Buổi 5

VD1

import java.io.IOException;

import java.net.InetSocketAddress;

import java.nio.ByteBuffer;

import java.nio.channels.ServerSocketChannel;

import java.nio.channels.SocketChannel;

public class VD1Server {

    public static void main(String[] args) {

        try {

            ServerSocketChannel serverChannel = ServerSocketChannel.open();

            serverChannel.bind(new InetSocketAddress("localhost",12345));

            while (true) {

                SocketChannel clientChannel=serverChannel.accept();

                ByteBuffer buffer = ByteBuffer.allocate(1024);

                clientChannel.read(buffer);

                buffer.flip();

                String message = new String(buffer.array(),0,buffer.limit());

                System.out.println("Received message from client: " + message);

                String response = "Hello from server";

                buffer.clear();

                buffer.put(response.getBytes());

                buffer.flip();

                clientChannel.write(buffer);

                clientChannel.close();

            }

        } catch (IOException e)

        {

            e.printStackTrace();

        }

    }

}

import java.io.IOException;

import java.net.InetSocketAddress;

import java.nio.ByteBuffer;

import java.nio.channels.SocketChannel;

public class VD1Client {

    public static void main (String [] args) {

        try {

            int x =1;

            while (true) {

                SocketChannel clientChannel= SocketChannel.open();

                clientChannel.connect(new InetSocketAddress("localhost",12345));

                String message = "Hello from client" + x++;

                ByteBuffer buffer = ByteBuffer.wrap(message.getBytes());

                clientChannel.write(buffer);

                ByteBuffer responseBuffer = ByteBuffer.allocate(1024);

                clientChannel.read(responseBuffer);

                responseBuffer.flip();

                String response = new String(responseBuffer.array(),0,responseBuffer.limit());

                System.out.println("Received response from server: " + response);

                clientChannel.close();

