LAN Speed Tester

[1 Introduzione 3](#_Toc59124085)

[1.1 Informazioni sul progetto 3](#_Toc59124086)

[1.2 Abstract 3](#_Toc59124087)

[1.3 Scopo 3](#_Toc59124088)

[2 Analisi 4](#_Toc59124089)

[2.1 Analisi del dominio 4](#_Toc59124090)

[2.2 Analisi e specifica dei requisiti 4](#_Toc59124091)

[2.3 Use case 7](#_Toc59124092)

[2.4 Pianificazione 1](#_Toc59124093)

[2.5 Analisi dei mezzi 1](#_Toc59124094)

[2.5.1 Software 1](#_Toc59124095)

[2.5.2 Hardware 1](#_Toc59124096)

[3 Progettazione 2](#_Toc59124097)

[3.1 Design dell’architettura del sistema 2](#_Toc59124098)

[3.2 Design delle interfacce 2](#_Toc59124099)

[3.3 Design procedurale 2](#_Toc59124100)

[3.3.1 Server class 3](#_Toc59124101)

[3.3.2 Client class 4](#_Toc59124102)

[3.3.3 Multi class 5](#_Toc59124103)

[3.3.4 Tester class 5](#_Toc59124104)

[4 Implementazione 6](#_Toc59124105)

[5 Test 7](#_Toc59124106)

[5.1 Protocollo di test 7](#_Toc59124107)

[5.2 Risultati test 9](#_Toc59124108)

[5.3 Mancanze/limitazioni conosciute 9](#_Toc59124109)

[6 Consuntivo 1](#_Toc59124110)

[7 Conclusioni 1](#_Toc59124111)

[7.1 Sviluppi futuri 1](#_Toc59124112)

[7.2 Considerazioni personali 1](#_Toc59124113)

[8 Bibliografia 1](#_Toc59124114)

[8.1 Sitografia 1](#_Toc59124115)

[9 Allegati 1](#_Toc59124116)

# Introduzione

## Informazioni sul progetto

Allievi coinvolti nel progetto: Samuel Agustoni

Classe: Informatica 3AC presso la sede Scuola Arti e Mestieri Trevano

Docenti responsabili: Geo Petrini

Data inizio: 03.09.2020  
Data consegna: 17.12.2020

## Abstract

Hai bisogno di testare le prestazioni di trasmissione della tua rete LAN? Questo software potrebbe essere la soluzione che fa per te! Il LAN Speed Tester è un’applicazione portable multipiattaforma, in grado di fornire una statistica molto dettagliata delle velocità di trasmissione con la quale le macchine all’interno della tua rete comunicano. L’obbiettivo di questo progetto è quello di sviluppare un software semplice, leggero e facilmente trasferibile da un dispositivo all’altro, pensato per testare le prestazioni dei collegamenti di una rete LAN. L’applicazione non presenta alcuna interfaccia grafica ma il suo utilizzo è molto intuitivo.

## Scopo

Lo scopo del progetto “LAN Speed Tester” è quello di creare un software facilmente trasferibile e multipiattaforma, in grado di misurare, con una precisione al millisecondo, la velocità con cui una macchina Server e una macchina Client comunicano all’interno di una rete LAN. L’applicazione, al termine dell’esecuzione del programma, dovrà mostrare a schermo, sulla macchina che abbiamo definito come Client, una statistica significativa delle velocità. Contenente il numero di pacchetti mandati, le dimensioni in byte dei vari pacchetti e le medie delle velocità impiegate per ricevere una risposta dal server ad ogni trasmissione. Un software a parer mio molto utile a livello sistemistico in quanto permette al sistemista di tenere sotto controllo il buon funzionamento e l’efficienza della comunicazione nella propria rete LAN.

# Analisi

## Analisi del dominio

È stato richiesto lo sviluppo di un’applicazione multipiattaforma facilmente trasferibile con la funzione di Speed Tester. Ovvero un tester per la velocità di trasmissione all’interno di una rete LAN. Dovrà essere compatibile con la maggior parte dei sistemi operativi Linux/Windows/Unix. Il software non avrà interfaccia grafica per mia scelta, verrà perciò lanciato via terminale da linea di comando.

