Estácio

Implementação de um cadastro de clientes em modo texto, com persistência em arquivos, baseado na tecnologia Java.

André Luis Gonçalves Carvalhal - Matrícula: 20220318540

Campus Barra World - Desenvolvimento Full-Stack
Por que não paralelizar – RPG0018 – 9001 – 2023.3

Objetivo da Prática

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

Link GitHub
https://github.com/ANDREC1986/RPG0018-202203185403

1º Procedimento | Criando o Servidor e Cliente de Teste

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.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

/**
 *
 * @author andre
 */
public class CadastroServer {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("CadastroServerPU");
    ProdutoJpaController ctrl = new ProdutoJpaController(emf);
    UsuarioJpaController ctrlUsu = new UsuarioJpaController(emf);
    PessoaJpaController ctrlPessoa = new PessoaJpaController(emf);
```

```
    MovimentoJpaController ctrlMov = new MovimentoJpaController(emf);
        try {
            ServerSocket serverSocket= new ServerSocket(4321);
            while(true) {
                Socket cliente = serverSocket.accept();
                Thread usrConnect = new Thread(new CadastroThread(ctrl,ctrlUsu,cliente));
                usrConnect.start();
            }
        } catch (IOException ex) {
            Logger.getLogger(CadastroServer.class.getName()).log(Level.SEVERE, null, ex);
        }

    }

}
```

CadastroThread.java

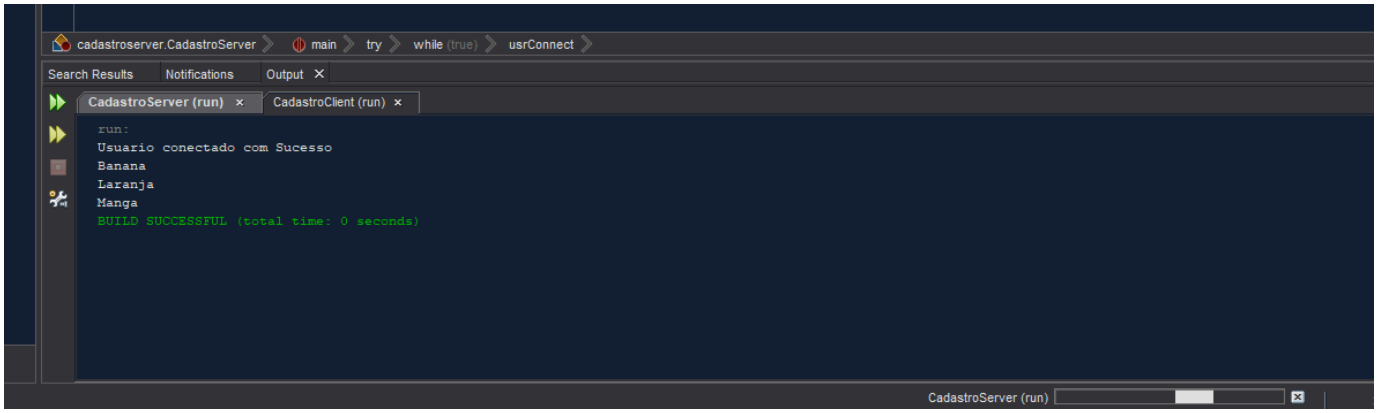
```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change
this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this
template
 */
package cadastroserver;

import controller.ProdutoJpaController;
import controller.UsuarioJpaController;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;
import java.util.logging.Level;
import java.util.logging.Logger;
import model.Usuario;

/**
 *
 * @author andre
 */
public class CadastroThread extends Thread {
    private ProdutoJpaController ctrl;
```

```
    private UsuarioJpaController ctrlUsu;
    private Socket s1;
    private boolean state;

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

@Override
public void run() {
        try {
            ObjectOutputStream saida = new ObjectOutputStream(s1.getOutputStream());
            ObjectInputStream entrada = new ObjectInputStream(s1.getInputStream());
            String login = (String) entrada.readObject();
            String senha = (String) entrada.readObject();
            Usuario usuario = ctrlUsu.findUsuario(login, senha);

            if (usuario == null) {
                saida.writeObject(null);
                s1.close();
                return;
            }

            while(this.state == true) {
                String comando = (String) entrada.readObject();
                if("l".equals(comando.toLowerCase())) {
                    saida.writeObject(ctrl.findProdutoEntities());
                }
                this.state=false;
            }