            }

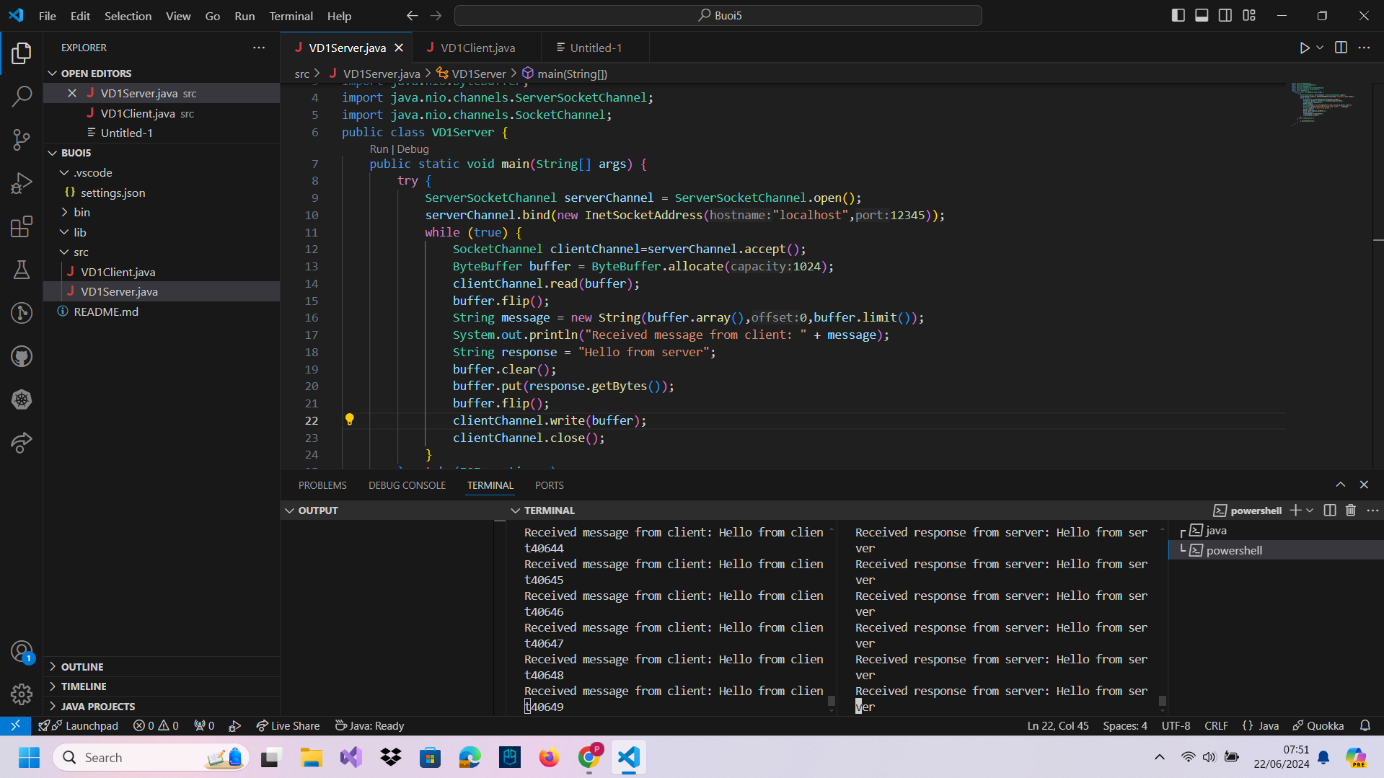
        } catch(IOException e){

            e.printStackTrace();

        }

    }

}



VD2

import java.io.IOException;

import java.net.InetSocketAddress;

import java.nio.ByteBuffer;

import java.nio.channels.\*;

import java.util.Iterator;

import java.util.Set;

public class VD2Server {

    public static void main (String [] args){

        try {

            Selector selector = Selector.open();

            ServerSocketChannel serverSocketChannel = ServerSocketChannel.open();

            serverSocketChannel.bind(new InetSocketAddress("localhost",12345));

            serverSocketChannel.configureBlocking(false);

            serverSocketChannel.register(selector, SelectionKey.OP\_ACCEPT);

            System.out.println("Server started. wating for connections....");

            while ( true) {

                int readyCount = selector.select();

                if( readyCount==0){

                    continue;

                }

                Set<SelectionKey> selectedKeys = selector.selectedKeys();

                Iterator<SelectionKey> keyIterator = selectedKeys.iterator();

                while (keyIterator.hasNext()) {

                    SelectionKey key = keyIterator.next();

                    if (key.isAcceptable()){

                        ServerSocketChannel serverChannel = (ServerSocketChannel) key.channel();

                        SocketChannel clientChannel = serverChannel.accept();

                        clientChannel.configureBlocking(false);

                        clientChannel.register(selector, SelectionKey.OP\_READ);

                        System.out.println("Client connected: " + clientChannel.getRemoteAddress());

                    } else if (key.isReadable()){

                        SocketChannel clientChannel = (SocketChannel) key.channel();

                        ByteBuffer buffer = ByteBuffer.allocate(1024);

                        int bytesRead = clientChannel.read(buffer);

                        try{

                            if(bytesRead == -1){

                                System.out.println("Client disconnected: " + clientChannel.getRemoteAddress());

                                clientChannel.close();

                            } else if ( bytesRead > 0) {

                                buffer.flip();

                                byte [] data = new byte[buffer.limit()];

                                buffer.get(data);

                                String message = new String (data);

                                System.out.println("Received message from client: " + message);

                                String response = "Hello from server";

                                ByteBuffer responseBuffer = ByteBuffer.wrap(response.getBytes());

                                clientChannel.write(responseBuffer);

                            }

                        } catch (Exception e){

                            e.printStackTrace();

                            continue;

                        }

                    }

                    keyIterator.remove();

            }

        }

    }catch (IOException e)

    {

        e.printStackTrace();

    }

}

}

import java.io.IOException;

import java.net.InetSocketAddress;

import java.nio.ByteBuffer;

import java.nio.channels.SocketChannel;

public class VD2Client {

    public static void main (String [] args){

        try{

            SocketChannel clientChannel = SocketChannel.open();

            clientChannel.connect(new InetSocketAddress("localhost",12345));

            System.out.println("Connected to server: " + clientChannel.getRemoteAddress());

            String message = "Hello from client";

            ByteBuffer buffer = ByteBuffer.wrap(message.getBytes());

            clientChannel.write(buffer);