Concettualmente l’applicazione si comporta in questo modo: una macchina (server) della rete si mette in ascolto su una porta pubblica disponibile, un’altra macchina (client) della stessa rete trasmette al server in ascolto sulla porta precedentemente definita, una o più sequenze di byte. Infine la macchina client produce in output una statistica significativa delle velocità delle varie trasmissioni. In questo modo si possono osservare le prestazioni della rete.

Il software è diviso in due parti, poiché deve avere due ruoli. Una parte per l’utilizzo su client e una parte per l’utilizzo su server. Una volta lanciata l’applicazione sul server, bisogna prima di tutto scegliere il ruolo giusto (S), dopodiché bisogna scegliere la porta sulla quale il server rimane in ascolto, se non viene scelta alcuna porta viene impostata automaticamente la prima disponibile. Quando il server è pronto a ricevere connessioni viene mostrato un avviso che spiega come procedere con la configurazione sul client. Fatto ciò, il software viene lanciato sul client che sceglie il proprio ruolo (C). Manualmente si inseriscono il nome host e la porta sulla quale il server è in ascolto e il numero e le dimensioni dei vari pacchetti di byte che verranno trasmessi al server.

## Analisi e specifica dei requisiti

Mi è stato richiesto da parte del committente, di realizzare un applicativo client. L’applicativo in questione dev’essere un software per testare le prestazioni e le capacità di una rete LAN. Il prodotto deve essere diviso in due parti, una parte utile a configurare una macchina come server e metterla in ascolto su una porta disponibile. Una seconda parte utile per la comunicazione con la macchina server.

|  |  |
| --- | --- |
| **ID: REQ-001** | |
| **Nome** | Il software deve essere portable. |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Deve essere possibile trasferire facilmente il software da una macchina all’altra e non deve lasciare files o impostazioni nel sistema in cui viene eseguito. |

|  |  |
| --- | --- |
| **ID: REQ-002** | |
| **Nome** | Il software deve essere multipiattaforma. |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Deve essere possibile utilizzare il software su qualsiasi sistema che non sia mobile. |

|  |  |
| --- | --- |
| **ID: REQ-003** | |
| **Nome** | L’utente server deve poter scegliere la porta di ascolto del server. |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | La porta scelta dall’utente deve essere una di quelle disponibili. |
| **Sotto requisiti** | |
| **001** | Se l’utente non sceglie una porta il server ne propone una libera. |

|  |  |
| --- | --- |
| **ID: REQ-004** | |
| **Nome** | Una volta scelta la porta, il server deve mostrare il suo stato (se è pronto e in ascolto) e quali sono i parametri che permettono al client di connettersi. |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** |  |

|  |  |
| --- | --- |
| **ID: REQ-005** | |
| **Nome** | Il software non avrà una GUI, verrà lanciato da terminale. |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** |  |

|  |  |
| --- | --- |
| **ID: REQ-006** | |
| **Nome** | L’utente client deve poter scegliere il numero di connessioni da mandare al server. |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | Deve scegliere il numero di pacchetti di dati da mandare e se vanno mandati in sequenza o contemporaneamente. |

|  |  |
| --- | --- |
| **ID: REQ-007** | |
| **Nome** | L’utente client deve poter scegliere la dimensione dei dati per ogni connessione da scambiare con il server. |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** |  |

|  |  |
| --- | --- |
| **ID: REQ-008** | |
| **Nome** | Il client deve produrre in output una statistica significativa dei dati scambiati con il server. |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Per ogni prova viene creata una statistica contenente la media, il massimo e il minimo della velocità di risposta dal server. I tempi vengono misurati al millisecondo. |

|  |  |
| --- | --- |
| **ID: REQ-009** | |
| **Nome** | Il client deve essere multithread. |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | Il client può trasmettere al server più connessioni contemporaneamente. |
|  |  |

|  |  |
| --- | --- |
| **ID: REQ-010** | |
| **Nome** | Il server deve essere multithread. |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | Il server deve poter accettare più di una connessione in entrata contemporaneamente. |

