov;
    private final Socket s1;
    private final ProdutosJpaController ctrlProd;
```

```
    private final PessoaJpaController ctrlPessoa;

    public CadastroThread(ProdutosJpaController ctrl, UsuariosJpaController
ctrlUsu, MovimentosJpaController ctrlMov, ProdutosJpaController ctrlProd,
PessoaJpaController ctrlPessoa, Socket s1) {
        this.ctrl = ctrl;
        this.ctrlUsu = ctrlUsu;
        this.s1 = s1;
        this.ctrlMov = ctrlMov;
        this.ctrlProd = ctrlProd;
        this.ctrlPessoa = ctrlPessoa;
    }

    @Override
    public void run() {
        try (
            ObjectOutputStream out = new
ObjectOutputStream(s1.getOutputStream());
            ObjectInputStream in = new ObjectInputStream(s1.getInputStream());
        ) {

            String login = (String) in.readObject();
            String senha = (String) in.readObject();

            Usuarios usuario = ctrlUsu.findUsuarioByLoginAndSenha(login, senha);
            if (usuario == null) {
```

```
            out.writeObject("Acesso negado! Login ou senha inválidos.");
            s1.close();
            return;
        }

        out.writeObject("Login bem-sucedido! Bem-vindo, " +
usuario.getLogin());
        //out.writeChars("RTeste");
        //out.flush();
        boolean ativo = true;

        while (ativo) {
            // Obter o comando do cliente
            String comando = (String) in.readObject();

            switch (comando) {
                case "L": // Listar produtos
                    List<Produtos> produtos = ctrl.findProdutoEntities();
                    //System.out.println(produtos.toString());
                    out.writeObject(produtos);
                    break;
                case "S":
                case "E":
                    Integer idPessoa = in.readInt();
                    System.out.println(idPessoa);
```

```
Integer idProduto = in.readInt();
System.out.println(idProduto);
Integer quantidade = in.readInt();
System.out.println(quantidade);
BigDecimal valorUnitario = (BigDecimal) in.readObject();
System.out.println(valorUnitario);

System.out.println(comando);

 //out.writeObject("Produto não encontrado");

Produtos produto = ctrlProd.findProdutoById(idProduto);
if (produto == null) {
    out.writeObject("Produto não encontrado");
    continue;
}

Pessoa pessoa = ctrlPessoa.findPessoaById(idPessoa);
if (pessoa == null) {
    out.writeObject("Pessoa não encontrado");
    continue;
}

// Criar o objeto Movimento
Movimentos movimento = new Movimentos();
```

```
                    movimento.setTipo(comando);
                    movimento.setUsuarioID(usuario);
                    movimento.setPessoaID(idPessoa);
                    movimento.setProdutoID(produto);
                    movimento.setQuantidade(quantidade);
                    movimento.setPrecoUnitario(valorUnitario);

                    System.out.println("Gravando movimento");
                    ctrlMov.create(movimento);

                    if ("E".equals(comando)) {
                        produto.setQuantidade(produto.getQuantidade() +
quantidade);
                    } else if ("S".equals(comando)) {
                        if (produto.getQuantidade() >= quantidade) {
                            produto.setQuantidade(produto.getQuantidade() -
quantidade);
                        } else {
                            out.writeObject("Quantidade insuficiente para saída");
                            continue;
                        }
                    }
                    ctrlProd.edit(produto);
                    out.writeObject("Movimento realizado com sucesso");
                case "X": // Sair
                    ativo = false;
```

```
                    out.writeObject("Conexão encerrada.");
                    break;

                default:
                    out.writeObject("Comando inválido2.");
                    break;
            }
        }

        s1.close();
    } catch (IOException | ClassNotFoundException e) {
        System.err.println("Erro na comunicação com o cliente: " +
e.getMessage());
    }
  }
}
```