            ByteBuffer responseBuffer = ByteBuffer.allocate(1024);

            clientChannel.read(responseBuffer);

            responseBuffer.flip();

            byte [] responseData = new byte[responseBuffer.limit()];

            responseBuffer.get(responseData);

            String response = new String (responseData);

            System.out.println("Received response from server " + response);

            clientChannel.close();

        } catch (IOException e)

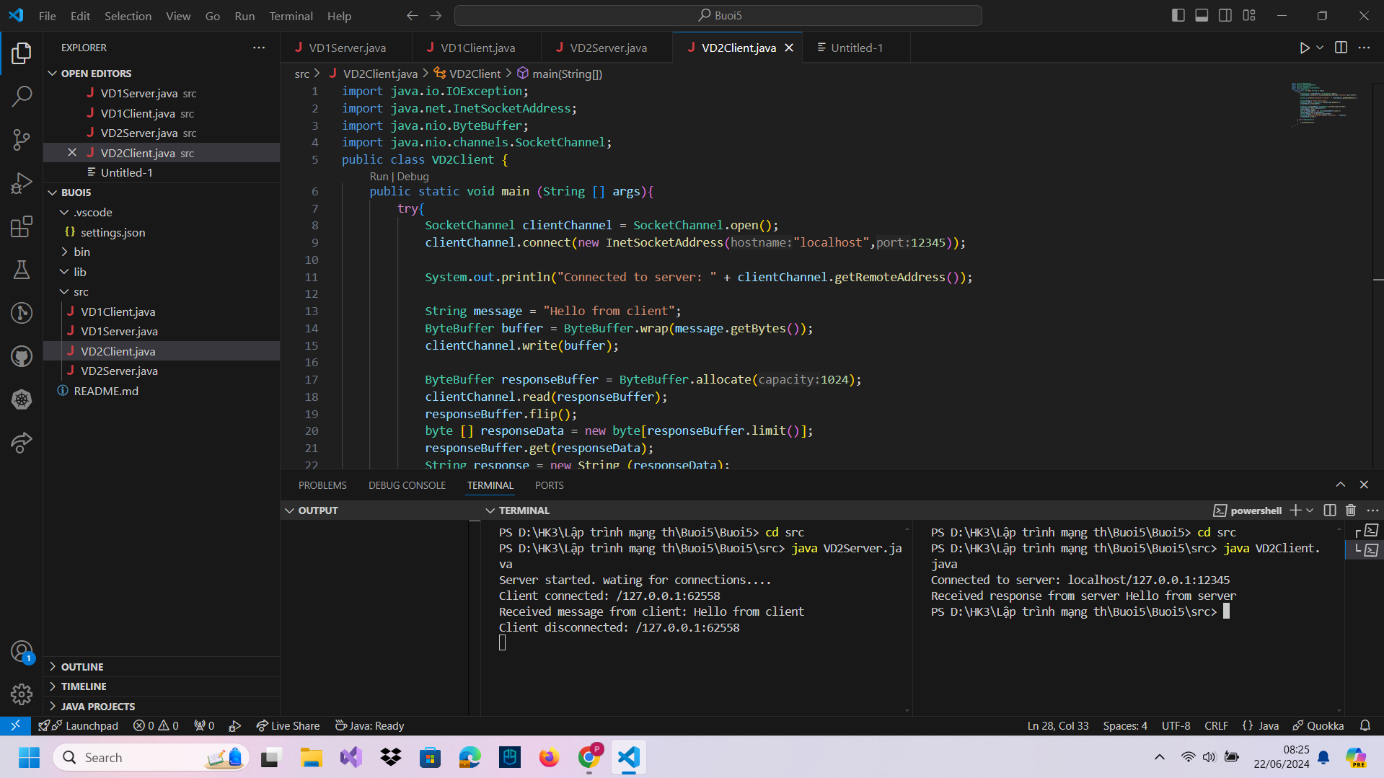
        {

            e.printStackTrace();

        }

    }

}



VD3

import java.io.IOException;

import java.net.InetSocketAddress;

import java.nio.ByteBuffer;

import java.nio.channels.AsynchronousServerSocketChannel;

import java.nio.channels.AsynchronousSocketChannel;

import java.nio.channels.CompletionHandler;

public class VD3Server {

    public static void main (String [] args){

        try{

            AsynchronousServerSocketChannel serverChannel =AsynchronousServerSocketChannel.open();

            serverChannel.bind(new InetSocketAddress("localhost",12345));

            serverChannel.accept(null, new CompletionHandler<AsynchronousSocketChannel, Void> (){