## Use case



Casi d'uso:LAN SpeedTester ‑

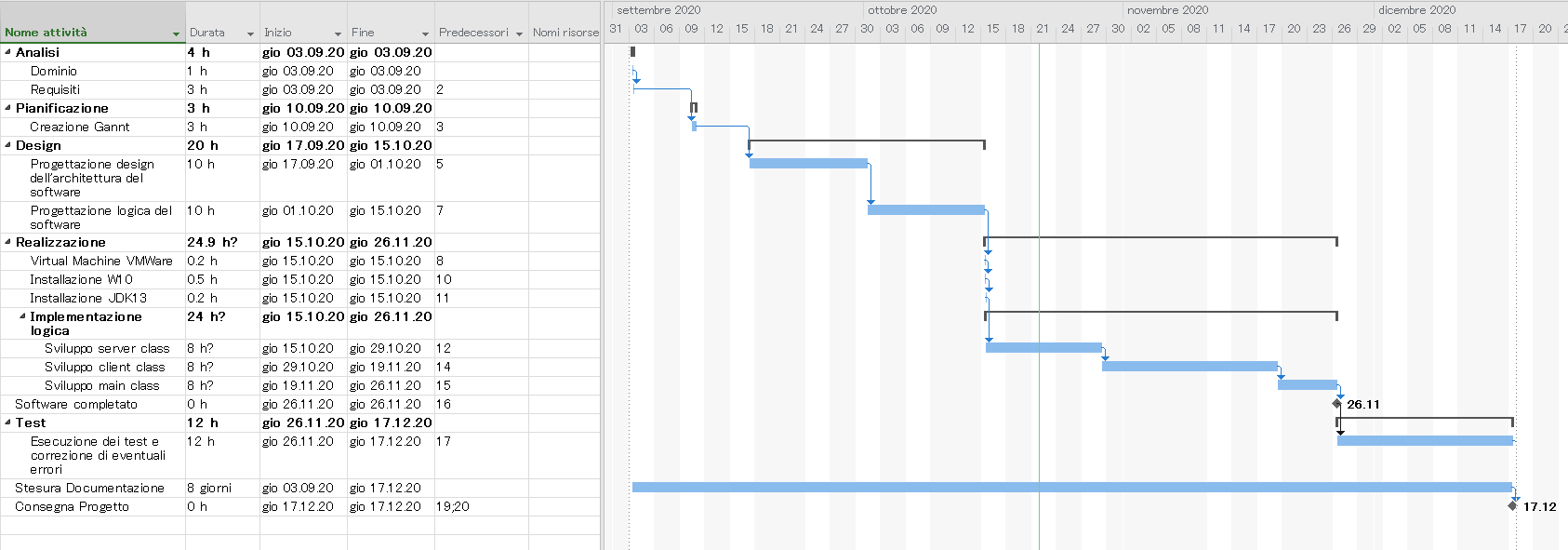
Il sistemista che dovrà occuparsi della misurazione della velocità della propria rete LAN, con il software che ho sviluppato, dovrà procedere nel modo seguente.

Prima di tutto dovrà preoccuparsi di configurare il server, scegliendo una porta tra quelle disponibili per poterlo mettere in ascolto.

Una volta configurato il server, il sistemista dovrà spostarsi su un’altra macchina e configurarla come client, in modo che le macchine possano comunicare. Una volta effettuata la connessione bisognerà decidere il numero di pacchetti da mandare e la loro dimensione. Infine il client mostrerà una statistica significativa delle velocità con cui le macchine hanno comunicato.

## Pianificazione

Ho deciso di dividere questo progetto in cinque fasi ben definite. Prima fra tutte la fase di analisi. Una fase molto importante e decisiva per il successo del progetto in quanto si estrapolano tutti i requisiti e il dominio del mandato. Al secondo posto abbiamo la fase di pianificazione che costituisce la creazione del diagramma di Gannt preventivo che definisce una programmazione iniziale di quelle che saranno le varie fasi e attività del progetto.



Gannt preventivo ‑

A seguito della pianificazione ho inserito una fase di design/progettazione. In questo punto bisogna definire l’infrastruttura dell’applicativo progettandone classi e interfacce. Una volta terminata la progettazione del lavoro si può passare alla realizzazione vera e propria del software. Realizzazione che ha una prima parte di preparazione allo sviluppo, che comprende l’installazione degli applicativi utili allo sviluppo. Infine ho inserito una fase di test per verificare il buon funzionamento del software una volta finito.

