Prova in itinere N. 1 di Programmazione ad Oggetti
Corso di laurea in Ingegneria Informatica, a.a. 2019/2020
Università degli Studi di Salerno
6 novembre 2019

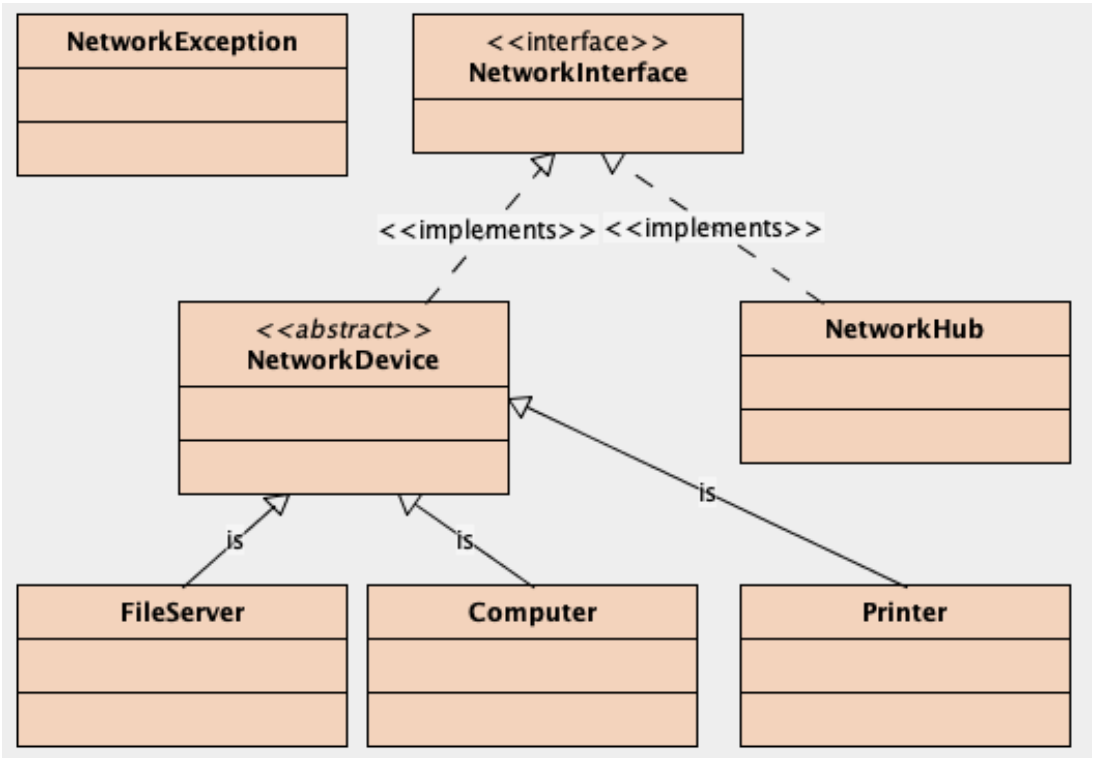
Si chiede di implementare il seguente insieme di classi ed interfacce necessarie alla realizzazione di applicazioni per la gestione di connessioni di rete:

Interface	**Classi**
NetworkInterface	NetworkDevice
	Computer
	FileServer
	Printer
	NetworkHub
	NetworkException

Le suddette classi ed interfacce devono essere inserite tutte nel seguente package:

oop2019.prova1.gruppoXX

dove XX deve essere sostituito con il numero del proprio gruppo espresso su due cifre (ad es. il gruppo 4 userà il package oop2019.prova1.gruppo04).
Le relazioni tra le classi ed interfacce da implementare sono indicate nel seguente diagramma delle classi (per facilitare la leggibilità nella figura sono visualizzate esclusivamente le relazioni di tipo "*is a*").



Al candidato è fornita la classe **oop2019.prova1.test.Main** per effettuare il test del codice scritto. Si consulti l'ultima pagina del presente documento per un esempio di esecuzione di tale classe in caso di corretta implementazione di tutte le classi ed interfacce.

