



# Estácio

## Universidade Estácio de Sá

- 
- Curso: Desenvolvimento Full stack - TURMA 9001
  - Disciplina: RPG0018 - Por que não paralelizar.
  - Semestre Letivo: 2024.4
  - Aluno: Jonathan Sendi Inowe
  - Matricula: 202311117502
- 

### Missão Prática | Nível 5 | Mundo 3

**Servidores e clientes baseados em Socket, com uso de Threads tanto no lado cliente quanto no lado servidor, acessando o banco de dados via JPA.**

**Procedimento 1: Criando o Servidor e Cliente de Teste**

**Procedimento 2: Servidor Completo e Cliente Assíncrono**

---



### Objetivos da Prática

- Criar servidores Java com base em Sockets.
  - Criar clientes síncronos para servidores com base em Sockets.
  - Criar clientes assíncronos para servidores com base em Sockets.
  - Utilizar Threads para implementação de processos paralelos.
  - 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.
- 

### Códigos

## Procedimento 1: Criando o Servidor e Cliente de Teste

- CadastroClient.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 cadastroclient;

import java.io.IOException;
import java.io.InputStreamReader;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.PrintStream;
import java.net.Socket;
import java.util.List;
import java.util.Scanner;
import model.Produto;

public class CadastroClient {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws ClassNotFoundException, IOException {
        Socket socket = new Socket("localhost", 4321);
        ObjectOutputStream out = new ObjectOutputStream(socket.getOutputStream());
        ObjectInputStream in = new ObjectInputStream(socket.getInputStream());

        out.writeObject("op1");

        out.writeObject("op1");

        System.out.println((String)in.readObject());

        out.writeObject("L");

        List<Produto> produtos = (List<Produto>) in.readObject();
        for (Produto produto : produtos) {
            System.out.println(produto.getNome());
        }

        out.close();
        in.close();
        socket.close();
    }
}
```

- 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 cadastrserver;
import controller.ProdutoJpaController;
import controller.UsuarioJpaController;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.net.ServerSocket;
import java.net.Socket;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

public class CadastroServer {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws IOException{
        ServerSocket serverSocket = new ServerSocket(4321);
        EntityManagerFactory emf = Persistence.createEntityManagerFactory("CadastroServerPU");
        ProdutoJpaController ctrl = new ProdutoJpaController(emf);
        UsuarioJpaController ctrlUsu = new UsuarioJpaController(emf);

        while (true) {

            Socket clienteSocket = serverSocket.accept();
            System.out.println("Cliente conectado: " + clienteSocket.getInetAddress());

            CadastroThread thread = new CadastroThread(ctrl, ctrlUsu, clienteSocket);

            thread.start();
            System.out.println("Aguardando nova conexÃ£o...");
        }
    }
}

```

- 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 cadastrserver;

import controller.ProdutoJpaController;
import controller.UsuarioJpaController;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.ObjectInputStream;

```

```
import java.io.ObjectOutputStream;
import java.net.Socket;
import java.util.Scanner;
import model.Usuario;
```

```
public class CadastroThread extends Thread {
```

```
    private ProdutoJpaController ctrl;
    private UsuarioJpaController ctrlUsu;
    private Socket s1;
    private ObjectOutputStream out;
    private ObjectInputStream in;
```

```
    CadastroThread (ProdutoJpaController ctrl, UsuarioJpaController ctrlUsu, Socket s1) {
        this.ctrl = ctrl;
        this.ctrlUsu = ctrlUsu;
        this.s1 = s1;
    }
```

```
    @Override
```

```
    public void run(){
```

```
        String login = "";
```

```
        try{
```

```
            out = new ObjectOutputStream(s1.getOutputStream());
            in = new ObjectInputStream(s1.getInputStream());
```

```
            System.out.println("Cliente conectado.");
```

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

```
            Usuario usuario = ctrlUsu.findUsuario(login, senha);
            if (usuario == null) {
                System.out.println("Usuário inválido."); //Login="+ login +", Senha="+ senha
                out.writeObject("Usuário inválido.");
                return;
            }
```