        } catch (IOException | ClassNotFoundException ex) {
            Logger.getLogger(CadastroThread.class.getName()).log(Level.SEVERE, null, ex);
        }
}

}
```

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.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import model.Produto;

/**
 *
 * @author andre
 */
public class CadastroClient {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        try {
            // TODO code application logic here
            Socket socket = new Socket("localhost", 4321);
            ObjectOutputStream saida = new ObjectOutputStream(socket.getOutputStream());
            ObjectInputStream entrada = new ObjectInputStream(socket.getInputStream());
            String login = "op1";
            String senha = "op1";
            System.out.println("Usuario conectado com Sucesso");
            saida.writeObject(login);
            saida.writeObject(senha);
            saida.writeObject("L");
            List<Produto> produtos = (List<Produto>) entrada.readObject();
            produtos.forEach((e) -> System.out.println(e.getNome()));
```

```
        } catch (IOException | ClassNotFoundException ex) {
            Logger.getLogger(CadastroClient.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```



Conclusão:

A. Como funcionam as classes Socket e ServerSocket?

As classes socket e serversocket são utilizadas para criar a conexão entre um cliente e um servidor. Sendo o Socket a extremidade do cliente e o ServerSocket a extremidade do servidor. Através desta conexão, cliente e servidor são capazes de enviar e receber dados um do outro.

B. Qual a importância das portas para a conexão com servidores?

As portas permitem ao servidor saber para qual aplicação direcionar o tráfego, cada aplicativo em execução no servidor tem sua porta associada que permite a comunicação com outros aplicativos remotos. Sendo algumas portas reservadas por padrão e outras necessitam abertura em Firewalls, servindo também como uma camada de segurança.

C. Para que servem as classes de entrada e saída ObjectInputStream e ObjectOutputStream, e por que os objetos transmitidos devem ser serializáveis?

ObjectOutputStream é a classe de java que permite que um objeto seja convertido em uma sequência de bytes, para posteriormente ser salvo em arquivo, um buffer de memória ou transmitido por um fluxo de rede. Já o InputStream pega esse fluxo de bytes e reconstrói o objeto para ser utilizado no programa. Os mesmo devem ser serializáveis para que possam ser convertidos em bytes e então sejam transmitidos pela rede.

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

Foi possível pois mesmo o cliente possuindo as entidade JPA, as solicitações ao banco de dados só poderão ser executadas pelo lado do servidor, em um ambiente real, mesmo existindo Querys nas entidades JPA elas não estariam direcionadas ao Banco de Dados que tem suas conexões geridas pelo provedor de persistência, no caso da aplicação desenvolvida, o EclipseLink.

2º Procedimento | Servidor Completo e Cliente Assíncrono

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.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

/**
 *
 * @author andre
 */
public class CadastroServer {

    /**
```

```
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
    EntityManagerFactory emf =
Persistence.createEntityManagerFactory("CadastroServerPU");
    ProdutoJpaController ctrl = new ProdutoJpaController(emf);
    UsuarioJpaController ctrlUsu = new UsuarioJpaController(emf);
    PessoaJpaController ctrlPessoa = new PessoaJpaController(emf);
    MovimentoJpaController ctrlMov = new MovimentoJpaController(emf);
        try {
            ServerSocket serverSocket= new ServerSocket(4321);
            while(true) {
                Socket cliente = serverSocket.accept();
                Thread usrConnect = new Thread(new
CadastroThreadV2(ctrl,ctrlUsu,ctrlMov,ctrlPessoa,cliente));
                usrConnect.start();
            }
        } catch (IOException ex) {
            Logger.getLogger(CadastroServer.class.getName()).log(Level.SEVERE, null, ex);
        }

    }

}
```

