1. Configure o `pom.xml`: Navegue até o diretório do projeto e abra o arquivo `pom.xml`.

O `pom.xml` será bem básico:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-
4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>
   <groupId>com.exemplo</groupId>
   <artifactId>JavaRPC-Sockets</artifactId>
   <version>1.0-SNAPSHOT</version>
</project>
```

2. Criação do Código

Interface RPC
No diretório `src/main/java/com/exemplo`, crie a interface `HelloService.java`:

```
package com.exemplo;

// Interface do serviço
public interface HelloService {
   String sayHello(String name);
}
```

Implementação do Serviço

Crie a classe `HelloServiceImpl.java` que implementa a interface:

```
package com.exemplo;

// Implementação do serviço
public class HelloServiceImpl implements HelloService {
   @Override
   public String sayHello(String name) {
      return "Olá, " + name + "! Este é um exemplo de RPC com sockets.";
   }
}
```

Servidor

Crie a classe `Server.java`, responsável por aceitar conexões e processar as chamadas RPC:

```
package com.exemplo;

import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;

public class Server {
   public static void main(String[] args) {
```

```
        try (ServerSocket serverSocket = new ServerSocket(5000)) {
            System.out.println("Servidor pronto na porta 5000...");

            // Espera por conexões
            while (true) {
                Socket clientSocket = serverSocket.accept();
                System.out.println("Cliente conectado!");

                try (BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
                     PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true)) {

                    // Recebe a requisição
                    String input = in.readLine();
                    System.out.println("Requisição recebida: " + input);

                    // Processa a chamada ao método
                    if (input.startsWith("sayHello:")) {
                        String name = input.split(":")[1];
                        HelloService service = new HelloServiceImpl();
                        String response = service.sayHello(name);

                        // Envia a resposta de volta ao cliente
                        out.println(response);
                    } else {
                        out.println("Método não suportado.");
                    }
                }
                clientSocket.close();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Cliente

Crie a classe `Client.java`, que envia as chamadas RPC para o servidor:

```
package com.exemplo;

import java.io.*;
import java.net.Socket;

public class Client {
    public static void main(String[] args) {
        try (Socket socket = new Socket("localhost", 5000);
             BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
             PrintWriter out = new PrintWriter(socket.getOutputStream(), true)) {
```

```
            // Envia a chamada do método para o servidor
            String request = "sayHello:Usuário";
            out.println(request);
            System.out.println("Requisição enviada: " + request);

            // Recebe a resposta do servidor
            String response = in.readLine();
            System.out.println("Resposta do servidor: " + response);

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