```
            System.out.println("Usuário conectado.");
            out.writeObject("Usuário conectado.");
```

```
            System.out.println("Aguardando comandos...");
            String comando = (String) in.readObject();
```

```
            if (comando.equals("L")) {
                System.out.println("Listando produtos.");
                out.writeObject(ctrl.findProdutoEntities());
            }
```

```

    }
} catch (IOException | ClassNotFoundException e) {
    e.printStackTrace();
} finally {
    close();
    System.out.println("Conexão finalizada.");
}
}

private void close() {
    try {
        if (out != null) {
            out.close();
        }
        if (in != null) {
            in.close();
        }
        if (s1 != null) {
            s1.close();
        }
    } catch (IOException ex) {
        System.out.println("Falha ao finalizar conexão.");
    }
}
}

```

## Procedimento 2: Servidor Completo e Cliente Assíncrono

- CadastroClientv2.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 cadastroclient;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.PrintStream;
import java.net.Socket;
import java.util.List;
import java.util.Scanner;
import model.Produto;

public class CadastroClientv2 {

    private static ObjectOutputStream socketOut;
    private static ObjectInputStream socketIn;

```

```
private static ThreadClient threadClient;
```

```
/**  
 * @param args the command line arguments  
 */
```

```
public static void main(String[] args) throws ClassNotFoundException, IOException {  
    Socket socket = new Socket("localhost", 4321);  
    socketOut = new ObjectOutputStream(socket.getOutputStream());  
    socketIn = new ObjectInputStream(socket.getInputStream());  
  
    BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));  
  
    SaidaFrame saidaFrame = new SaidaFrame();  
    saidaFrame.setVisible(true);  
  
    threadClient = new ThreadClient(socketIn, saidaFrame.texto);  
    threadClient.start();  
  
    socketOut.writeObject("op1");  
  
    socketOut.writeObject("op1");  
  
    Character commando = ' ';  
    try {  
        while (!commando.equals('X')) {  
            System.out.println("Escolha uma opção:");  
            System.out.println("L - Listar | X - Finalizar | E - Entrada | S - Saida");  
  
            commando = reader.readLine().charAt(0);  
  
            processaComando(reader, commando);  
        }  
    } catch (Exception e) {  
        e.printStackTrace();  
    } finally {  
        saidaFrame.dispose();  
        socketOut.close();  
        socketIn.close();  
        socket.close();  
        reader.close();  
    }  
}
```

```
static void processaComando(BufferedReader reader, Character commando) throws IOException {  
    socketOut.writeChar(commando);  
    socketOut.flush();  
  
    switch (commando) {  
        case 'L':
```

```

        break;
    case 'S':
    case 'E':
        socketOut.flush();

        System.out.println("Digite o Id da pessoa:");
        int idPessoa = Integer.parseInt(reader.readLine());
        System.out.println("Digite o Id do produto:");
        int idProduto = Integer.parseInt(reader.readLine());
        System.out.println("Digite a quantidade:");
        int quantidade = Integer.parseInt(reader.readLine());
        System.out.println("Digite o valor unitário:");
        long valorUnitario = Long.parseLong(reader.readLine());

        socketOut.writeInt(idPessoa);
        socketOut.flush();
        socketOut.writeInt(idProduto);
        socketOut.flush();
        socketOut.writeInt(quantidade);
        socketOut.flush();
        socketOut.writeLong(valorUnitario);
        socketOut.flush();
        break;
    case 'X':
        threadClient.cancela();
        break;
    default:
        System.out.println("Operação inválida!");
    }
}
}
}

```

## ● SaidaFrame

```

/*
 * 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.*.*;

public class SaidaFrame extends JDialog {
    public JTextArea texto;

    public SaidaFrame() {
        setBounds(100, 100, 400, 300);

        setModal(false);

        texto = new JTextArea(25, 40);
        texto.setEditable(false); // Bloqueia edição do campo de texto

        JScrollPane scroll = new JScrollPane(texto);
    }
}

```

```

        scroll.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);
// Bloqueia rolagem horizontal
        add(scroll);
    }
}

