



java apache netty이용해서 tcp 서버를 구현해줘.

좋아요, 바로 달립니다. Netty로 "줄바꿈(\n) 기준 텍스트"를 주고받는 TCP 에코 서버 예제를 드릴게요. Maven 프로젝트로 바로 빌드/실행 가능합니다. 필요하면 길이기반(Length-Field) 프로토콜 스니펫도 넣었습니다.

1) pom.xml

```
xml

<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>dev.example</groupId>
  <artifactId>netty-tcp-server</artifactId>
  <version>1.0.0</version>
  <properties>
    <maven.compiler.source>17</maven.compiler.source>
    <maven.compiler.target>17</maven.compiler.target>
    <netty.version>4.1.109.Final</netty.version>
  </properties>

  <dependencies>
    <!-- Netty core / transport / codec / handler -->
    <dependency>
      <groupId>io.netty</groupId>
      <artifactId>netty-buffer</artifactId>
      <version>${netty.version}</version>
    </dependency>
    <dependency>
      <groupId>io.netty</groupId>
      <artifactId>netty-transport</artifactId>
      <version>${netty.version}</version>
    </dependency>
    <dependency>
      <groupId>io.netty</groupId>
      <artifactId>netty-codec</artifactId>
      <version>${netty.version}</version>
    </dependency>
    <dependency>
      <groupId>io.netty</groupId>
      <artifactId>netty-handler</artifactId>
      <version>${netty.version}</version>
    </dependency>
  </dependencies>
</project>
```

2) 서버 부트스트랩 (NettyTcpServer.java)

```
java

package dev.example.nettytcp;

import io.netty.bootstrap.ServerBootstrap;
import io.netty.channel.*;
import io.netty.channel.nio.NioEventLoopGroup;
import io.netty.channel.socket.SocketChannel;
import io.netty.channel.socket.nio.NioServerSocketChannel;
```

```
import io.netty.handler.codec.LineBasedFrameDecoder;
import io.netty.handler.codec.string.StringDecoder;
import io.netty.handler.codec.string.StringEncoder;
import io.netty.handler.logging.LogLevel;
import io.netty.handler.logging.LoggingHandler;
import io.netty.util.CharsetUtil;

public class NettyTcpServer {

    public static void main(String[] args) throws InterruptedException {
        final int port = args.length > 0 ? Integer.parseInt(args[0]) :
            Integer.parseInt(System.getenv().getOrDefault("PORT", "7000"));

        // boss: accept 전담, worker: IO 처리
        EventLoopGroup boss = new NioEventLoopGroup(1);
        EventLoopGroup worker = new NioEventLoopGroup(); // CPU 코어 수 기반

        try {
            ServerBootstrap b = new ServerBootstrap();
            b.group(boss, worker)
                .channel(NioServerSocketChannel.class)
                // 서버 소켓 옵션
                .option(ChannelOption.SO_BACKLOG, 1024)
                .handler(new LoggingHandler(LogLevel.INFO))
                // 클라이언트 채널 옵션
                .childOption(ChannelOption.TCP_NODELAY, true)
                .childOption(ChannelOption.SO_KEEPALIVE, true)
                .childHandler(new ChannelInitializer<SocketChannel>() {
                    @Override
                    protected void initChannel(SocketChannel ch) {
                        ChannelPipeline p = ch.pipeline();
                        // 1) 줄 단위 프레임링: 너무 긴 줄로 인한 OOM 방지를 위해 최대길이 제한
                        p.addLast(new LineBasedFrameDecoder(8 * 1024));
                        // 2) 텍스트 디코더/인코더 (UTF-8)
                        p.addLast(new StringDecoder(CharsetUtil.UTF_8));
                        p.addLast(new StringEncoder(CharsetUtil.UTF_8));
                        // 3) 비즈니스 핸들러
                        p.addLast(new EchoServerHandler());
                    }
                });

            ChannelFuture bindFuture = b.bind(port).sync();
            System.out.println("[NettyTcpServer] started on port " + port);

            // 종료 신호 대기
            bindFuture.channel().closeFuture().sync();
        } finally {
            // 그라운드스톱을 셧다운
            boss.shutdownGracefully();
            worker.shutdownGracefully();
        }
    }
}
```

