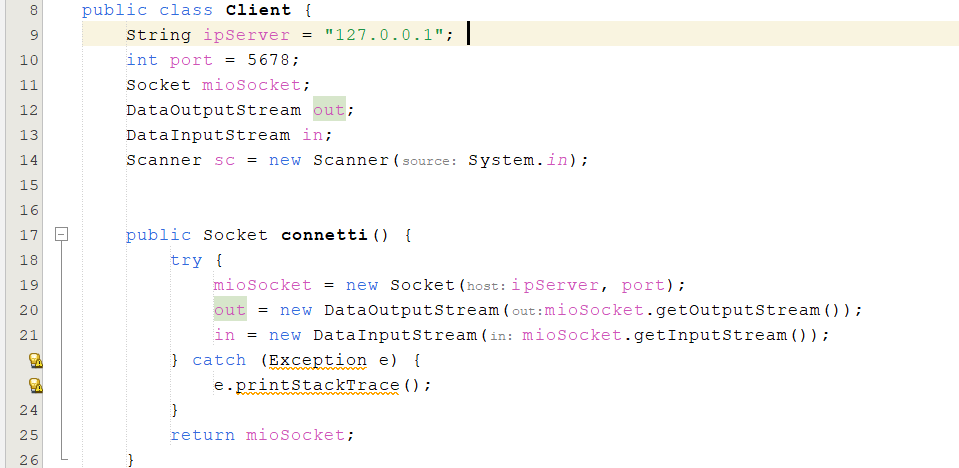
# Attività 1.1 (15 min):

Analizza il seguente scenario e prova a prevedere cosa succede:

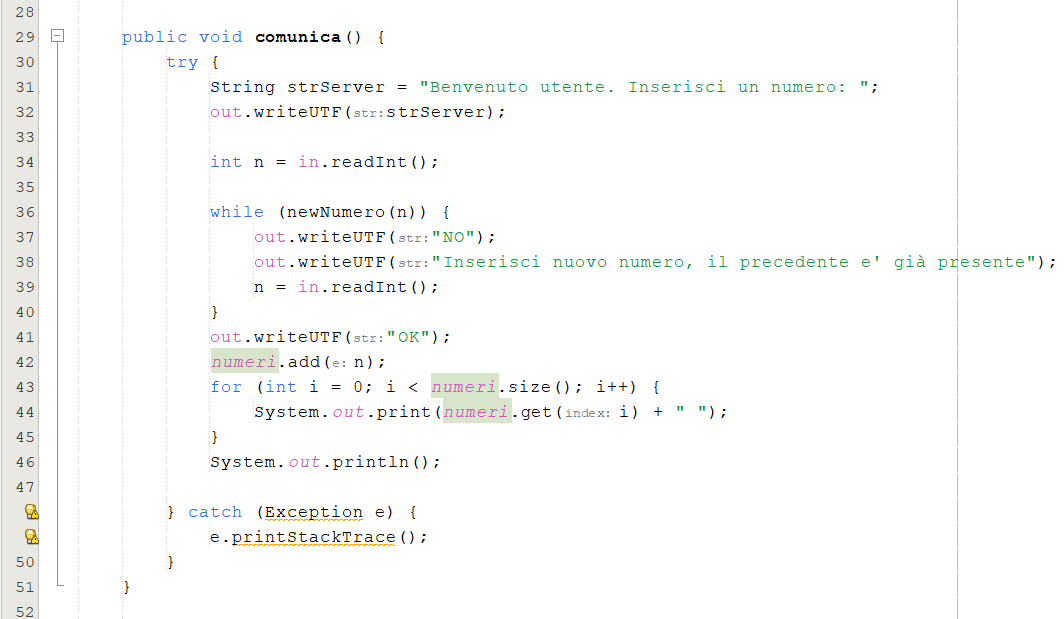
**Classe CLIENT:**

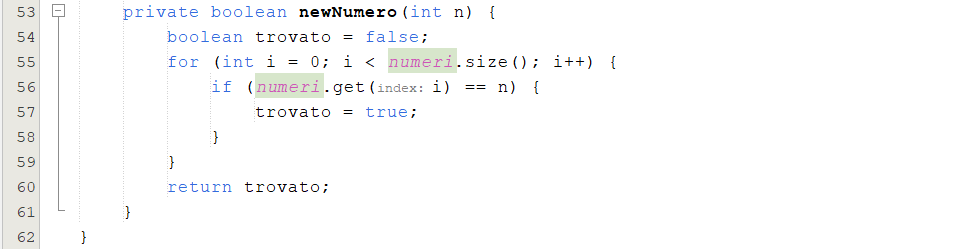




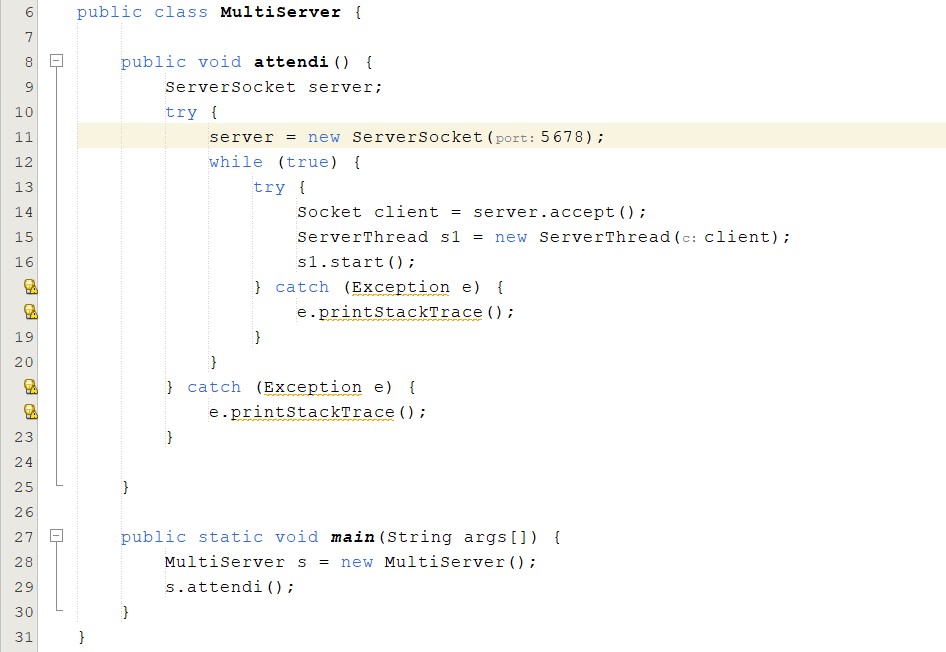
**Classe Server Thread**







**Classe Multi Server**



Qual è lo scopo del metodo run in ServerThread?

Il metodo run() in ServerThread ha lo scopo di inizializzare le variabili per l’input e l’output tra client e server, successivamente lancia il metodo comunica().

Cosa succede quando un utente invia un numero già presente nell’array numeri?

Quando un client inserisce un numero già presente nell’array il server chiede il reinserimento (mandando una stringa al client) a loop fino a che non è diverso rispetto agli altri numeri.

Cosa dovrebbe stampare il server quando un numero è aggiunto correttamente?

Il server quando un numero è stato aggiunto all’array correttamente manda al client una stringa “OK”. Poi viene visualizzato l’array aggiornato sul terminale del server.

**Attività 1.2 (15 min):**

Importare il codice. Testano il programma eseguendo il MultiServer e uno o più Client:

* Inserendo numeri univoci;
* Inserendo numeri duplicati.

**Domande:**

1. Cosa accade quando si inserisce un numero già presente?

Il server manda un messaggio di richiesta di cambiare il numero fino a che non è diverso nell’array.

1. Come reagisce il server se più client si collegano contemporaneamente?

Il server funziona in modo analogo a prima mantendendo l’input dei client.

1. Cosa viene stampato sulla console del server?

Ad ogni inserimento di un nuovo numero, viene stampato l’array aggiornato.

**Attività 2 (10 min):**

* Come funziona newNumero(int n)?

Il metodo newNumero(n) semplicemente controlla in tutto l’array se è presente il numero inserito (come parametron), se è gia presente restituisce **true** sennò restituisce **false.**

* Qual è il ruolo della sincronizzazione in un server multithread?

La sincronizzazione ci permette di poter avere un server multithread che gestisce molteplici client senza collisioni di accesso (per questo esempio) all’array numeri.