```

- 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 java.io.IOException;
import java.io.ObjectInputStream;
import java.net.SocketException;
import java.util.List;
import javax.swing.JTextArea;
import javax.swing.SwingUtilities;
import model.Produto;

```

```

public class ThreadClient extends Thread {
    private ObjectInputStream entrada;
    private JTextArea textArea;
    private Boolean cancelada;

    public ThreadClient(ObjectInputStream entrada, JTextArea textArea) {
        this.entrada = entrada;
        this.textArea = textArea;
        this.cancelada = false;
    }

```

```

@Override
public void run() {
    while (!cancelada) {
        try {
            Object resposta = entrada.readObject();
            SwingUtilities.invokeLater(() -> {
                processaResposta(resposta);
            });
        } catch (IOException | ClassNotFoundException e) {
            if (!cancelada) {
                System.err.println(e);
            }
        }
    }
}

```

```

public void cancela() {
    cancelada = true;
}

```

```

private void processaResposta(Object resposta) {

```



```

        textArea.append(">> Nova comunica  o em " + java.time.LocalDateTime.now() + ":\n");

        if (resposta instanceof String) {
            textArea.append((String) resposta + "\n");
        } else if (resposta instanceof List<?>) {
            textArea.append("> Listagem dos produtos:\n");
            List<Produto> lista = (List<Produto>) resposta;
            for (Produto item : lista) {
                textArea.append("Produto=[" + item.getNome() + "], Quantidade=[" + item.getQuantidade() +
                "]\n");
            }
        }
        textArea.append("\n");
        textArea.setCaretPosition(textArea.getDocument().getLength());
    }
}

```

- 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.MovimentoJpaController;
import controller.PessoaJpaController;
import controller.ProdutoJpaController;
import controller.UsuarioJpaController;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.net.ServerSocket;
import java.net.Socket;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

public class CadastroServer {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws IOException{
        ServerSocket serverSocket = new ServerSocket(4321);
        EntityManagerFactory emf = Persistence.createEntityManagerFactory("CadastroServerPU");
        ProdutoJpaController ctrl = new ProdutoJpaController(emf);
        UsuarioJpaController ctrlUsu = new UsuarioJpaController(emf);
        MovimentoJpaController ctrlMov = new MovimentoJpaController(emf);
        PessoaJpaController ctrlPessoa = new PessoaJpaController(emf);

        while (true) {
            Socket clienteSocket = serverSocket.accept();

```

```

        System.out.println("Cliente conectado: ");

        CadastroThreadv2 thread = new CadastroThreadv2(ctrl, ctrlUsu, ctrlMov, ctrlPessoa,
clienteSocket);

        thread.start();
        System.out.println("Aguardando nova conexÃ£o...");
    }

}
}

```

- CadastroThreadv2.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.MovimentoJpaController;
import controller.PessoaJpaController;
import controller.ProdutoJpaController;
import controller.UsuarioJpaController;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;
import model.Movimento;
import model.Produto;
import model.Usuario;

public class CadastroThreadv2 extends Thread {

    private ProdutoJpaController ctrl;
    private UsuarioJpaController ctrlUsu;
    private MovimentoJpaController ctrlMov;
    private PessoaJpaController ctrlPessoa;
    private Socket s1;
    private ObjectOutputStream out;
    private ObjectInputStream in;
    private Usuario usuario;
    private Boolean continuaProcesso = true;

    CadastroThreadv2 (ProdutoJpaController ctrl, UsuarioJpaController ctrlUsu,
MovimentoJpaController ctrlMov, PessoaJpaController ctrlPessoa, Socket s1) {
        this.ctrl = ctrl;
        this.ctrlUsu = ctrlUsu;
    }
}

```

```
this.ctrlMov = ctrlMov;
this.ctrlPessoa = ctrlPessoa;
this.s1 = s1;
}
```

```
@Override
public void run(){
```

```
    String login = "";
```

```
    try{
        out = new ObjectOutputStream(s1.getOutputStream());
        in = new ObjectInputStream(s1.getInputStream());
```

```
        System.out.println("Cliente conectado.");
```

```
        login = (String) in.readObject();
        String senha = (String) in.readObject();
        usuario = ctrlUsu.findUsuario(login, senha);
```