3) 에코 처리 핸들러 (EchoServerHandler.java)

```
java

package dev.example.nettytcp;

import io.netty.channel.ChannelHandlerContext;
import io.netty.channel.SimpleChannelInboundHandler;

public class EchoServerHandler extends SimpleChannelInboundHandler<String> {

    @Override
    public void channelActive(ChannelHandlerContext ctx) {
        ctx.writeAndFlush("Welcome to Netty TCP server. Type something and hit Enter.\n");
    }
}
```

```

}

@Override
protected void channelRead0(ChannelHandlerContext ctx, String msg) {
    String trimmed = msg.strip();
    if (trimmed.equalsIgnoreCase("quit") || trimmed.equalsIgnoreCase("exit")) {
        ctx.writeAndFlush("Bye!\n").addListener(f -> ctx.close());
        return;
    }
    // 간단한 에코 + 길이 표시
    String response = "[echo] (" + trimmed.length() + " bytes) " + trimmed + "\n";
    ctx.writeAndFlush(response);
}

@Override
public void exceptionCaught(ChannelHandlerContext ctx, Throwable cause) {
    // 필요 시 로깅 추가
    cause.printStackTrace();
    ctx.close();
}
}

```

실행

```

bash

mvn -q -DskipTests package
java -cp target/netty-tcp-server-1.0.0.jar dev.example.nettytcp.NettyTcpServer 7000

```

테스트 (리눅스/WSL/맥)

```

bash

# 1) nc로 접속
nc 127.0.0.1 7000
hello
# -> [echo] (5 bytes) hello
exit
# -> Bye!

# 윈도우 PowerShell
# choco install nmap (또는 Git Bash에서 nc 사용) / telnet도 가능

```

(옵션) 길이필드 기반 프레임िंग으로 바꾸고 싶다면

바이너리/가변길이 프로토콜을 쓰고 싶을 때 **LengthFieldBasedFrameDecoder/Prepender**를 쓰면 안전합니다.

```

java

// import io.netty.handler.codec.LengthFieldBasedFrameDecoder;
// import io.netty.handler.codec.LengthFieldPrepender;
// import io.netty.handler.codec.string.StringDecoder;
// import io.netty.handler.codec.string.StringEncoder;
// frame: [length(4bytes)][payload...]
p.addLast(new LengthFieldBasedFrameDecoder(
    64 * 1024, // maxFrameLength
    0,        // lengthFieldOffset
    4,        // lengthFieldLength
    0,        // lengthAdjustment

```

```
4)); // initialBytesToStrip (길이필드 제거)

p.addLast(new LengthFieldPrepender(4));
p.addLast(new StringDecoder(CharsetUtil.UTF_8));
p.addLast(new StringEncoder(CharsetUtil.UTF_8));
p.addLast(new EchoServerHandler());
```

운영 팁 한 줄

- 백프레서: `channelWritabilityChanged()`를 오버라이드해 `!ctx.channel().isWritable()`일 때 생산 속도 조절.
- Idle 감시: `IdleStateHandler`로 죽은 연결 정리(예: 60초 read idle → close).
- TLS: `SslContextBuilder.forServer(cert, key)`를 파이프라인 최상단에 추가.
- Graceful Stop: 종료 신호 받을 때 `Channel.close()` 후 `EventLoopGroup.shutdownGracefully()`.

필요하시면 Gradle 템플릿이나, 멀티포트/멀티프로토콜(예: 텔넷형 + 길이필드형) 동시 구동 예제로 확장해 드리겠습니다.



이클립스 윈도우빌드를 이용해서 gui코드를 만들고 tcp서버 코드와 연동해줘

좋습니다. “이클립스(Windows)에서 바로 돌릴 수 있는 SWT GUI + Netty TCP 서버” 통합 예제를 드릴게요. GUI에서 포트를 입력하고 [Start]/[Stop]으로 서버를 제어하고, 접속/메시지 로그를 창에 뿌립니다. Maven 하나로 끝.

