a. TimeoutServer (SO\_TIMEOUT)

import java.io.IOException;

import java.net.ServerSocket;

import java.net.Socket;

import java.net.SocketTimeoutException;

public class TimeoutServer {

public static void main(String[] args) {

int port = 12345;

try (ServerSocket serverSocket = new ServerSocket(port)) {

serverSocket.setSoTimeout(5000); // Set timeout to 5000 milliseconds (5 seconds)

System.out.println("TimeoutServer is listening on port " + port);

while (true) {

try {

Socket socket = serverSocket.accept(); // This will block for up to 5 seconds

System.out.println("New client connected to TimeoutServer");

// Handle the client connection

} catch (SocketTimeoutException e) {

System.out.println("Accept timed out, no incoming connection in 5 seconds");

}

}

} catch (IOException ex) {

ex.printStackTrace();

}

}

}

b. ReuseAddressServer (SO\_REUSEADDR)

import java.io.IOException;

import java.net.ServerSocket;

public class ReuseAddressServer {

public static void main(String[] args) {

int port = 12345;

try (ServerSocket serverSocket = new ServerSocket(port)) {

serverSocket.setReuseAddress(true); // Enable SO\_REUSEADDR

System.out.println("ReuseAddressServer is listening on port " + port);

while (true) {

try {

Socket socket = serverSocket.accept();

System.out.println("New client connected to ReuseAddressServer");

// Handle the client connection

} catch (IOException ex) {

ex.printStackTrace();

}

}

} catch (IOException ex) {

ex.printStackTrace();

}

}

}

c. ReceiveBufferServer (SO\_RCVBUF)

import java.io.IOException;

import java.net.ServerSocket;

public class ReceiveBufferServer {

public static void main(String[] args) {

int port = 12345;

try (ServerSocket serverSocket = new ServerSocket(port)) {

serverSocket.setReceiveBufferSize(65536); // Set receive buffer size to 64 KB

System.out.println("ReceiveBufferServer is listening on port " + port);

while (true) {

try {

Socket socket = serverSocket.accept();

System.out.println("New client connected to ReceiveBufferServer with receive buffer size: " +

socket.getReceiveBufferSize());

// Handle the client connection

} catch (IOException ex) {

ex.printStackTrace();

}

}

} catch (IOException ex) {

ex.printStackTrace();

}

}

}

d. SimpleClient

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.Socket;

public class SimpleClient {

public static void main(String[] args) {

String hostname = "localhost";

int port = 12345;

try (Socket socket = new Socket(hostname, port)) {

System.out.println("Connected to the server");

BufferedReader input = new BufferedReader(new InputStreamReader(socket.getInputStream()));

String serverResponse = input.readLine();

System.out.println("Server response: " + serverResponse);

} catch (IOException ex) {

System.out.println("Client exception: " + ex.getMessage());

ex.printStackTrace();

}

}

}

**HTTP server**

1. SingleFileServer.java

import java.io.\*;

import java.net.ServerSocket;

import java.net.Socket;

import java.net.URL;

import java.nio.charset.StandardCharsets;

public class SingleFileServer {

public static void main(String[] args) {

int port = 8080;

String fileUrl = "https://raw.githubusercontent.com/Sharatmaharjan/Np/main/code/index.html"; // Raw file URL from GitHub

try {

// Fetch the HTML content from the URL

String content = fetchContentFromUrl(fileUrl);

try (ServerSocket serverSocket = new ServerSocket(port)) {

System.out.println("SingleFileServer is listening on port " + port);

while (true) {

try (Socket socket = serverSocket.accept()) {

System.out.println("New client connected");

BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

// Read the HTTP request (only the first line for simplicity)

String requestLine = in.readLine();

System.out.println("Request: " + requestLine);

if (requestLine != null && requestLine.startsWith("GET")) {

// Send HTTP response

out.println("HTTP/1.1 200 OK");

out.println("Content-Type: text/html");

out.println("Content-Length: " + content.length());

out.println();

out.println(content);

}

socket.close();

}

}

}

} catch (IOException e) {

e.printStackTrace();

}

}

private static String fetchContentFromUrl(String fileUrl) throws IOException {

StringBuilder content = new StringBuilder();

URL url = new URL(fileUrl);

try (BufferedReader in = new BufferedReader(new InputStreamReader(url.openStream(), StandardCharsets.UTF\_8))) {

String inputLine;

while ((inputLine = in.readLine()) != null) {

content.append(inputLine).append("\n");

}

}

return content.toString();

}

}

2. Redirector.java

import java.io.\*;

import java.net.ServerSocket;

import java.net.Socket;

public class Redirector {

public static void main(String[] args) {

int port = 8080;

String redirectUrl = "http://example.com";

try (ServerSocket serverSocket = new ServerSocket(port)) {

System.out.println("Redirector is listening on port " + port);

while (true) {

try (Socket socket = serverSocket.accept()) {

System.out.println("New client connected");

BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

// Read the HTTP request (only the first line for simplicity)

String requestLine = in.readLine();

System.out.println("Request: " + requestLine);

if (requestLine != null && requestLine.startsWith("GET")) {

// Send HTTP redirect response

out.println("HTTP/1.1 302 Found");

out.println("Location: " + redirectUrl);

out.println();

}

socket.close();

}

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

3. FullFledgedHttpServer.java

import com.sun.net.httpserver.HttpServer;

import com.sun.net.httpserver.HttpHandler;

import com.sun.net.httpserver.HttpExchange;

import java.io.IOException;

import java.io.OutputStream;

import java.net.InetSocketAddress;

public class FullFledgedHttpServer {

public static void main(String[] args) throws IOException {

int port = 8080;

HttpServer server = HttpServer.create(new InetSocketAddress(port), 0);

server.createContext("/", new RootHandler());

server.createContext("/hello", new HelloHandler());

server.setExecutor(null); // creates a default executor

System.out.println("FullFledgedHttpServer is listening on port " + port);

server.start();

}

static class RootHandler implements HttpHandler {

@Override

public void handle(HttpExchange exchange) throws IOException {

String response = "Welcome to the Full-Fledged HTTP Server!";

exchange.sendResponseHeaders(200, response.length());

OutputStream os = exchange.getResponseBody();

os.write(response.getBytes());

os.close();

}

}

static class HelloHandler implements HttpHandler {

@Override

public void handle(HttpExchange exchange) throws IOException {

String response = "Hello, world!";

exchange.sendResponseHeaders(200, response.length());

OutputStream os = exchange.getResponseBody();

os.write(response.getBytes());

os.close();

}

}

}