```
        if (usuario == null) {
            System.out.println("Usuário inválido.");
            out.writeObject("Usuário inválido.");
            return;
        }
```

```
        System.out.println("Usuário conectado.");
        out.writeObject("Usuário conectado.");
        out.flush();
```

```
        while (continuaProcesso) {
            continuaProcesso = processaComando();
        }
```

```
    } catch (IOException | ClassNotFoundException e) {
        e.printStackTrace();
    } catch (Exception ex) {
        Logger.getLogger(CadastroThreadv2.class.getName()).log(Level.SEVERE, null, ex);
    } finally {
        close();
        System.out.println("Conexão finalizada.");
    }
}
```

```
}
```

```
private Boolean processaComando() throws Exception {
    System.out.println("Aguardando comandos...");
    Character comando = in.readChar();
```

```

switch (comando) {
    case 'L':
        System.out.println("Comando recebido, listando produtos.");
        out.writeObject(ctrl.findProdutoEntities());
        continuaProcesso = true;
        return true;
    case 'E':
        continuaProcesso = true;
        return true;
    case 'S':
        System.out.println("Comando Movimento tipo [" + comando + "] recebido.");
        int idPessoa = in.readInt();
        int idProduto = in.readInt();
        int quantidade = in.readInt();
        Float valorUnitario = in.readFloat();

        Produto produto = ctrl.findProduto(idProduto);
        if (produto == null) {
            out.writeObject("Produto inválido.");
            continuaProcesso = true;
            return true;
        }

        if (comando.equals('E')) {
            produto.setQuantidade(produto.getQuantidade() + quantidade);
            continuaProcesso = true;
            return true;
        } else if (comando.equals('S')) {
            produto.setQuantidade(produto.getQuantidade() - quantidade);
            continuaProcesso = true;
            return true;
        }

        ctrl.edit(produto);

        Movimento movimento = new Movimento();
        movimento.setTipo(comando);
        movimento.setUsuarioidUsuario(usuario);
        movimento.setPessoaidPessoa(ctrlPessoa.findPessoa(idPessoa));
        movimento.setProdutoidproduto(produto);
        movimento.setQuantidade(quantidade);
        movimento.setValorUnitario(valorUnitario);

        ctrlMov.create(movimento);
        out.writeObject("Movimento registrado com sucesso.");
        out.flush();
        System.out.println("Movimento registrado com sucesso.");
        continuaProcesso = true;
        return true;
    case 'X':
        continuaProcesso = false;
        return false;
}

```

```

        default:
            System.out.println("Operação inválida!");
            continuaProcesso = false;
            return true;
        }
    }

    private void close() {
        try {
            if (out != null) {
                out.close();
            }
            if (in != null) {
                in.close();
            }
            if (s1 != null) {
                s1.close();
            }
        } catch (IOException ex) {
            System.out.println("Falha ao fechar conexão.");
        }
    }
}