Descrizione delle classi da implementare

NetworkInterface

E' una interface pubblica.
Rappresenta un'interfaccia di rete che può essere usata per scambiare messaggi con altri dispositivi. Rende disponibili i seguenti due metodi:

- **`public void accept(NetworkInterface sourceInterface, int sourceAddress, int destAddress, String message)`** – *Accetta un messaggio proveniente da un'altra interfaccia di rete*
  - **`sourceInterface`** - l'interfaccia di rete da cui proviene il messaggio. Nota: sourceInterface potrebbe non essere l'interfaccia del mittente del messaggio, perché tra il mittente e l'interfaccia corrente potrebbero esserci altri dispositivi intermedi.
  - **`sourceAddress`** - l'indirizzo del mittente del messaggio
  - **`destAddress`** - l'indirizzo del destinatario del messaggio
  - **`message`** - il messaggio trasmesso

- **`public void connect(NetworkInterface other)`** - *Connette questa interfaccia di rete con un'altra interfaccia. A seconda del tipo di dispositivo, la connessione può essere uno a uno, oppure uno a molti.*
  - **`other`** - l'interfaccia di rete che viene connessa all'interfaccia corrente

NetworkHub

E' una classe pubblica che implementa NetworkInterface.
NetworkHub rappresenta un hub di rete, che può essere utilizzato per connettere altri dispositivi, tra cui anche altri hub. Un hub mantiene un insieme di connessioni a interfacce di rete; quando riceve un messaggio, lo inoltra a tutte le interfacce tranne a quella da cui il messaggio è stato ricevuto. Un hub ha un identificativo univoco che è una stringa, passato al costruttore. Due hub si considerano uguali se hanno lo stesso identificativo.
La classe rende disponibili i seguenti metodi:

- **`public NetworkHub(String id)`** – *Costruttore*
  - **`id`** - Identificativo univoco dell'hub

- **`public void accept(NetworkInterface sourceInterface, int sourceAddress, int destAddress, String message)`** – *Accetta un messaggio da un'interfaccia di rete. Il messaggio viene inoltrato a tutte le interfacce di rete connesse, tranne quella da cui proviene il messaggio. Inoltre, stampa a video le informazioni sul messaggio ricevuto. Nota: per le interfacce di rete connesse, il messaggio inoltrato avrà come sourceInterface l'hub, mentre sarà preservato il sourceAddress della NetworkInterface che ha originato il messaggio. La stampa a video dovrà includere le informazioni relative al NetworkHub corrente, al* sourceAddress, alla sourceInterface, al destAddress, al message, *rispettando il formato dell'esempio seguente:*

```
Hub H07P01: da 1 via Device 1: Computer Notebook Pasquale per 4: Accesso a hello.docx
```

- **`sourceInterface`** - l'interfaccia di rete da cui proviene il messaggio. Nota: sourceInterface potrebbe non essere l'interfaccia del mittente del messaggio, perché tra il mittente e l'interfaccia corrente potrebbero esserci altri dispositivi intermedi.
- **`sourceAddress`** - l'indirizzo del mittente del messaggio
- **`destAddress`** - l'indirizzo del destinatario del messaggio
- **`message`** - il messaggio trasmesso

- **`public void connect(NetworkInterface other)`** - *Aggiunge una connessione all'hub*
  - **`other`** - l'interfaccia di rete da connettere

- **`public boolean equals(Object other)`** - *Verifica se l'hub è uguale a un altro oggetto. Restituisce true se l'altro oggetto non è null, se è della stessa classe dell'oggetto corrente e se hanno lo stesso identificativo.*
  - **`other`** - l'altro oggetto

- **`public String getId()`** – *Restituisce l'identificativo dell'hub*

- **`public int hashCode()`** - *Calcola l'hash code dell'hub. L'hash code dipende solo dall'identificativo*.

- **`public void printConnections()`** - *Stampa a video l'elenco delle connessioni, rispettando il formato dell'esempio seguente:*

```
Connessioni di: Hub H07P01
  Device 1: Computer Notebook Pasquale
  Device 2: Printer
  Hub H42P00
```

- **`public String toString()`** - *Rappresenta l'hub con una stringa, nel formato: "Hub id", dove id è l'identificativo dell'hub*

NetworkDevice

E' una classe pubblica astratta ed implementa NetworkInterface.
NetworkDevice rappresenta un dispositivo (computer o periferica) che può essere connesso alla rete. Ha un indirizzo univoco (di tipo int) che viene assegnato al momento della creazione e non può essere più modificato. Un NetworkDevice ha un attributo connection del tipo NetworkInterface che rappresenta un'eventuale altra interfaccia di rete connessa a questo NetworkDevice; se connection==null, allora questo NetworkDevice risulta non connesso in rete.
La classe rende disponibili i seguenti metodi:

- **`public NetworkDevice()`** – *Costruttore. Crea il NetworkDevice, assegnandogli un indirizzo univoco. All'atto della creazione il NetworkDevice non è connesso in rete.*