1) pom.xml

```
xml

<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>dev.example</groupId>
    <artifactId>swt-netty-server</artifactId>
    <version>1.0.0</version>

    <properties>
        <maven.compiler.source>17</maven.compiler.source>
        <maven.compiler.target>17</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <!-- Netty & SWT 버전은 환경에 맞게 조정 가능 -->
        <netty.version>4.1.109.Final</netty.version>
        <swt.version>3.124.0</swt.version> <!-- 예시: 최신 사용 가능 -->
    </properties>

    <dependencies>
        <!-- Netty -->
        <dependency>
            <groupId>io.netty</groupId>
            <artifactId>netty-transport</artifactId>
            <version>${netty.version}</version>
        </dependency>
        <dependency>
            <groupId>io.netty</groupId>
```

```

    <artifactId>netty-codec</artifactId>
    <version>${netty.version}</version>
</dependency>
<dependency>
    <groupId>io.netty</groupId>
    <artifactId>netty-handler</artifactId>
    <version>${netty.version}</version>
</dependency>
<dependency>
    <groupId>io.netty</groupId>
    <artifactId>netty-buffer</artifactId>
    <version>${netty.version}</version>
</dependency>

<!-- SWT (Windows x86_64용) -->
<dependency>
    <groupId>org.eclipse.platform</groupId>
    <artifactId>org.eclipse.swt.win32.win32.x86_64</artifactId>
    <version>${swt.version}</version>
</dependency>
</dependencies>

<build>
    <plugins>
        <!-- 실행가능 JAR -->
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-jar-plugin</artifactId>
            <version>3.3.0</version>
            <configuration>
                <archive>
                    <manifest>
                        <mainClass>dev.example.gui.App</mainClass>
                    </manifest>
                </archive>
            </configuration>
        </plugin>

        <!-- 의존성까지 모으는 assembly (단일 폴더 실행용) -->
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-assembly-plugin</artifactId>
            <version>3.6.0</version>
            <configuration>
                <descriptorRefs>
                    <descriptorRef>jar-with-dependencies</descriptorRef>
                </descriptorRefs>
                <archive>
                    <manifest>
                        <mainClass>dev.example.gui.App</mainClass>
                    </manifest>
                </archive>
                <finalName>${project.artifactId}-${project.version}-all</finalName>
            </configuration>
            <executions>
                <execution>
                    <id>make-assembly</id>
                    <phase>package</phase>
                    <goals><goal>single</goal></goals>
                </execution>
            </executions>
        </plugin>

        <!-- (선택) exe 래퍼: launch4j (WiX 없이 exe 생성) -->
        <plugin>
            <groupId>com.akathist.maven.plugins.launch4j</groupId>
            <artifactId>launch4j-maven-plugin</artifactId>
            <version>2.5.0</version>
            <configuration>
                <outfile>target/swt-netty-server.exe</outfile>
                <jar>target/${project.artifactId}-${project.version}-all.jar</jar>
                <classPath>
                    <mainClass>dev.example.gui.App</mainClass>
                </classPath>
            </configuration>
        </plugin>
    </plugins>
</build>

```

```

<dontWrapJar>true</dontWrapJar>
<headerType>gui</headerType>
<jre>
  <minVersion>17</minVersion>
  <jdkPreference>preferJre</jdkPreference>
</jre>
<icon></icon>
<companyName>Example</companyName>
<productName>SWT Netty Server</productName>
</configuration>
<executions>
  <execution>
    <id>l4j-cl</id>
    <phase>package</phase>
    <goals><goal>launch4j</goal></goals>
  </execution>
</executions>
</plugin>
</plugins>
</build>
</project>

```

참고: MSI 설치파일까지 원하면 `jpackage` + WiX가 필요합니다. 단순 exe면 위 Launch4j가 편합니다.

2) Netty 서버 수명주기 (ServerManager.java)

```

java

package dev.example.server;

import io.netty.bootstrap.ServerBootstrap;
import io.netty.channel.*;
import io.netty.channel.nio.NioEventLoopGroup;
import io.netty.channel.socket.nio.NioServerSocketChannel;
import io.netty.handler.codec.LineBasedFrameDecoder;
import io.netty.handler.codec.string.StringDecoder;
import io.netty.handler.codec.string.StringEncoder;
import io.netty.util.CharsetUtil;

import java.util.concurrent.CompletableFuture;
import java.util.concurrent.atomic.AtomicBoolean;
import java.util.function.Consumer;

public class ServerManager {
    private NioEventLoopGroup boss;
    private NioEventLoopGroup worker;
    private Channel serverChannel;
    private final AtomicBoolean running = new AtomicBoolean(false);
    private final Consumer<String> log;

    public ServerManager(Consumer<String> logAppender) {
        this.log = logAppender;
    }

    public CompletableFuture<Void> startAsync(int port) {
        CompletableFuture<Void> cf = new CompletableFuture<>();
        if (running.get()) {
            log.accept("[WARN] already running");
            cf.complete(null);
            return cf;
        }
        new Thread(() -> {
            try {
                boss = new NioEventLoopGroup(1);
                worker = new NioEventLoopGroup();
                ServerBootstrap b = new ServerBootstrap();
                b.group(boss, worker)
                  .channel(NioServerSocketChannel.class)
                  .option(ChannelOption.SO_BACKLOG, 1024)
                  .childOption(ChannelOption.TCP_NODELAY, true)