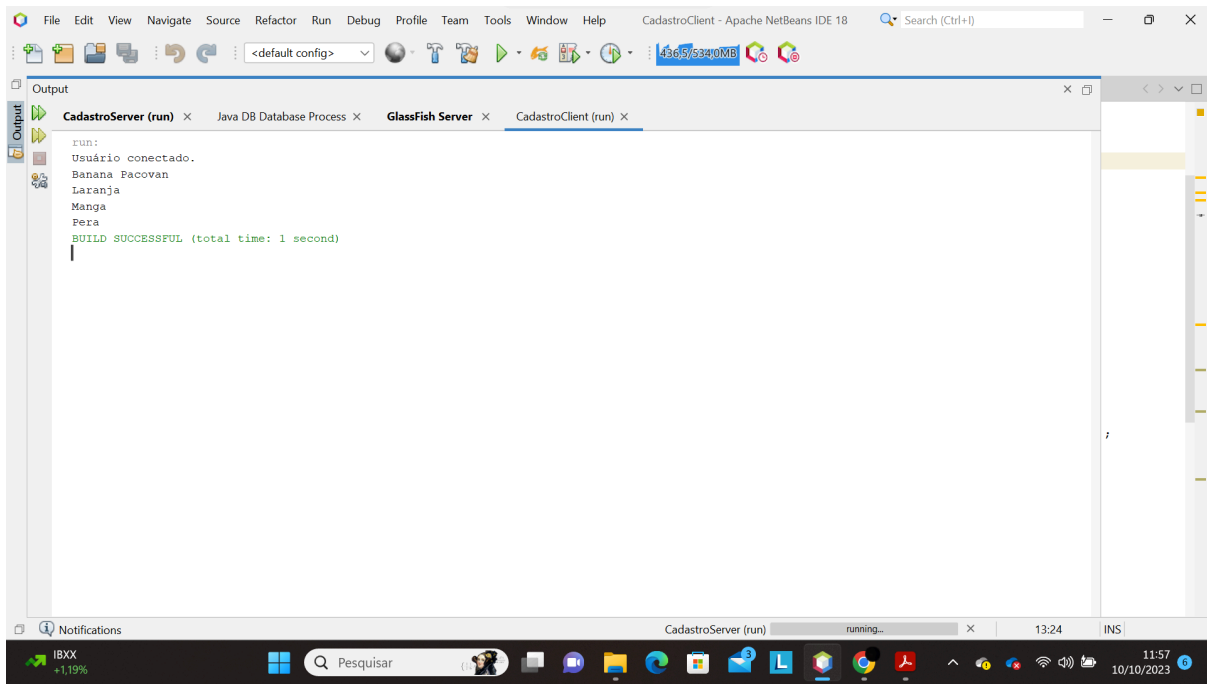
```

---

## Resultados:

 Procedimento 1:

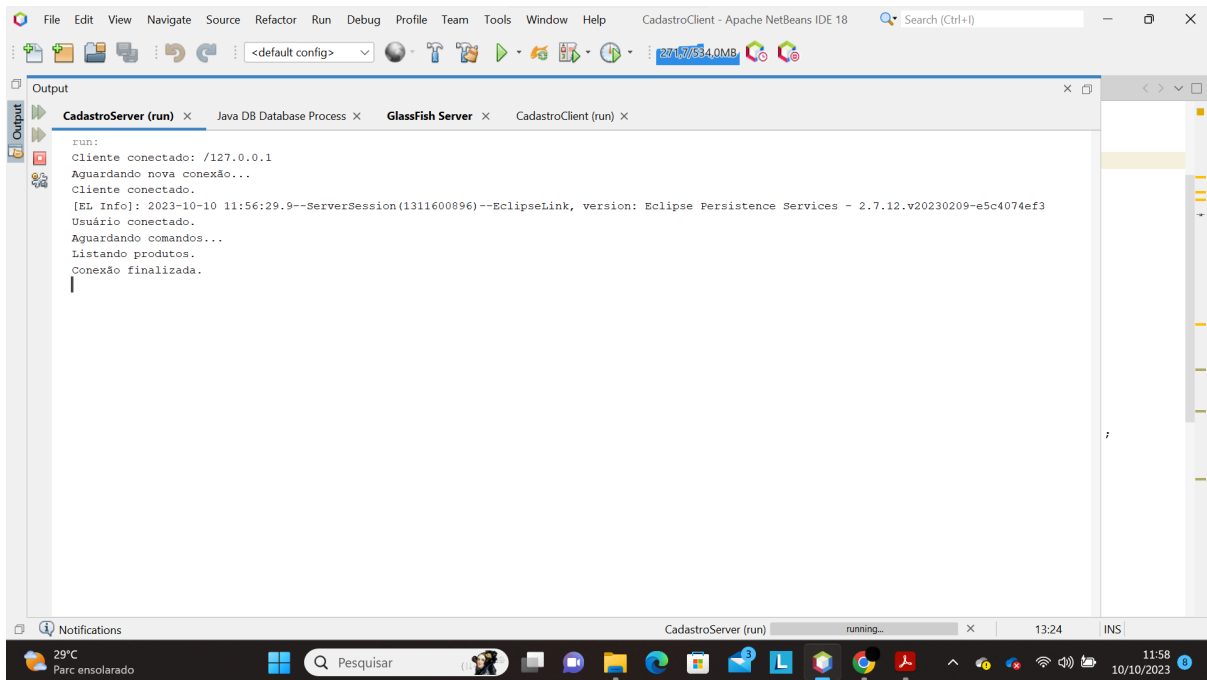
<https://github.com/joninowe/Missao-nivel-5-mundo-3/tree/main/Procedimento%201>



The screenshot shows the NetBeans IDE interface with the 'CadastroClient' project selected. The 'Output' window displays the following text:

```
run:
Usuário conectado.
Banana Pacovan
Laranja
Manga
Pera
BUILD SUCCESSFUL (total time: 1 second)
```

The taskbar at the bottom shows the system clock as 11:57 on 10/10/2023.

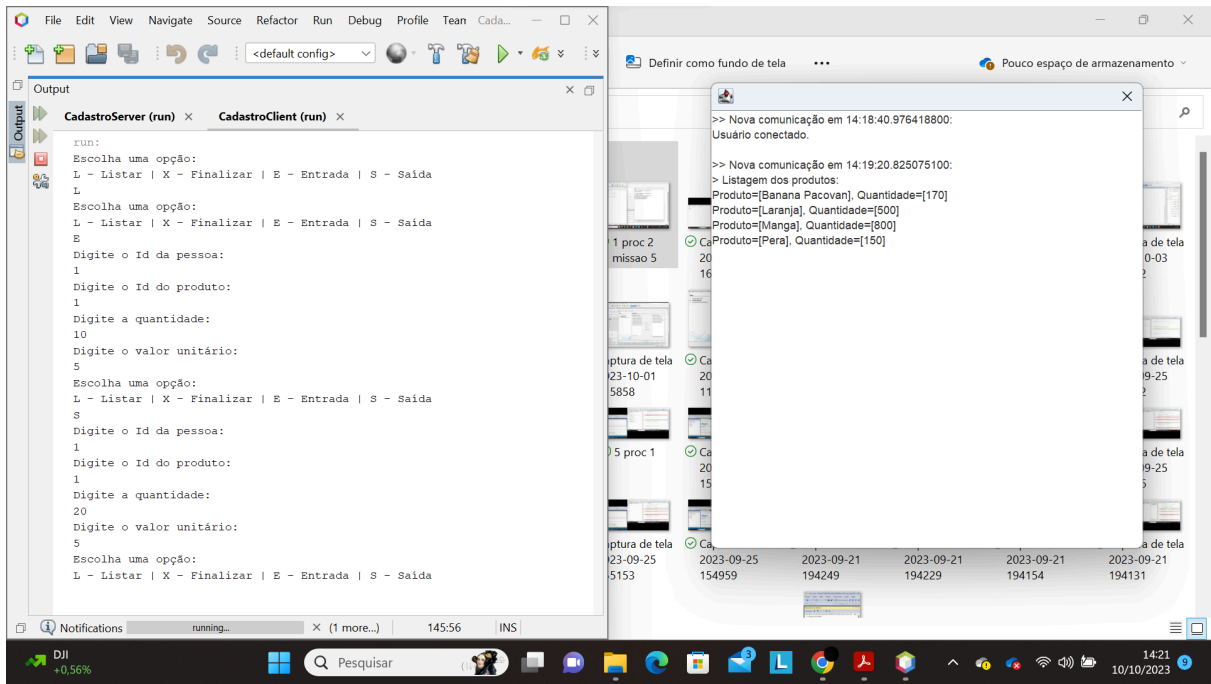


The screenshot shows the NetBeans IDE interface with the 'CadastroClient' project selected. The 'Output' window displays the following text:

```
run:
Cliente conectado: /127.0.0.1
Aguardando nova conexão...
Cliente conectado.
[EL Info]: 2023-10-10 11:56:29.9--ServerSession(1311600896)--EclipseLink, version: Eclipse Persistence Services - 2.7.12.v20230209-e5c4074ef3
Usuário conectado.
Aguardando comandos...
Listando produtos.
Conexão finalizada.
```

The taskbar at the bottom shows the system clock as 11:58 on 10/10/2023.

🚩 Procedimento 2:  
(<https://github.com/joninowe/Missao-nivel-5-mundo-3/tree/main/Procedimento%202>)