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 controller.exceptions.NonexistentEntityException;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.math.BigDecimal;
import java.net.Socket;
import java.time.DayOfWeek;
import java.time.LocalDateTime;
import java.time.Month;
import java.time.format.DateTimeFormatter;
import java.util.logging.Level;
import java.util.logging.Logger;
import model.Movimento;
import model.Produto;
import model.Usuario;

/**
 *
 * @author andre
 */
public class CadastroThreadV2 extends Thread {
    private ProdutoJpaController ctrl;
    private UsuarioJpaController ctrlUsu;
    private MovimentoJpaController ctrlMov;
    private PessoaJpaController ctrlPessoa;
    private Socket s1;
    private boolean state;

    public 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() {
        try {
            ObjectOutputStream saida = new ObjectOutputStream(s1.getOutputStream());
            ObjectInputStream entrada = new ObjectInputStream(s1.getInputStream());
            String login = (String) entrada.readObject();
            String senha = (String) entrada.readObject();
```

```
Usuario usuario = ctrlUsu.findUsuario(login, senha);
if (usuario == null) {
   saida.writeObject(null);
   s1.close();
   return;
} else {
   saida.writeObject(data());
   saida.writeObject("Usuario conectado com sucesso");
}

while(true) {
   String comando = (String) entrada.readObject();
   if("l".equals(comando.toLowerCase())) {
      saida.writeObject(data());
      saida.writeObject(ctrl.findProdutoEntities());
   } else if("e".equals(comando) || "E".equals(comando)) {
      Movimento mov = new Movimento();
      mov.setIdUsuario(usuario);
      mov.setTipo('E');
      int pessoaId = (int) entrada.readObject();
      mov.setIdPessoa(this.ctrlPessoa.findPessoa(pessoaId));
      int produtoId = (int) entrada.readObject();
      mov.setIdProduto(this.ctrl.findProduto(produtoId));
      int prodQnt = (int) entrada.readObject();
      mov.setQuantidade(prodQnt);
      BigDecimal prodPrc = (BigDecimal) entrada.readObject();
      mov.setValorUnitario(prodPrc);
      ctrlMov.create(mov);
      Produto prod = ctrl.findProduto(produtoId);
      prod.setQuantidade(prod.getQuantidade()+prodQnt);
      ctrl.edit(prod);
      saida.writeObject(data());
      saida.writeObject("Entrada efetuada com sucesso");
   } else if("s".equals(comando) || "s".equals(comando)) {
      Movimento mov = new Movimento();
      mov.setIdUsuario(usuario);
      mov.setTipo('S');
      int pessoaId = (int) entrada.readObject();
      mov.setIdPessoa(this.ctrlPessoa.findPessoa(pessoaId));
      int produtoId = (int) entrada.readObject();
      mov.setIdProduto(this.ctrl.findProduto(produtoId));
      int prodQnt = (int) entrada.readObject();
      mov.setQuantidade(prodQnt);
      BigDecimal prodPrc = (BigDecimal) entrada.readObject();
      mov.setValorUnitario(prodPrc);
```

```
                ctrlMov.create(mov);
                Produto prod = ctrl.findProduto(produtoId);
                prod.setQuantidade(prod.getQuantidade() - prodQnt);
                ctrl.edit(prod);
                saida.writeObject(data());
                saida.writeObject("Saida efetuada com sucesso");

            }
        }

    } catch (IOException | ClassNotFoundException ex) {
        Logger.getLogger(CadastroThreadV2.class.getName()).log(Level.SEVERE, null,
ex);
    } catch (NonexistentEntityException ex) {
        Logger.getLogger(CadastroThreadV2.class.getName()).log(Level.SEVERE, null,
ex);
    } catch (Exception ex) {
        Logger.getLogger(CadastroThreadV2.class.getName()).log(Level.SEVERE, null,
ex);
    }
}

private String data(){
   LocalDateTime now = LocalDateTime.now();
   DayOfWeek diaDaSemana = now.getDayOfWeek();
   Month mes = now.getMonth();
   DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd HH:mm:ss");
   String time = now.format(formatter);
   String data = ">> Nova comunicação em " + diaDaSemana.name() + " " + mes.name()
+ " " + time + " BRT " + now.getYear();
   return data;
}