```

```

        .childOption(ChannelOption.SO_KEEPALIVE, true)
        .childHandler(new ChannelInitializer<Channel>() {
            @Override
            protected void initChannel(Channel ch) {
                ChannelPipeline p = ch.pipeline();
                p.addLast(new LineBasedFrameDecoder(8 * 1024));
                p.addLast(new StringDecoder(CharsetUtil.UTF_8));
                p.addLast(new StringEncoder(CharsetUtil.UTF_8));
                p.addLast(new TcpHandler(log));
            }
        });

        ChannelFuture bind = b.bind(port).sync();
        serverChannel = bind.channel();
        running.set(true);
        log.accept("[INFO] server started on port " + port);

        // 블로킹 대기
        serverChannel.closeFuture().sync();
        cf.complete(null);
    } catch (Exception e) {
        log.accept("[ERROR] " + e.getMessage());
        cf.completeExceptionally(e);
    } finally {
        shutdownGroups();
        running.set(false);
        log.accept("[INFO] server stopped");
    }
}, "netty-server-thread").start();
return cf;
}

public void stop() {
    if (!running.get()) {
        log.accept("[WARN] server not running");
        return;
    }
    try {
        if (serverChannel != null) {
            serverChannel.close().syncUninterruptibly();
        }
    } finally {
        shutdownGroups();
        running.set(false);
    }
}

private void shutdownGroups() {
    if (boss != null) boss.shutdownGracefully();
    if (worker != null) worker.shutdownGracefully();
}

public boolean isRunning() {
    return running.get();
}
}

```

3) TCP 핸들러 (TcpHandler.java)

```

java

package dev.example.server;

import io.netty.channel.ChannelHandlerContext;
import io.netty.channel.SimpleChannelInboundHandler;

import java.net.InetSocketAddress;
import java.util.function.Consumer;

public class TcpHandler extends SimpleChannelInboundHandler<String> {

```

```
private final Consumer<String> log;

public TcpHandler(Consumer<String> log) {
    this.log = log;
}

@Override
public void channelActive(ChannelHandlerContext ctx) {
    InetSocketAddress r = (InetSocketAddress) ctx.channel().remoteAddress();
    log.accept("[CONN] " + r.getAddress().getHostAddress() + ":" + r.getPort() + "
connected");
    ctx.writeAndFlush("Welcome to Netty TCP server.\n");
}

@Override
protected void channelRead0(ChannelHandlerContext ctx, String msg) {
    String t = msg.strip();
    log.accept("[RECV] " + t);
    if (t.equalsIgnoreCase("quit") || t.equalsIgnoreCase("exit")) {
        ctx.writeAndFlush("Bye!\n").addListener(f -> ctx.close());
        return;
    }
    ctx.writeAndFlush("[echo] " + t + "\n");
}

@Override
public void exceptionCaught(ChannelHandlerContext ctx, Throwable cause) {
    log.accept("[EX] " + cause.getClass().getSimpleName() + ": " + cause.getMessage());
    ctx.close();
}

@Override
public void channelInactive(ChannelHandlerContext ctx) {
    log.accept("[DISC] connection closed");
}
}
```

4) SWT GUI (App.java)

```
java

package dev.example.gui;

import dev.example.server.ServerManager;
import org.eclipse.swt.SWT;
import org.eclipse.swt.custom.StyledText;
import org.eclipse.swt.layout.GridData;
import org.eclipse.swt.layout.GridLayout;
import org.eclipse.swt.widgets.*;

import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

public class App {

    private static final DateTimeFormatter TS = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");
    private ServerManager server;
    private StyledText logArea;
    private Text portText;
    private Button startBtn;
    private Button stopBtn;
    private Display display;

    public static void main(String[] args) {
        new App().run();
    }

    private void run() {
        display = new Display();
        Shell shell = new Shell(display);
```



```

shell.setText("Netty TCP Server (SWT)");
shell.setLayout(new GridLayout(4, false));
shell.setSize(760, 520);

Label portLabel = new Label(shell, SWT.NONE);
portLabel.setText("Port:");

portText = new Text(shell, SWT.BORDER);
portText.setLayoutData(new GridData(SWT.FILL, SWT.CENTER, false, false));
portText.setText("7000");
((GridData) portText.getLayoutData()).widthHint = 100;

startBtn = new Button(shell, SWT.PUSH);
startBtn.setText("Start");
stopBtn = new Button(shell, SWT.PUSH);
stopBtn.setText("Stop");
stopBtn.setEnabled(false);

// 로그 영역
logArea = new StyledText(shell, SWT.BORDER | SWT.V_SCROLL | SWT.H_SCROLL);
GridData gd = new GridData(SWT.FILL, SWT.FILL, true, true);
gd.horizontalSpan = 4;
logArea.setLayoutData(gd);
logArea.setEditable(false);

// 서버 매니저 (GUI 스레드와 분리, UI append는 asyncExec로 안전하게)
server = new ServerManager(this::appendLog);

startBtn.addListener(SWT.Selection, e -> {
    int port = parsePort(portText.getText());
    if (port <= 0) {
        alert(shell, "유효한 포트를 입력하세요 (1~65535).");
        return;
    }
    setButtonsRunning(true);
    appendLog("[UI] starting server on port " + port + " ...");
    server.startAsync(port).whenComplete((v, ex) -> {
        // 서버 스레드 종료 시 버튼 상태 복구
        async(() -> setButtonsRunning(false));
        if (ex != null) {
            async(() -> alert(shell, "서버 시작 실패: " + ex.getMessage()));
        }
    });
});

stopBtn.addListener(SWT.Selection, e -> {
    appendLog("[UI] stopping server ...");
    server.stop();
    setButtonsRunning(false);
});

// 창 닫을 때 서버 정리
shell.addListener(SWT.Close, e -> {
    if (server.isRunning()) {
        server.stop();
    }
});

shell.open();
while (!shell.isDisposed()) {
    if (!display.readAndDispatch()) display.sleep();
}
display.dispose();
}

private int parsePort(String s) {
    try {
        int p = Integer.parseInt(s.trim());
        return (p >= 1 && p <= 65535) ? p : -1;
    } catch (Exception e) {
        return -1;
    }
}