                @Override

                public void completed (AsynchronousSocketChannel clientChannel, Void attachment){

                    serverChannel.accept(null,this);

                    ByteBuffer buffer = ByteBuffer.allocate(1024);

                    clientChannel.read(buffer, buffer, new CompletionHandler<Integer, ByteBuffer>() {

                        @Override

                        public void completed(Integer bytesRead, ByteBuffer buffer){

                            if(bytesRead>0){

                                buffer.flip();

                                byte [] data = new byte[bytesRead];

                                buffer.get(data);

                                String message = new String(data);

                                System.out.println("Received message from client: " + message);

                                String response = "Hello from server" + message;

                                byte [] responseData = response.getBytes();

                                ByteBuffer responseBuffer = ByteBuffer.wrap(responseData);

                                clientChannel.write(responseBuffer);

                            }

                        }

                        @Override

                        public void failed (Throwable exc, ByteBuffer buffer){

                            exc.printStackTrace();

                        }

                    });

                }

                @Override

                public void failed (Throwable exc, Void attachment){

                    exc.printStackTrace();

                }

            });

            Thread.currentThread().join();

        }catch (IOException | InterruptedException e){

            e.printStackTrace();

        }

    }

}

import java.io.IOException;

import java.net.InetSocketAddress;

import java.nio.ByteBuffer;

import java.nio.channels.AsynchronousSocketChannel;

import java.nio.channels.CompletionHandler;

public class VD3Client {

    public static int x = 1;

    public static void main(String[] args) {

        try {

            // Tạo AsynchronousSocketChannel

            while (x <= 10) {

                AsynchronousSocketChannel clientChannel = AsynchronousSocketChannel.open();

                // Kết nối tới máy chủ

                clientChannel.connect(new InetSocketAddress("localhost", 12345), null, new CompletionHandler<Void, Void>() {

                    @Override

                    public void completed(Void result, Void attachment) {

                        // Gửi tin nhắn tới máy chủ

                        String message = "Hello from client" + x++;

                        ByteBuffer buffer = ByteBuffer.wrap(message.getBytes());

                        clientChannel.write(buffer, null, new CompletionHandler<Integer, Void>() {

                            @Override

                            public void completed(Integer bytesWritten, Void attachment) {

                                // Đọc phản hồi từ máy chủ

                                ByteBuffer responseBuffer = ByteBuffer.allocate(1024);

                                clientChannel.read(responseBuffer, null, new CompletionHandler<Integer, Void>() {

                                    @Override

                                    public void completed(Integer bytesRead, Void attachment) {

                                        if (bytesRead > 0) {

                                            responseBuffer.flip();

                                            byte[] responseData = new byte[bytesRead];

                                            responseBuffer.get(responseData);

                                            String response = new String(responseData);

                                            System.out.println("Received response from server: " + response);

                                        }

                                    }

                                    @Override

                                    public void failed(Throwable exc, Void attachment) {

                                        // Xử lý lỗi khi đọc dữ liệu

                                        exc.printStackTrace();

                                    }

                                });

                            }

                            @Override

                            public void failed(Throwable exc, Void attachment) {

                                // Xử lý lỗi khi ghi dữ liệu

                                exc.printStackTrace();

                            }

                        });

                    }

                    @Override

                    public void failed(Throwable exc, Void attachment) {

                        // Xử lý lỗi khi kết nối

                        exc.printStackTrace();

                    }

                });

            }

            // Chờ cho kết nối và gửi/nhận dữ liệu hoàn tất

            Thread.currentThread().join();