## Analisi dei mezzi

### Software

Per realizzare questo progetto ho utilizzato una macchina virtuale con OS Windows 10 attraverso l’ambiente di virtualizzazione *VMWare 14* fornitoci dalla scuola.

All’interno della macchina virtuale ho installato la *JDK 13,* siccome il progetto è stato sviluppato nel linguaggio JAVA. Inoltre per facilitarmi la gestione di esso, ho installato *Apache NetBeans 12.1*, un IDE per la gestione di progetti in diversi linguaggi. Molto comodo quando si ha a che fare con un gran numero di classi. Nonostante non fosse questo il caso, ho deciso di utilizzarlo comunque per comodità personale.

### Hardware

Il progetto è stato sviluppato su un PC fisso scolastico.

Le specifiche hardware sono:

* 16 GB di RAM
* Intel Core i7-7700 con 8 core

# Progettazione

## Design dell’architettura del sistema

Architettura del sistema ‑

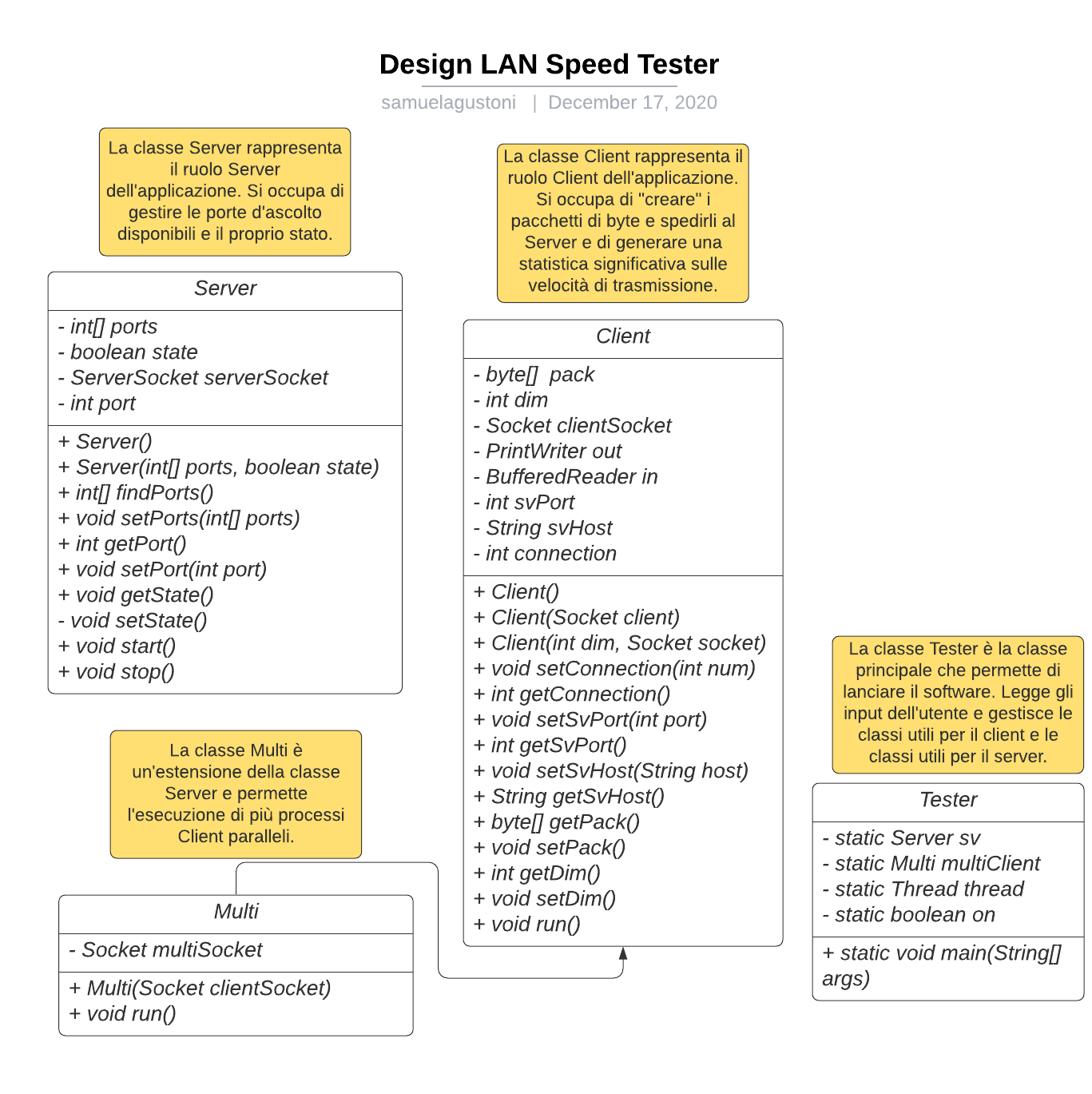
L’architettura concettuale del funzionamento di questo sistema è molto semplice in realtà. Ci sono uno o più server, che si mettono in ascolto su una porta ben precisa e attendono le richieste di connessione da parte dei client. I client si connettono singolarmente o contemporaneamente allo stesso o a differenti server, e mandano informazioni ad essi che gli rispondono in tempo reale. Infine su ogni client ci sarà la statistica della velocità di comunicazione con la quale hanno appena comunicato con il server a cui erano connessi.