```

```
private void setButtonsRunning(boolean running) {
    startBtn.setEnabled(!running);
    stopBtn.setEnabled(running);
    portText.setEnabled(!running);
}

private void appendLog(String line) {
    String msg = String.format("%s %s%n", TS.format(LocalDateTime.now()), line);
    async(() -> {
        if (logArea.isDisposed()) return;
        logArea.append(msg);
        logArea.setTopIndex(logArea.getLineCount() - 1);
    });
}

private void async(Runnable r) {
    if (display == null || display.isDisposed()) return;
    display.asyncExec(r);
}

private void alert(Shell parent, String msg) {
    MessageBox box = new MessageBox(parent, SWT.ICON_WARNING | SWT.OK);
    box.setText("알림");
    box.setMessage(msg);
    box.open();
}
}
```

빌드 & 실행

```
bash

# 패키징 (의존성 포함 JAR 생성)
mvn -q -DskipTests package

# 실행 (all-jar 사용)
java -jar target/swt-netty-server-1.0.0-all.jar
```

동작 확인

1. GUI에서 포트 7000으로 Start.
2. 다른 터미널에서:

```
bash

nc 127.0.0.1 7000
hello
# -> [echo] hello
exit
# -> Bye!
```

3. GUI 로그창에 [CONN], [RECV], [DISC] 메시지 흐르는지 확인.

Windows exe 만들기 (WiX 없이)

```
bash

mvn -q -DskipTests package
# 결과: target/swt-netty-server.exe
```

- 실행에 JRE 17+이 필요합니다. JRE 동봉까지 원하면 **jlink + jpackage**로 커스텀 런타임 번들링을 검토하세요(이 경우 MSI엔 WiX 필요).

확장 아이디어 (필요하시면 붙여드립니다)

- **TLS on/off** 토글 + cert/key 선택 UI → 파이프라인 **SslHandler** 추가.
- **IdleStateHandler** 설정값 UI → 유휴 연결 정리.
- **프로토콜 선택**: LineBased ↔ LengthField 기반 스위치, 최대 프레임 크기 조절.
- **접속 리스트뷰**(Table) + 강제 close 버튼.
- **로그 파일로도 저장**(java.util.logging 또는 logback 추가).

원하시면 JavaFX 버전이나, “서버+클라이언트 테스트 탭”까지 들어간 올인원 툴로 확장해서 드릴게요.



gui에서 tcp 서버에 접속기능을 넣어줘. 접속해서 커멘드를 입력하면 tcp서버 응답하는 정보를 gui에 출력 하는 기능을 구현해줘

좋아요. 기존 SWT GUI에 **TCP 클라이언트** 기능을 붙여서, GUI에서 서버에 **접속** → **명령 전송** → **응답 수신**을 **GUI에 출력**하도록 확장해드립니다.

서버/클라이언트 둘 다 Netty로 통일했고, 라인기반 프레이밍(개행 \n)을 씁니다.

1) pom.xml (그대로 사용)

앞서 드린 pom.xml 그대로 쓰면 됩니다(Netty + SWT 의존성 포함).