* Perché il metodo comunica() utilizza un ciclo per gestire input invalidi?

Cosi il server continua a chiedere un nuovo input senza dover andare in errore e interrompere il servizio

**Domande (15 min):**

**Modifica 1:**

Cambiare il messaggio del server per fornire più dettagli, ad esempio includendo l’elenco dei numeri attualmente presenti.

* Esempio: "Numero già presente. Numeri inseriti: [3, 7, 12]. Inserisci un nuovo numero:"

**Modifica 2:**

Il server deve accettare un numero massimo di client connessi (max 3).

**Esercizio (20 min)**

Esegui l’esercizio di seguito riportato:

Realizza un’applicazione che simuli una lotteria di 90 numeri; alcuni giocatori si collegano e “acquistano” alcuni numeri (per esempio 5) estratti casualmente dal server: quando si raggiunge un numero adeguato di giocatori (almeno 4), il server provvede all'estrazione di 5 premi, comunicando ai giocatori l’esito dell'estrazione; a loro volta i client, se hanno un numero vincente, lo segnalano al server per ritirare la vincita.

Il **Client**:

* Si connette;
* Riceve 5 numeri casuali;
* Aspetta il risultato dell'estrazione;
* Riceve quanti numeri ha indovinato.

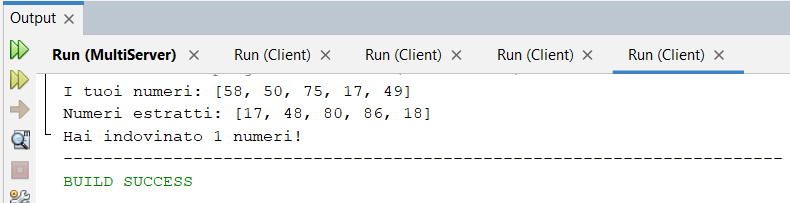
Il **ServerThread**:

* Genera 5 numeri casuali per il client;
* Invia i numeri al client;
* Invia i risultati quando avviene l'estrazione.

Il **MultiServer**:

* Aspetta 4 giocatori;
* Quando arrivano 4 giocatori, estrae 5 numeri;
* Controlla le vincite;
* Comunica i risultati.

*Esempio di output:*



**Tabella utile**

|  |  |  |  |
| --- | --- | --- | --- |
| **Metodo di scrittura** | **Descrizione** | **Tipo di dato** | **Metodo di lettura corrispondente** |
| writeBoolean(boolean v) | Scrive un valore booleano come 1 byte (1 per true, 0 per false). | boolean | readBoolean() |
| writeByte(int v) | Scrive un valore byte come 1 byte, usando i primi 8 bit di un int. | byte | readByte() |
| writeShort(int v) | Scrive un valore short di 2 byte in formato big-endian. | short | readShort() |
| writeChar(int v) | Scrive un carattere Unicode di 2 byte in formato big-endian. | char | readChar() |
| writeInt(int v) | Scrive un valore intero di 4 byte in formato big-endian. | int | readInt() |
| writeLong(long v) | Scrive un valore long di 8 byte in formato big-endian. | long | readLong() |
| writeFloat(float v) | Scrive un valore float di 4 byte in formato IEEE 754 (big-endian). | float | readFloat() |
| writeUTF(String s) | Scrive una stringa come testo UTF-8, con la lunghezza iniziale di 2 byte seguita dai caratteri UTF-8. | String | readUTF() |
| writeDouble(double v) | Scrive un valore double di 8 byte in formato IEEE 754 (big-endian). | double | readDouble() |
| writeBytes(String s) | Scrive una stringa come una sequenza di byte (un byte per carattere), senza lunghezza prefissata o codifica. | String | readFully(byte[] b) |
| writeChars(String s) | Scrive una stringa come una sequenza di caratteri Unicode, dove ogni carattere occupa 2 byte. | String | readChar() per ogni carattere |
| write(byte[] b) | Scrive un array di byte senza aggiungere lunghezza o delimitatori. | byte[] | readFully(byte[] b) |
| write(byte[] b, int off, int len) | Scrive una porzione specifica di un array di byte, partendo dall'offset off e scrivendo len byte. | byte[] | readFully(byte[] b, int off, int len) |