}
```

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 cadastroclientv2;

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.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author andre
 */
public class CadastroClientV2 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
            try {
            Socket socket = new Socket("localhost", 4321);
            ObjectOutputStream saida = new ObjectOutputStream(socket.getOutputStream());
            ObjectInputStream entrada = new ObjectInputStream(socket.getInputStream());
```

```
        BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));
        String login = "op1";
        String senha = "op1";
        SaidaFrame frame = new SaidaFrame();
        frame.setVisible(true);
        frame.setBounds(1000, 800, 500, 180);
        frame.setDefaultCloseOperation(SaidaFrame.DISPOSE_ON_CLOSE);
        ThreadCliente thread = new ThreadCliente(entrada,frame.texto);
        thread.start();
        saida.writeObject(login);
        saida.writeObject(senha);
        while(true) {
           System.out.print("L - Listar | X - Finalizar | E - Entrada | S - Saida \n");
           String option = reader.readLine();
           switch(option.toLowerCase()) {
              case "e" -> {
                 saida.writeObject("e");
                 doMovement(reader,saida);
              }
              case "s" -> {
                 saida.writeObject("s");
                 doMovement(reader,saida);
              }

              case "l" -> {
                 saida.writeObject("l");
              }
              case "x" -> {
                 frame.dispose();
                 System.exit(0);
              }
              default -> {
                 System.out.println("Opcao nao existe");}
           }
        }
     } catch (IOException ex) {
        Logger.getLogger(CadastroClientV2.class.getName()).log(Level.SEVERE, null,
ex);
     }
  }

   private static void doMovement(BufferedReader reader, ObjectOutputStream saida)
throws IOException {
     System.out.println("Id da Pessoa:");
```

```
        saida.writeObject(Integer.valueOf(reader.readLine()));
        System.out.println("Id do Produto");
        saida.writeObject(Integer.valueOf(reader.readLine()));
        System.out.println("Quantidade:");
        saida.writeObject(Integer.valueOf(reader.readLine()));
        System.out.println("Valor Unitario:");
        saida.writeObject(BigDecimal.valueOf(Float.parseFloat(reader.readLine())));
    }
}
```