2) 클라이언트 매니저 (ClientManager.java)

```
java

package dev.example.client;

import io.netty.bootstrap.Bootstrap;
import io.netty.channel.*;
import io.netty.channel.nio.NioEventLoopGroup;
import io.netty.channel.socket.nio.NioSocketChannel;
import io.netty.handler.codec.LineBasedFrameDecoder;
import io.netty.handler.codec.string.StringDecoder;
import io.netty.handler.codec.string.StringEncoder;
import io.netty.util.CharsetUtil;

import java.util.concurrent.CompletableFuture;
```

```
import java.util.concurrent.atomic.AtomicBoolean;
import java.util.function.Consumer;

public class ClientManager {
    private final Consumer<String> log;
    private NioEventLoopGroup group;
    private Channel clientChannel;
    private final AtomicBoolean connected = new AtomicBoolean(false);

    public ClientManager(Consumer<String> logAppender) {
        this.log = logAppender;
    }

    public CompletableFuture<Void> connectAsync(String host, int port) {
        CompletableFuture<Void> cf = new CompletableFuture<>();
        if (connected.get()) {
            log.accept("[WARN] client already connected");
            cf.complete(null);
            return cf;
        }

        new Thread(() -> {
            try {
                group = new NioEventLoopGroup();
                Bootstrap b = new Bootstrap();
                b.group(group)
                  .channel(NioSocketChannel.class)
                  .option(ChannelOption.TCP_NODELAY, true)
                  .handler(new ChannelInitializer<Channel>() {
                      @Override
                      protected void initChannel(Channel ch) {
                          ChannelPipeline p = ch.pipeline();
                          p.addLast(new LineBasedFrameDecoder(8 * 1024));
                          p.addLast(new StringDecoder(CharsetUtil.UTF_8));
                          p.addLast(new StringEncoder(CharsetUtil.UTF_8));
                          p.addLast(new ClientHandler(log));
                      }
                  });

                ChannelFuture f = b.connect(host, port).sync();
                clientChannel = f.channel();
                connected.set(true);
                log.accept("[CLIENT] connected to " + host + ":" + port);

                // 블로킹 close 대기(백그라운드 스레드)
                clientChannel.closeFuture().sync();
                cf.complete(null);
            } catch (Exception e) {
                log.accept("[CLIENT-ERROR] " + e.getMessage());
                cf.completeExceptionally(e);
            } finally {
                disconnectInternal();
                log.accept("[CLIENT] disconnected");
            }
        }, "netty-client-thread").start();

        return cf;
    }

    public void sendLine(String line) {
        if (clientChannel == null || !clientChannel.isActive()) {
            log.accept("[CLIENT] not connected");
            return;
        }
        // 서버가 LineBased이므로 개행 붙여 전송
        clientChannel.writeAndFlush(line + "\n");
        log.accept("[CLIENT-SENT] " + line);
    }

    public void disconnect() {
        if (clientChannel != null) {
            clientChannel.close().syncUninterruptibly();
        } else {
            disconnectInternal();
        }
    }
}
```

```

    }
}

private void disconnectInternal() {
    connected.set(false);
    if (group != null) group.shutdownGracefully();
    group = null;
    clientChannel = null;
}

public boolean isConnected() {
    return connected.get();
}
}

```

3) 클라이언트 핸들러 (ClientHandler.java)

```

java

package dev.example.client;

import io.netty.channel.ChannelHandlerContext;
import io.netty.channel.SimpleChannelInboundHandler;

import java.util.function.Consumer;

public class ClientHandler extends SimpleChannelInboundHandler<String> {
    private final Consumer<String> log;

    public ClientHandler(Consumer<String> log) {
        this.log = log;
    }

    @Override
    public void channelActive(ChannelHandlerContext ctx) {
        log.accept("[CLIENT] channel active");
    }

    @Override
    protected void channelRead0(ChannelHandlerContext ctx, String msg) {
        log.accept("[CLIENT-RECV] " + msg.strip());
    }

    @Override
    public void exceptionCaught(ChannelHandlerContext ctx, Throwable cause) {
        log.accept("[CLIENT-EX] " + cause.getClass().getSimpleName() + ": " +
        cause.getMessage());
        ctx.close();
    }

    @Override
    public void channelInactive(ChannelHandlerContext ctx) {
        log.accept("[CLIENT] channel inactive");
    }
}

```

4) SWT GUI 수정 (App.java)