## Design delle interfacce

Per questo lavoro potevo decidere se sviluppare un’interfaccia utente grafica, oppure lavorare attraverso la buona e vecchia linea di comando. Siccome non sapevo quanto tempo avrei impiegato per sviluppare completamente la logica, ho deciso di non fare un’interfaccia grafica per avere più tempo a disposizione per pensare alla logica. Infatti non ho fatto una vera e propria progettazione per il design di quello che sarebbe stata la mia CLI. Basandomi sui requisiti ho sviluppato nel corso del progetto un’interfaccia molto semplice e intuitiva.

## Design procedurale

Questo qui sotto è il design definitivo dell’architettura del mio sistema che ho ideato nel corso del progetto. Inizialmente ho sviluppato una versione molto più semplificata, utilizzando una classe in più ma molti meno attributi e metodi. Andando avanti mi sono però accorto che non avevo considerato molti aspetti. Nonostante ciò non ho avuto grandi difficoltà nel modificare l’architettura. In realtà sapevo con certezza che avrei cambiato sicuramente delle cose nel corso del tempo, per questo inizialmente avevo deciso di fare un’architettura piuttosto generale, di modo che a grandi linee sapevo cosa avrei dovuto fare ma per i dettagli avrei avuto carta bianca durante tutta la fase di sviluppo.



Design LANSpeedTester ‑

### 3.3.1 Server class

#### 3.3.1.1 Attributes

* **private int[] ports**: è l’array contenente le porte libere utilizzabili dal server per mettersi in ascolto e ricevere pacchetti di byte.
* **private boolean state**: è lo stato del server, se true significa che il server è pronto e in ascolto, se false non è pronto.
* **private ServerSocket serverSocket**: il socket che il Server crea per mettersi in ascolto sulla rete.
* **private int port**: la porta sulla quale il server si mette in ascolto.

#### 3.3.1.2 Methods

* **public Server()**: permette di istanziare un oggetto Server senza fornirgli alcun attributo.
* **public Server(int[] ports, boolean state)**: permette di istanziare un oggetto Server fornendogli la lista di porte libere e lo stato del server.
* **public int[] findPorts**: cerca le porte disponibili e le inserisce nell'attributo "ports".
* **public void setPorts(int[] ports)**: permette di settare le porte libere.
* **public getPort()**: ritorna la porta sulla quale il server è in ascolto.
* **public void setPort(int port)**: setta la porta sulla quale il server deve mettersi in ascolto.
* **public void getState()**: si occupa di riferire all'utente se il server è pronto e in ascolto.
* **private void setState()**: imposta lo stato del server.
* **public void start()**: crea il socket del Server, si mette in ascolto e attende le richieste dei Client.
* **public void stop()**: chiude il socket del Server.