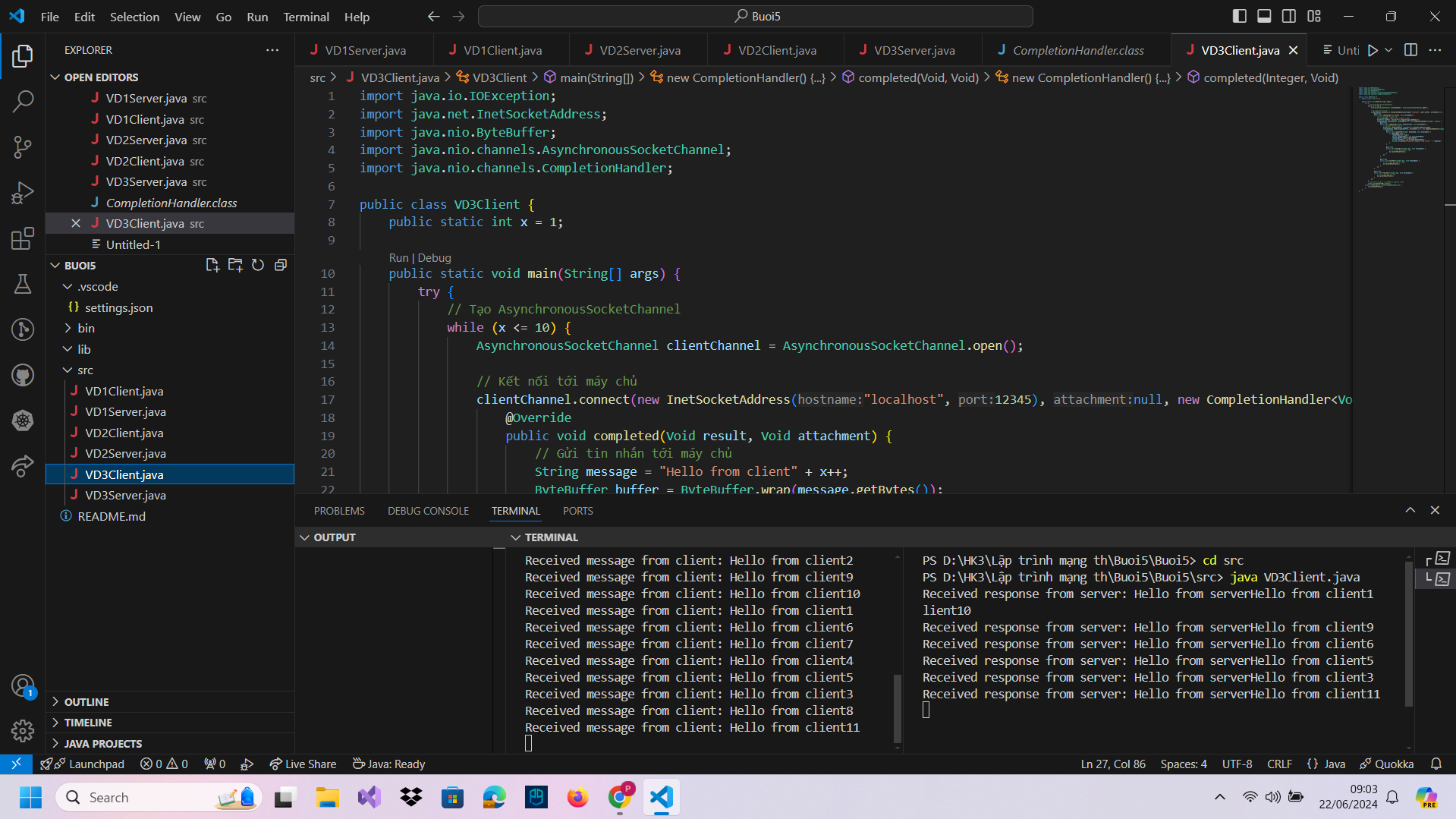
        } catch (IOException | InterruptedException e) {

            e.printStackTrace();

        }

    }

}



Bài1

Server

import java.io.IOException;

import java.net.InetSocketAddress;

import java.nio.ByteBuffer;

import java.nio.channels.SelectionKey;

import java.nio.channels.Selector;

import java.nio.channels.ServerSocketChannel;

import java.nio.channels.SocketChannel;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.Iterator;

import java.util.Set;

public class Bai1Server {

    public static void main(String[] args) {

        try {

            // Tạo ServerSocketChannel

            ServerSocketChannel serverChannel = ServerSocketChannel.open();

            serverChannel.socket().bind(new InetSocketAddress(5000));

            serverChannel.configureBlocking(false);