기존 서버 탭에 **클라이언트 패널**(호스트/포트/Connect/Disconnect/입력/Send)을 추가하고, 로그는 한 곳 (StyledText)에 태그로 구분해 적재합니다.

```

java

package dev.example.gui;

import dev.example.server.ServerManager;
import dev.example.client.ClientManager;
import org.eclipse.swt.SWT;
import org.eclipse.swt.custom.StyledText;
import org.eclipse.swt.events.KeyAdapter;
import org.eclipse.swt.events.KeyEvent;
import org.eclipse.swt.layout.GridData;
import org.eclipse.swt.layout.GridLayout;
import org.eclipse.swt.widgets.*;

import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;

public class App {

    private static final DateTimeFormatter TS = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");

    // Server
    private ServerManager server;
    private Text srvPortText;
    private Button srvStartBtn;
    private Button srvStopBtn;

    // Client
    private ClientManager client;
    private Text cliHostText;
    private Text cliPortText;
    private Button cliConnBtn;
    private Button cliDiscBtn;
    private Text cliInputText;
    private Button cliSendBtn;

    // Common
    private StyledText logArea;
    private Display display;

    public static void main(String[] args) {
        new App().run();
    }

    private void run() {
        display = new Display();
        Shell shell = new Shell(display);
        shell.setText("Netty TCP Server & Client (SWT)");
        shell.setLayout(new GridLayout(6, false));
        shell.setSize(900, 600);

        // ===== Server Panel =====
        Label srvTitle = new Label(shell, SWT.NONE);
        srvTitle.setText("[Server]");
        GridData tgd = new GridData(SWT.LEFT, SWT.CENTER, false, false);
        tgd.horizontalSpan = 1;
        srvTitle.setLayoutData(tgd);

        new Label(shell, SWT.NONE).setText("Port:");
        srvPortText = new Text(shell, SWT.BORDER);
        srvPortText.setText("7000");
        GridData srvPortGD = new GridData(SWT.FILL, SWT.CENTER, false, false);
        srvPortGD.widthHint = 90;
        srvPortText.setLayoutData(srvPortGD);

        srvStartBtn = new Button(shell, SWT.PUSH);
        srvStartBtn.setText("Start");
        srvStopBtn = new Button(shell, SWT.PUSH);
        srvStopBtn.setText("Stop");
        srvStopBtn.setEnabled(false);

        // 빈칸 채우기
    }
}

```

```

new Label(shell, SWT.NONE).setLayoutData(new GridData(SWT.FILL, SWT.CENTER, true,
false));

// ===== Client Panel =====
Label cliTitle = new Label(shell, SWT.NONE);
cliTitle.setText("[Client]");
GridData cgd1 = new GridData(SWT.LEFT, SWT.CENTER, false, false);
cgd1.horizontalSpan = 1;
cliTitle.setLayoutData(cgd1);

new Label(shell, SWT.NONE).setText("Host:");
cliHostText = new Text(shell, SWT.BORDER);
cliHostText.setText("127.0.0.1");
GridData hostGD = new GridData(SWT.FILL, SWT.CENTER, false, false);
hostGD.widthHint = 140;
cliHostText.setLayoutData(hostGD);

new Label(shell, SWT.NONE).setText("Port:");
cliPortText = new Text(shell, SWT.BORDER);
cliPortText.setText("7000");
GridData cportGD = new GridData(SWT.FILL, SWT.CENTER, false, false);
cportGD.widthHint = 90;
cliPortText.setLayoutData(cportGD);

cliConnBtn = new Button(shell, SWT.PUSH);
cliConnBtn.setText("Connect");
cliDiscBtn = new Button(shell, SWT.PUSH);
cliDiscBtn.setText("Disconnect");
cliDiscBtn.setEnabled(false);

// 입력 라인
new Label(shell, SWT.NONE).setText("Input:");
cliInputText = new Text(shell, SWT.BORDER);
GridData inGD = new GridData(SWT.FILL, SWT.CENTER, true, false);
inGD.horizontalSpan = 4;
cliInputText.setLayoutData(inGD);
cliSendBtn = new Button(shell, SWT.PUSH);
cliSendBtn.setText("Send");
cliSendBtn.setEnabled(false);

// 로그 창
logArea = new StyledText(shell, SWT.BORDER | SWT.V_SCROLL | SWT.H_SCROLL);
GridData lgd = new GridData(SWT.FILL, SWT.FILL, true, true);
lgd.horizontalSpan = 6;
logArea.setLayoutData(lgd);
logArea.setEditable(false);

// 매니저들
server = new ServerManager(this::appendLog);
client = new ClientManager(this::appendLog);

// === 이벤트 바인딩 ===
srvStartBtn.addListener(SWT.Selection, e -> {
    int port = parsePort(srvPortText.getText());
    if (port <= 0) { alert(shell, "유효한 포트를 입력하세요 (1~65535)."); return; }
    setServerButtons(true);
    appendLog("[UI] starting server on port " + port + " ...");
    server.startAsync(port).whenComplete((v, ex) -> {
        async(() -> setServerButtons(false)); // 서버 스레드 종료 시
        if (ex != null) async(() -> alert(shell, "서버 시작 실패: " + ex.getMessage()));
    });
});

srvStopBtn.addListener(SWT.Selection, e -> {
    appendLog("[UI] stopping server ...");
    server.stop();
    setServerButtons(false);
});

cliConnBtn.addListener(SWT.Selection, e -> {
    String host = cliHostText.getText().trim();
    int port = parsePort(cliPortText.getText());
    if (host.isEmpty() || port <= 0) { alert(shell, "Host/Port를 확인하세요."); return; }

