대략적인 구현의 흐름

서버클래스

메인

ExecutorService threadPool을 통해 여러 스레드를 자동으로 관리

여러 클라이언트들을 수용해야 하므로 클라이언트에 대한처리를 따로 하고 각 클라이언트를 담는 vector clients 생성

start에서 시작되면 인터페이스를 구성한다.

인터페이스를 구성하면서 각 기능의 액션에 대한 처리를 한다.

javafx특성상 인자로 Stage primaryStage를 가지고 있다.

버튼이 눌릴경우 버튼의 내용에따라 다른 기능을 수행한다.

서버시작의 경우 서버를 시작하는 메시지를 textarea에 나타낸뒤 서버를 시작하는 메소드를 수행하고

서버종료의 경우도 마찬가지로 textarea아 메시지를 나타낸뒤 서버를 종료하는 메소드를 수행한다.

서버를 종료하는 메소드는

벡터에 저장된 각 클라이언트들을 iterator로 접근하여 해당 클라이언트의 소켓들을 닫고 최후에는 서버소켓을 닫고 threadpool을 종료한다.

서버를 시작하는 메소드는

서버소켓을 생성한뒤 bind메소드를 통해 자신의 ip와 지정한 포트번호로 바인딩한다

이후 지속적으로 클라이언트의 접속을 기다리는 스레드를 만들어 클라이언트가 접근하면 해당 소켓을 이용하여 새 클라이언트 클래스를 생성하여 벡터에 추가한다.

Sever-client클래스

생성자를 통해 서버의 메인클래스로부터 소켓을 전달받으면 클라이언트로부터 메시지 수신을 기다린다,

서버는 항상 클라이언트로부터의 메시지를 기다리도록 해야하며 송신과 별도로 이루어져야하기 때문에 별도의 Runnable thread로 처리한다.

getInputstream을 통해 메시지를 기다리다 메시지가 수신되면 해당 메시지를 모든 클라이언트가 볼 수 있도록 전송한다.

해당 스레드를 관리할 수 있도록 메인 스레드에서는 이 스레드를 메인의 스레드풀에 submit한다.

전송하는 메소드

마찬가지로 별개로 이루어지는 작업이므로 Runnable 스레드로 만들고

전송은 한번만 이루어지면 되므로 반복하지않는다.

Getoutputstream을 통해 아웃풋 스트림을 만들고

Write를 통해 메시지를 가져온다.

이 가져온 메시지는 flush를 통해 클라이언트측에 전달되어 receive에서 처리된다.

클라이언트측 메인 클래스

start에서 시작되면 인터페이스를 구성한다.

인터페이스를 구성하면서 각 기능의 액션에 대한 처리를 한다.

javafx특성상 인자로 Stage primaryStage를 가지고 있다.

서버시작버튼을 누른뒤 서버클라이언트메소드에 입력된 port와 ip값을 전달하여 시작한다.

클라이언트 측은 전송과 수신에 하나씩의 스레드만 필요하기 때문에 일반적인 Thread를 생성한다.

소켓을 만들어 인자로 ip와 포트번호를 넣어 서버와 연결한 후

수신 recive함수를 수행한다.

클라이언트를 멈추는 것은 간한히 소켓을 닫는것으로 구현한다.

메시지를 받는 receive 메소드는

Inputstream을 통해 메시지 수신을 대기하고 수신시 처리를 한 뒤 textarea에 수신된 메시지들을 표시한다. 자신이 메시지를 보낼때도 다른 클라이언트가 메시지를 보낼때도 모든 내용을 화면에 보여주도록 한다.(append를 통해 기존내용에 추가하는 방식)

별개로 recive가 수행되는 동안 메시지를 전송했을 때 send메소드가 수행된다

엔터를 누르거나 버튼이 눌릴 때 마다 해당 메소드가 수행되며 별개의 thread가 생성되어 전송을 처리한다.

Output stream을 통해 데이터를 내보낼 준비를하고 버퍼로부터 메시지를 읽어와 flush로 전송한다.