            // Tạo Selector

            Selector selector = Selector.open();

            serverChannel.register(selector, SelectionKey.OP\_ACCEPT);

            System.out.println("Server started...");

            while (true) {

                selector.select();

                Set<SelectionKey> selectedKeys = selector.selectedKeys();

                Iterator<SelectionKey> it = selectedKeys.iterator();

                while (it.hasNext()) {

                    SelectionKey key = it.next();

                    if (key.isAcceptable()) {

                        SocketChannel clientChannel = serverChannel.accept();

                        clientChannel.configureBlocking(false);

                        clientChannel.register(selector, SelectionKey.OP\_WRITE);

                        System.out.println("Client connected...");

                    } else if (key.isWritable()) {

                        SocketChannel clientChannel = (SocketChannel) key.channel();

                        String currentTime = new SimpleDateFormat("HH:mm:ss").format(new Date());

                        ByteBuffer buffer = ByteBuffer.allocate(64);

                        buffer.put(currentTime.getBytes());

                        buffer.flip();

                        clientChannel.write(buffer);

                        Thread.sleep(1000);

                    }

                    it.remove();

                }

            }

        } catch (IOException | InterruptedException e) {

            e.printStackTrace();

        }

    }

}

Client

import java.io.IOException;

import java.net.InetSocketAddress;

import java.nio.ByteBuffer;

import java.nio.channels.SelectionKey;

import java.nio.channels.Selector;

import java.nio.channels.SocketChannel;

import java.util.Iterator;

import java.util.Set;

public class Bai1Client {

    public static void main(String[] args) {

        try {

            // Tạo SocketChannel

            SocketChannel clientChannel = SocketChannel.open(new InetSocketAddress("localhost", 5000));

            clientChannel.configureBlocking(false);

            // Tạo Selector

            Selector selector = Selector.open();

            clientChannel.register(selector, SelectionKey.OP\_READ);

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

            while (true) {

                selector.select();

                Set<SelectionKey> selectedKeys = selector.selectedKeys();

                Iterator<SelectionKey> it = selectedKeys.iterator();

                while (it.hasNext()) {

                    SelectionKey key = it.next();

                    if (key.isReadable()) {

                        ByteBuffer buffer = ByteBuffer.allocate(64);

                        SocketChannel channel = (SocketChannel) key.channel();

                        channel.read(buffer);

                        buffer.flip();

                        byte[] bytes = new byte[buffer.remaining()];

                        buffer.get(bytes);

                        String currentTime = new String(bytes);

                        System.out.println("Current time: " + currentTime);

                    }

                    it.remove();