### 3.3.2 Client class

#### 3.3.2.1 Attributes

* **private byte[] pack**: L'array di byte da spedire al Server. Le sue dimensioni possono essere definite dall'utente.
* **private int dim**: è la dimensione dell'array pack.
* **private Socket clientSocket**: il socket del client per poter mettersi in comunicazione con il server.
* **private PrintWriter out**: il canale di output per mandare dati al server.
* **private BufferedReader in**: il canale di input per ricevere dati dal server.
* **private int svPort**: la porta sulla quale il server è in ascolto.
* **private String svHost**: il nome host del server.
* **private int connection**: il numero di pacchetti di byte da mandare al server.

#### 3.3.2.2 Methods

* **public Client()**: permette di istanziare un oggetto client senza fornirgli alcun attributo.
* **public Client(Socket client)**: permette di istanziare un oggetto client fornendogli l’attributo Socket client.
* **public Client(int dim, Socket client)**: permette di istanziare un oggetto client fornendogli gli attributi dim e client.
* **public void setConnection(int num)**: imposta il numero di connessioni da mandare al server.
* **public int getConnection()**: ritorna il numero di connessioni da mandare al server.
* **public void setSvPort(int port)**: permette di impostare la porta sulla quale il server è in ascolto.
* **public int getSvPort()**: ritorna la porta sulla quale il server è in ascolto.
* **public void setSvHost(String host)**: permette di impostare il nome host del server.
* **public String getSvHost()**: ritorna il nome host del server.
* **public byte[] getPack()**: ritorna il pacchetto di byte da mandare al server.
* **public void setPack()**: imposta il pacchetto di byte da mandare al server.
* **public int getDim()**: ritorna la dimensione dell’array pack.
* **public void setDim()**: imposta la dimensione dell’array pack.
* **public void run()**: permette la comunicazione tra client e server.

### 3.3.3 Multi class

#### 3.3.3.1 Attributes

* **private Socket multiSocket**: il socket che permette il multithreading tra server e client.

#### 3.3.3.2 Methods

* **public Multi(Socket clientSocket)**: permette di istanziare un oggetto Multi fornendo come parametro il Socket del client.
* **public void run()**: è utile per eseguire più processi Client alla volta su di un Server.

### 3.3.4 Tester class

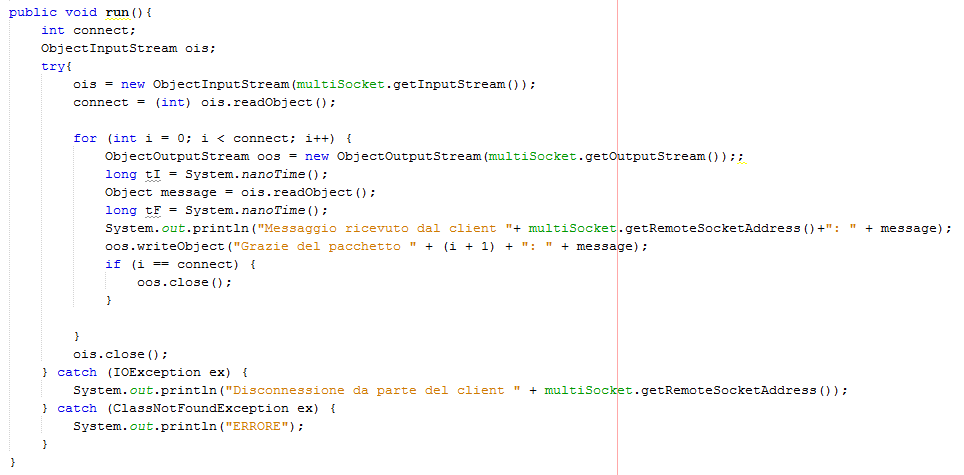
#### 3.3.4.1 Attributes

* **private static Server sv**: l’oggetto Server del software.
* **private static Multi multiClient**: l’oggetto Multi per il multithreading del server.
* **private static Thread thread**: thread per il multithreading.
* **private static boolean on**: la condizione di un while.

#### 3.3.4.2 Methods

* **public static void main(String[] args)**: il metodo main che si occupa della gestione dell’utilizzo dei vari metodi delle varie classi.