```

```

        setClientButtons(true);
        appendLog("[UI] connecting to " + host + ":" + port + " ...");
        client.connectAsync(host, port).whenComplete((v, ex) -> {
            async(() -> setClientButtons(false)); // 연결 종료 시
            if (ex != null) async(() -> alert(shell, "클라이언트 연결 실패: " +
ex.getMessage()));
        });

        cliDiscBtn.addListener(SWT.Selection, e -> {
            appendLog("[UI] disconnecting client ...");
            client.disconnect();
            setClientButtons(false);
        });

        cliSendBtn.addListener(SWT.Selection, e -> sendFromInput());
        cliInputText.addKeyListener(new KeyAdapter() {
            @Override public void keyPressed(KeyEvent e) {
                if (e.character == SWT.CR || e.keyCode == SWT.KEYPAD_CR) {
                    sendFromInput();
                }
            }
        });

        // 종료 처리
        shell.addListener(SWT.Close, e -> {
            if (server != null && server.isRunning()) server.stop();
            if (client != null && client.isConnected()) client.disconnect();
        });

        shell.open();
        while (!shell.isDisposed()) {
            if (!display.readAndDispatch()) display.sleep();
        }
        display.dispose();
    }

    private void sendFromInput() {
        String line = cliInputText.getText().trim();
        if (line.isEmpty()) return;
        client.sendLine(line);
        cliInputText.setText("");
    }

    private int parsePort(String s) {
        try {
            int p = Integer.parseInt(s.trim());
            return (p >= 1 && p <= 65535) ? p : -1;
        } catch (Exception e) { return -1; }
    }

    private void setServerButtons(boolean running) {
        srvStartBtn.setEnabled(!running);
        srvStopBtn.setEnabled(running);
        srvPortText.setEnabled(!running);
    }

    private void setClientButtons(boolean connected) {
        cliConnBtn.setEnabled(!connected);
        cliDiscBtn.setEnabled(connected);
        cliHostText.setEnabled(!connected);
        cliPortText.setEnabled(!connected);
        cliSendBtn.setEnabled(connected);
        cliInputText.setEnabled(connected);
    }

    private void appendLog(String line) {
        String msg = String.format("%s %s\n", TS.format(LocalDateTime.now()), line);
        async(() -> {
            if (logArea.isDisposed()) return;
            logArea.append(msg);
            logArea.setTopIndex(logArea.getLineCount() - 1);
        });
    }
}

```



```
private void async(Runnable r) {
    if (display == null || display.isDisposed()) return;
    display.asyncExec(r);
}

private void alert(Shell parent, String msg) {
    MessageBox box = new MessageBox(parent, SWT.ICON_WARNING | SWT.OK);
    box.setText("알림");
    box.setMessage(msg);
    box.open();
}
}
```

5) 서버 핸들러(재사용)

앞서 제공한 **ServerManager**, **TcpHandler**(개행 기반 에코) 그대로 사용하시면 됩니다.

- 클라이언트에서 입력한 라인이 서버로 전송되고, 서버는 **[echo] ...** 형태로 응답 → GUI 로그에 **[CLIENT-RECV]** ...로 찍힙니다.
- exit/quit**를 보내면 서버가 해당 연결을 종료합니다.

빌드 & 실행

```
bash

mvn -q -DskipTests package
java -jar target/swt-netty-server-1.0.0-all.jar
```

사용 흐름

- Server:** Port 7000 → Start. (로그에 **[INFO] server started...**)
- Client:** Host 127.0.0.1, Port 7000 → Connect.
- Input**에 **hello** 입력 후 Enter 또는 Send → 로그에
 - [CLIENT-SENT] hello**
 - [CLIENT-RECV] [echo] hello**
- 종료는
 - Client 쪽 **Disconnect** 또는 **exit** 전송
 - Server 쪽 **Stop**

확장 포인트

- 길이필드 프레임ING 선택 토글(서버/클라이언트 파이프라인 모두 스위치)
- TLS 체크박스 + cert/key 파일 선택 → **SslHandler** 추가
- 접속 목록/현재 세션 상태(Table) + 강제 종료 버튼

- **로그 파일 저장**(일자 로테이션)
- **자동 재연결**(지수 백오프)

원하시면 길이필드 기반 버전으로 한 번에 스위치되는 옵션까지 GUI에 넣어 드릴게요.