                }

            }

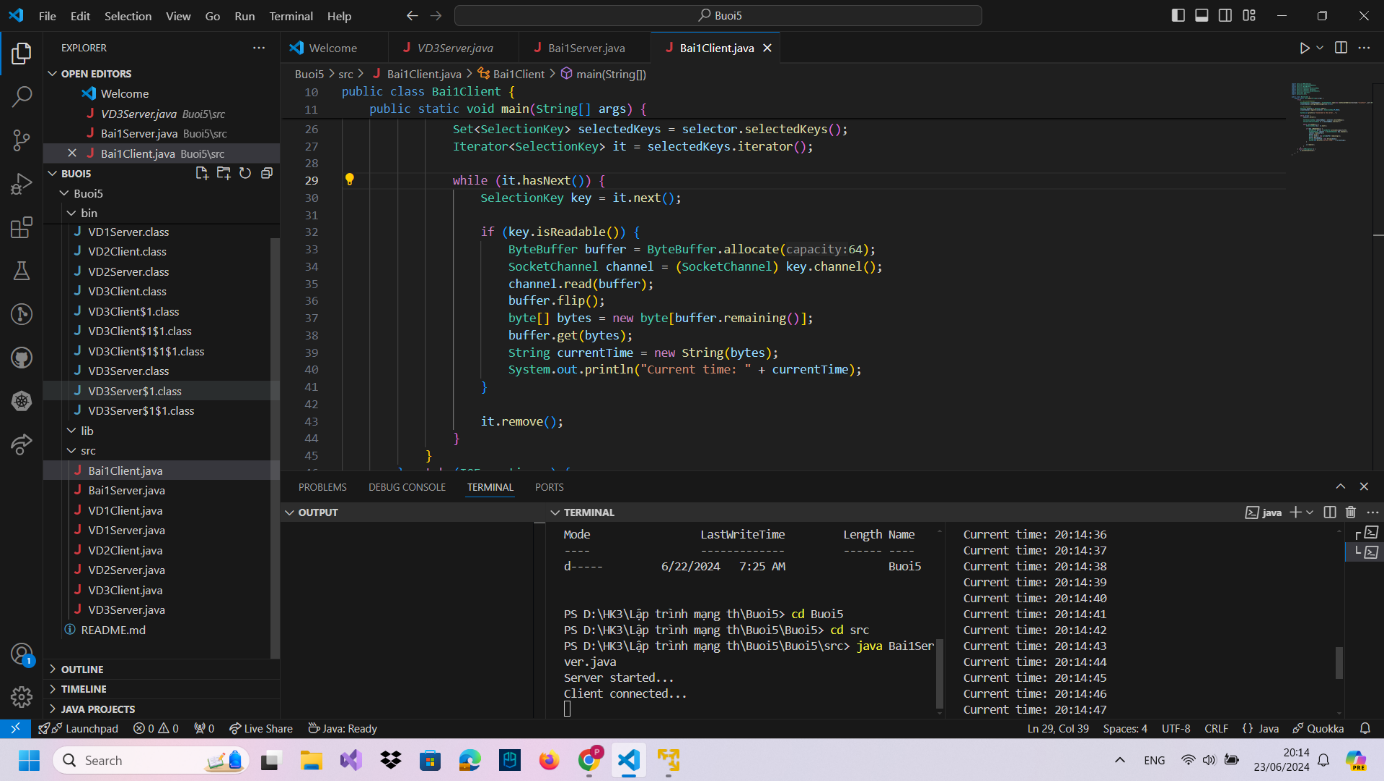
        } catch (IOException e) {

            e.printStackTrace();

        }

    }

}



Bài 2

Server

import java.io.IOException;

import java.net.InetSocketAddress;

import java.nio.ByteBuffer;

import java.nio.channels.SelectionKey;

import java.nio.channels.Selector;

import java.nio.channels.ServerSocketChannel;

import java.nio.channels.SocketChannel;

import java.util.Iterator;

import java.util.Set;

public class Bai2Server {

    public static void main(String[] args) {

        try {

            // Tạo ServerSocketChannel và cài đặt cổng

            ServerSocketChannel serverChannel = ServerSocketChannel.open();

            serverChannel.socket().bind(new InetSocketAddress(5000));

            serverChannel.configureBlocking(false);

            // Tạo Selector và đăng ký ServerSocketChannel để nhận yêu cầu kết nối từ client

            Selector selector = Selector.open();

            serverChannel.register(selector, SelectionKey.OP\_ACCEPT);

            System.out.println("Server started...");

            while (true) {

                // Chờ đợi sự kiện xảy ra trên các channel đã đăng ký với Selector

                selector.select();

                // Lấy tập hợp các key đã chọn

                Set<SelectionKey> selectedKeys = selector.selectedKeys();

                Iterator<SelectionKey> it = selectedKeys.iterator();

                while (it.hasNext()) {

                    SelectionKey key = it.next();

                    if (key.isAcceptable()) {

                        // Nếu có yêu cầu kết nối mới từ client

                        handleAcceptableEvent(serverChannel, selector);

                    } else if (key.isReadable()) {

                        // Nếu có dữ liệu đến từ client

                        handleReadableEvent(key);

                    }

                    // Xóa key hiện tại để tránh xử lý lại

                    it.remove();

                }

            }

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

    // Phương thức xử lý yêu cầu kết nối từ client

    private static void handleAcceptableEvent(ServerSocketChannel serverChannel, Selector selector) throws IOException {

        SocketChannel clientChannel = serverChannel.accept();

        clientChannel.configureBlocking(false);

        clientChannel.register(selector, SelectionKey.OP\_READ);

        System.out.println("Client connected...");

    }

    // Phương thức xử lý khi có dữ liệu đến từ client

    private static void handleReadableEvent(SelectionKey key) throws IOException {

        SocketChannel clientChannel = (SocketChannel) key.channel();