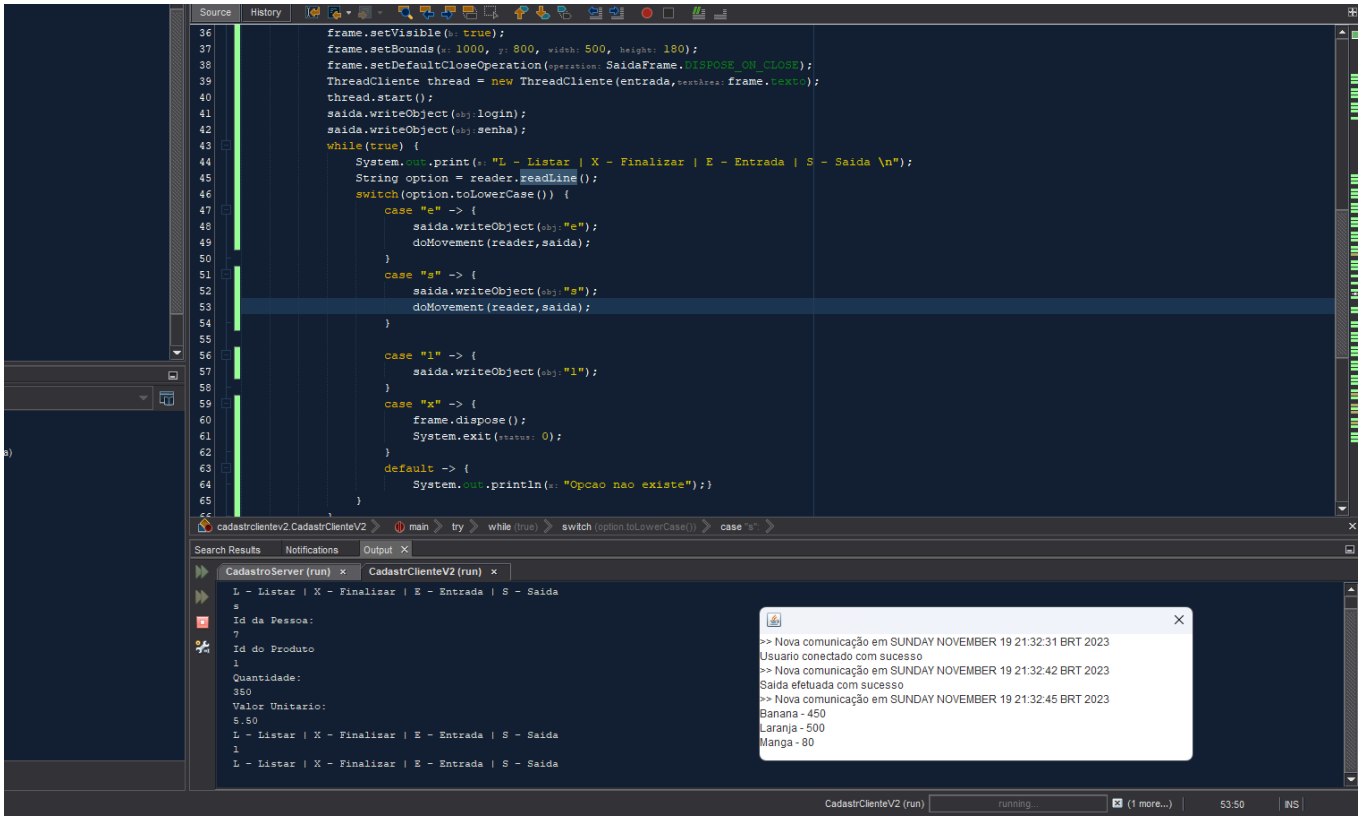
SaidaFrame.java

```
/*
 * 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 cadastroclientv2;
import javax.swing.JDialog;
import javax.swing.JTextArea;
/**
 *
 * @author andre
 */
class SaidaFrame extends JDialog {
    public JTextArea texto;

    public SaidaFrame() {
        texto = new JTextArea();
        setBounds(100, 100, 400, 400);
        setModal(false);
        add(texto);
    }
}
```

ThreadCliente.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 cadastroclientv2;

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

/**
 *
 * @author andre
 */
public class ThreadCliente extends Thread {
    private ObjectInputStream entrada;
    private JTextArea textArea;

    public ThreadCliente(ObjectInputStream entrada, JTextArea textArea) {
    this.entrada = entrada;
    this.textArea = textArea;
    }

    @Override
    public void run() {
        while(true) {
        try {
            Object objeto = entrada.readObject();
            if (objeto instanceof String) {
            SwingUtilities.invokeLater(() -> textArea.append((String) objeto + "\n"));
            } else if (objeto instanceof List) {
                List<Produto> lista = (List<Produto>) objeto;
                lista.forEach((e) -> {
                    SwingUtilities.invokeLater(() -> textArea.append(e.getNome() + " - " + e.getQuantidade() + "\n"));
                });
```

```
            }
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();

        }
        }
    }
}
```



Conclusão:

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

As Threads podem ser utilizadas para tratar as respostas do servidor, pois assim desocupam o processo principal que ficaria bloqueado até que a resposta fosse recebida e processada.

B. Para que serve o método invokeLater, da classe SwingUtilities?

Ele serve para agendar a execução do código que manipula os componentes do swing, garantindo que a resposta de uma thread seja executada no tempo correto, evitando assim problemas de concorrência.

C. Como os objetos são enviados e recebidos pelo Socket Java?

Eles são enviados serializados, e transmitidos na forma de um fluxo de bytes, sendo gerenciados pelas classes ObjectOutputStream e ObjectInputStream.

D. Compare a utilização de comportamento assíncrono ou síncrono nos clientes com Socket Java, ressaltando as características relacionadas ao bloqueio do processamento.

Em um cliente síncrono, sempre haverá o bloqueio de E/S pois a Thread principal, só pode continuar após a conclusão da operação de E/S. Já no caso assíncrono as operações de E/S são executadas por Threads secundárias, permitindo que a Thread principal continue executando outras tarefas. Este método é ideal quando se necessita lidar com múltiplas operações simultaneamente.