```
Escolha uma opção:  
L - Listar | X - Finalizar | E - Entrada | S - Saída  
L  
Escolha uma opção:  
L - Listar | X - Finalizar | E - Entrada | S - Saída  
E  
Digite o Id da pessoa:  
1  
Digite o Id do produto:  
1  
Digite a quantidade:  
10  
Digite o valor unitário:  
5  
Escolha uma opção:  
L - Listar | X - Finalizar | E - Entrada | S - Saída  
S  
Digite o Id da pessoa:  
1  
Digite o Id do produto:  
1  
Digite a quantidade:  
20  
Digite o valor unitário:  
5  
Escolha uma opção:  
L - Listar | X - Finalizar | E - Entrada | S - Saída  
X  
BUILD SUCCESSFUL (total time: 3 minutes 1 second)
```

## Análise e Conclusão

- Como funcionam as classes Socket e ServerSocket?  
No servidor, ele é usado para aguardar e aceitar conexões da rede. Já no cliente, o Socket é utilizado para iniciar uma conexão com o servidor.
- Qual a importância das portas para a conexão com servidores?  
Permitem que o cliente e o servidor se comuniquem entre si criando um canal identificado, evitando conflitos.
- Para que servem as classes de entrada e saída ObjectInputStream e ObjectOutputStream, e por que os objetos transmitidos devem ser serializáveis?  
Permitem que objetos sejam convertidos em um formato serializado, ou seja,



transformados em uma representação que pode ser transmitida ou armazenada. A serialização é fundamental para o envio de objetos pela rede ou seu armazenamento em arquivos, pois converte os dados em bytes, possibilitando sua reconstrução futura.

- Por que, mesmo utilizando as classes de entidades JPA no cliente, foi possível garantir o isolamento do acesso ao banco de dados?  
A responsabilidade pelo acesso ao banco de dados recai sobre as classes Controllers, que, neste caso, estão presentes apenas no lado do Servidor. Isso assegura o isolamento do acesso ao banco de dados.
- Como as Threads podem ser utilizadas para o tratamento assíncrono das respostas enviadas pelo servidor?  
Com o uso de threads no cliente, a atualização dos dados na interface pode ser realizada pela classe SaidaFrame, que herda de JDialog, sem interromper o processo principal ou bloquear a interface. Isso possibilita que o cliente continue constantemente “ouvindo” as respostas do servidor.
- Para que serve o método invokeLater, da classe SwingUtilities?  
Serve para agendar a execução de um trecho de código na Event Dispatch Thread (EDT) do Swing. Essa abordagem é fundamental para garantir que as operações da interface gráfica sejam processadas na thread correta, evitando problemas de concorrência e mantendo a responsividade da interface em aplicativos Swing.
- Como os objetos são enviados e recebidos pelo Socket Java?  
O envio e recebimento de objetos via Socket é realizado por meio das classes ObjectOutputStream e ObjectInputStream. Para transmitir um objeto, utiliza-se o método writeObject() da classe ObjectOutputStream, passando o objeto como argumento. Já para receber um objeto, emprega-se o método readObject() da classe ObjectInputStream. Além desses, existem métodos específicos para transmissão de diferentes tipos de dados, como writeChar(), writeInt(), writeLong(), readChar(), readInt(), readLong(), entre outros.
- Compare a utilização de comportamento assíncrono ou síncrono nos clientes com Socket Java, ressaltando as características relacionadas ao bloqueio do processamento.  
No síncrono, as operações de socket interrompem a execução do cliente até serem concluídas, fazendo com que ele aguarde a resposta do servidor antes de prosseguir. Já no modelo assíncrono, essas operações ocorrem em segundo plano, sem bloquear o cliente. Isso possibilita que outras tarefas sejam executadas simultaneamente, garantindo maior responsividade ao aplicativo e evitando atrasos no processamento.