        ByteBuffer buffer = ByteBuffer.allocate(64);

        int bytesRead = clientChannel.read(buffer);

        if (bytesRead == -1) {

            // Nếu client đã đóng kết nối

            clientChannel.close();

            return;

        }

        buffer.flip();

        int number = Integer.parseInt(new String(buffer.array()).trim());

        String result = "";

        // Xử lý logic tại đây để kiểm tra số hoàn thiện

        if (isPerfectNumber(number)) {

            result = number + " là số hoàn thiện.";

        } else {

            int nextPerfect = nextPerfectNumber(number);

            result = number + " không phải là số hoàn thiện. Số hoàn thiện lớn hơn và gần nhất là " + nextPerfect + ".";

        }

        // Chuẩn bị phản hồi để gửi lại cho client

        ByteBuffer responseBuffer = ByteBuffer.allocate(result.getBytes().length);

        responseBuffer.put(result.getBytes());

        responseBuffer.flip();

        clientChannel.write(responseBuffer);

        // Đóng kết nối với client sau khi gửi phản hồi

        clientChannel.close();

    }

    // Phương thức kiểm tra xem một số có phải là số hoàn thiện hay không

    private static boolean isPerfectNumber(int number) {

        int sum = 1;

        for (int i = 2; i <= number / 2; i++) {

            if (number % i == 0) {

                sum += i;

            }

        }

        return sum == number;

    }

    // Phương thức tìm số hoàn thiện tiếp theo lớn hơn số đã cho

    private static int nextPerfectNumber(int number) {

        int next = number + 1;

        while (!isPerfectNumber(next)) {

            next++;

        }

        return next;

    }

}

Client

import java.io.IOException;

import java.net.InetSocketAddress;

import java.nio.ByteBuffer;

import java.nio.channels.SelectionKey;

import java.nio.channels.Selector;

import java.nio.channels.SocketChannel;

import java.util.Iterator;

import java.util.Scanner;

import java.util.Set;

public class Bai2Client {

    public static void main(String[] args) {

        try {

            // Tạo kênh SocketChannel và kết nối tới server địa phương trên cổng 5000

            SocketChannel clientChannel = SocketChannel.open(new InetSocketAddress("localhost", 5000));

            clientChannel.configureBlocking(false);

            // Tạo Selector và đăng ký kênh để ghi dữ liệu lên server

            Selector selector = Selector.open();

            clientChannel.register(selector, SelectionKey.OP\_WRITE);

            // Nhập số nguyên dương từ người dùng

            Scanner scanner = new Scanner(System.in);

            System.out.print("Nhập số nguyên dương: ");

            int number = scanner.nextInt();

            while (true) {

                // Chờ đợi sự kiện xảy ra trên các channel đã đăng ký với Selector

                selector.select();

                // Lấy tập hợp các key đã chọn

                Set<SelectionKey> selectedKeys = selector.selectedKeys();

                Iterator<SelectionKey> it = selectedKeys.iterator();

                while (it.hasNext()) {

                    SelectionKey key = it.next();

                    if (key.isWritable()) {

                        // Ghi số nguyên dương vào server

                        handleWritableEvent(clientChannel, number);

                        clientChannel.register(selector, SelectionKey.OP\_READ);

                    } else if (key.isReadable()) {

                        // Đọc phản hồi từ server

                        handleReadableEvent(clientChannel);

                        clientChannel.close();

                        return;

                    }

                    // Xóa key hiện tại để tránh xử lý lại

                    it.remove();

                }

            }

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

    // Phương thức xử lý khi có thể ghi dữ liệu lên server

    private static void handleWritableEvent(SocketChannel clientChannel, int number) throws IOException {

        ByteBuffer buffer = ByteBuffer.allocate(64);

        buffer.put(String.valueOf(number).getBytes());

        buffer.flip();

        clientChannel.write(buffer);

    }

    // Phương thức xử lý khi có thể đọc phản hồi từ server

    private static void handleReadableEvent(SocketChannel clientChannel) throws IOException {

        ByteBuffer buffer = ByteBuffer.allocate(256);

        clientChannel.read(buffer);

        buffer.flip();

        byte[] bytes = new byte[buffer.remaining()];

        buffer.get(bytes);

        String response = new String(bytes);

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

    }

}

A screenshot of a computer

Description automatically generated