# Implementazione

Ho impiegato circa un mese di sviluppo per il completamento del software. È andato tutto abbastanza tranquillo, fino al giorno in cui ho dovuto approcciarmi al multithreading, di cui sinceramente sapevo ben poco. Con la giusta motivazione mi sono informato e documentato e sono riuscito a rendere il Server multithread, ovvero che può accettare più client alla volta contemporaneamente su porte diverse.

Multithread Server ‑

Questo metodo, il metodo run dell’interfaccia Runnable nella classe Multi, che ho scritto, mi ha permesso di rendere il Server multithread. Semplicemente nel metodo run si definisce il comportamento del server nel momento in cui un client si connette. E, grazie all’interfaccia implementata, esso può essere utilizzato nella classe Tester assieme alla classe Thread di Java per l’esecuzione di più processi contemporanei. In pratica ogni volta che un client si collega viene creata una Thread con il proprio metodo run. Una volta che il client ha concluso la comunicazione la Thread viene chiusa. Questo è possibile grazie allo slicing del processore

# Test

## Protocollo di test

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-003 | **Nome:** | L’utente server deve poter scegliere la porta di ascolto del server. |
| **Descrizione:** | La porta scelta dall’utente dev’essere una di quelle disponibili. | | |
| **Prerequisiti:** | La macchina server dev’essere accesa e si deve aver avviato LanSpeedTester selezionando il ruolo server. | | |
| **Procedura:** | 1. Avviare il LanSpeedTester su una macchina.  2. Selezionare il ruolo server.  3. Attendere che venga generata la lista delle porte disponibili.  4. Scegliere se inserire manualmente la porta o farla inserire automaticamente dal server.  5. Nel caso in cui si scelga di inserirla manualmente inserire la porta desiderata. | | |
| **Risultati attesi:** | Se la porta inserita dall’utente è disponibile il server si mette in ascolto correttamente, se l’utente non sceglie una porta il server ne propone una libera. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-004 | **Nome:** | Il software mostra lo stato del server |
| **Descrizione:** | Una volta scelta la porta, il server deve mostrare il suo stato (se è pronto e in ascolto) e quali sono i parametri che permettono al client di connettersi. | | |
| **Prerequisiti:** | La macchina server dev’essere accesa, si deve aver avviato LanSpeedTester selezionando il ruolo server, si deve aver impostato la porta sulla quale metterlo in ascolto. | | |
| **Procedura:** | 1. Avviare il LanSpeedTester su una macchina.  2. Selezionare il ruolo server.  3. Attendere che venga generata la lista delle porte disponibili.  4. Scegliere se inserire manualmente la porta o farla inserire automaticamente dal server.  5. Nel caso in cui si scelga di inserirla manualmente inserire la porta desiderata. | | |
| **Risultati attesi:** | Eseguiti questi cinque semplici passaggi l’applicazione dovrebbe mostrare lo stato del server e una spiegazione su come procedure con la configurazione sul client. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-006/007 | **Nome:** | L’utente client deve poter scegliere il numero di connessioni da mandare al server e la loro dimensione |
| **Descrizione:** | L’utente client deve scegliere il numero di pacchetti di dati da mandare e la loro dimensione. | | |
| **Prerequisiti:** | La macchina server dev’essere accesa e dev’essere pronta e in ascolto su una porta ben precisa. | | |
| **Procedura:** | 1. Avviare il LanSpeedTester su una macchina.  2. Selezionare il ruolo client.  3. Inserire il nome host della macchina server.  4. Inserire la porta sulla quale è in ascolto.  5. Inserire numero di pacchetti da mandare e la dimensione. | | |
| **Risultati attesi:** | I pacchetti vengono mandate al server e una volta raggiunto rispediti indietro. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-008 | **Nome:** | Il client deve produrre in output una statistica significativa dei dati scambiati con il server. |
| **Descrizione:** | Il client deve produrre in output una statistica significativa dei dati scambiati con il server. | | |
| **Prerequisiti:** | La macchina server dev’essere accesa e dev’essere pronta e in ascolto su una porta ben precisa. | | |
| **Procedura:** | 1. Avviare il LanSpeedTester su una macchina.  2. Selezionare il ruolo client.  3. Inserire il nome host della macchina server.  4. Inserire la porta sulla quale è in ascolto.  5. Inserire numero di pacchetti da mandare e la dimensione. | | |
| **Risultati attesi:** | I pacchetti vengono mandate al server e una volta raggiunto rispediti indietro, sul client viene prodotta una statistica contenente media minimo e massimo delle velocità. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-009 | **Nome:** | Il client dev’essere multithread. |
| **Descrizione:** | Il client può connettersi a più server contemporaneamente. | | |
| **Prerequisiti:** |  | | |
| **Procedura:** | Questo requisito non è stato sviluppato e quindi soddisfatto. | | |
| **Risultati attesi:** | Il client può connettersi a più server contemporaneamente. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-010 | **Nome:** | Il server dev’essere multithread. |
| **Descrizione:** | Il server può accettare la connessione di più client contemporaneamente. | | |
| **Prerequisiti:** |  | | |
| **Procedura:** | 1. Avviare il LanSpeedTester su una macchina.  2. Selezionare il ruolo server.  3. Attendere che venga generata la lista delle porte disponibili.  4. Scegliere se inserire manualmente la porta o farla inserire automaticamente dal server.  5. Nel caso in cui si scelga di inserirla manualmente inserire la porta desiderata.  6. Avviare il LanSpeedTester su più macchine  7. Selezionare il ruolo client.  8. Inserire il nome host della macchina server.  9. Inserire la porta sulla quale è in ascolto. | | |
| **Risultati attesi:** | Il client può connettersi a più server contemporaneamente. | | |

## Risultati test

Tutti i test hanno avuto successo, meno il test sul requisito REQ-010. Questo perché non ho implementato la parte che permette al client di connettersi a più server.

## Mancanze/limitazioni conosciute

Potevo gestire meglio la utilizzabilità della parte client in quanto non è possibile uscire dal programma fino alla connessione con un server. In più dopo che un client ha finito di comunicare con il server il software sul client si interrompe e se si vuole ripetere una ulteriore comunicazione bisogna riavviarlo. Sicuramente molte cose potevo gestirle meglio e renderle più gradevoli e user friendly ma lo scopo del progetto posso dire di averlo praticamente raggiunto. In più un’applicazione che gira via linea di comando è già poco user friendly di suo.

# 

# Consuntivo

Gannt Consuntivo ‑

Ecco il diagramma di Gannt consuntivo del mio progetto. Sono stato abbastanza fedele a quello che avevo pianificato, infatti fino all’implementazione non ci sono state differenze. Quando ho iniziato la fase di sviluppo mi sono accorto che quello che avevo pianificato non aveva senso. Infatti era impossibile che sviluppassi le classi separatamente. Così le ho posizionate parallelamente. Inoltre ho aggiunto una classe, per permettere e separare dal resto il multithreading.

# Conclusioni

Per tutta la durata di questo percorso ho cercato di fare meno domande possibili. Il motivo è semplice, volevo vedere fino a dove sarei arrivato grazie alle mie conoscenze e alla mia forza di volontà. Sono abbastanza soddisfatto del risultato.

## Sviluppi futuri

Sicuramente in futuro per conto mio continuerò a darci un occhio. Sarebbe interessante aggiungere un’interfaccia grafica e migliorare la utilizzabilità generale.

## Considerazioni personali

Grazie a questo progetto ho ampliato le mie conoscenze nel linguaggio JAVA, imparando ad utilizzare i socket per le comunicazioni client server e le thread per l’esecuzione di più processi in parallelo.

# Bibliografia

## Sitografia

* https://www.baeldung.com/a-guide-to-java-sockets, *A Guide to Java Sockets,* 15-10-2020.
* <https://docs.oracle.com/en/java/javase/13/docs/api/index.html>, *Java doc 13, 15-10-2020 fino a 03.12.2020.*
* <https://www.html.it/articoli/thread-in-java/>, *Thread in Java: tutorial ed esempi d’uso, 19-11-2020*

# Allegati

* Diari di lavoro
* Cartella “dist” contenente i componenti utili all’utilizzo del programma. Come l’eseguibile, l’installer di JDK13 e il manuale utente.
* Cartella “Project” contenente tutto il progetto e i codici sorgente nella sottocartella “src”.
* Cartella “doc” contenente tutti i documenti di pianificazione da me creati nel corso del lavoro.
* Abstract