- **`public void accept(NetworkInterface sourceInterface, int sourceAddress, int destAddress, String message)`** – *Accetta un messaggio dalla rete. Se il messaggio non è destinato all'oggetto corrente, lo ignora; altrimenti lo elabora chiamando il metodo 'process'.*

    - **`sourceInterface`** - l'interfaccia di rete da cui proviene il messaggio. Nota: sourceInterface potrebbe non essere l'interfaccia del mittente del messaggio, perché tra il mittente e l'interfaccia corrente potrebbero esserci altri dispositivi intermedi.
    - **`sourceAddress`** - l'indirizzo del mittente del messaggio
    - **`destAddress`** - l'indirizzo del destinatario del messaggio
    - **`message`** - il messaggio trasmesso

- **`public void connect(NetworkInterface other)`** - *Connette un'altra interfaccia di rete. Se l'oggetto era già connesso a un'interfaccia di rete, la nuova connessione sostituisce la precedente.*
    - **`other`** - l'interfaccia di rete che viene connessa all'interfaccia corrente

- **`public boolean equals(Object other)`** - *Verifica l'oggetto è uguale a un altro oggetto. Due dispositivi si considerano uguali se sono della stessa classe e hanno lo stesso indirizzo. In particolare, restituisce true se l'altro oggetto non è null, è della stessa classe e ha lo stesso indirizzo, false in tutti gli altri casi.*
    - **`other`** - l'altro oggetto

- **`public int getAddress()`** - *Restituisce l'indirizzo dell'oggetto*.

- **`public NetworkInterface getConnection()`** - *Restituisce l'interfaccia di rete connessa al dispositivo.*

- **`public int hashCode()`** - *Calcola e restituisce l'hash code dell'oggetto corrente. L'hash code dipende solo dall'indirizzo.*

- **`??? abstract void process(int sourceAddress, String message)`** - *Elabora un messaggio ricevuto dalla rete. Il metodo è visibile solo nelle classi derivate e nello stesso package (sostituire ??? con il modificatore d'accesso adeguato)*.
    - **`sourceAddress`** - indirizzo del mittente del messaggio
    - **`message`** - contenuto del messaggio

- **`public String toString()`** - *Restituisce la rappresentazione dell'oggetto come stringa, usando il formato: "Device addr", dove 'addr' è l'indirizzo del dispositivo.*

NetworkException

Eccezione non controllata che viene sollevata in caso di errori relativi alla rete.

- **`public NetworkException()`** - *Costruttore di default.*

- **`public NetworkException(String errorMessage)`** - *Costruisce una NetworkException con un messaggio di errore.*
    - **`errorMessage`** - il messaggio di errore

Computer

E’ una classe pubblica ed estende NetworkDevice.
Computer rappresenta un personal computer connesso alla rete. Un computer ha un nome specificato attraverso il costruttore, e può eseguire alcune operazioni come inviare richieste a una stampante remota o a un disco remoto.
La classe rende disponibili i seguenti metodi:

- **`public Computer(String name)`** - *Costruttore.*
  - **`name`** - il nome del computer

- **`public String getName()`** - *Accessor dell'attributo name. Restituisce il nome del computer.*

- **`??? void process(int sourceAddress, String message)`** - *Elabora un messaggio ricevuto dalla rete, stampandone il contenuto a video. (Mantiene lo stesso modificatore di accesso della superclasse). La stampa a video deve essere congruente con il seguente esempio:*
  `Device 1: Computer Notebook Pasquale ha ricevuto da 4: Dati risposta: Accesso a hello.docx`
  - **`sourceAddress`** - indirizzo del mittente del messaggio
  - **`message`** - contenuto del messaggio

- **`public void remoteFileAccessRequest(int serverAddress, String fileName)`** - *Invia una richiesta di accesso a un file remoto; la richiesta può essere inviata solo se il dispositivo è connesso, altrimenti lancia una NetworkException.*
  - **`serverAddress`** - l'indirizzo del file server remoto
  - **`fileName`** - il nome del file da usare

- **`public void remotePrintRequest(int printerAddress, String fileName)`** - *Invia una richiesta di stampa a una stampante remota; la richiesta può essere inviata solo se il dispositivo è connesso, altrimenti lancia una NetworkException.*
  - **`printerAddress`** - l'indirizzo della stampante remota
  - **`fileName`** - il nome del file da stampare

- **`public String toString()`** - *Restituisce una stringa che rappresenta il computer, nel formato: "Device addr: Computer name" dove addr è l'indirizzo e name è il nome del computer.*

FileServer

E’ una classe pubblica ed estende NetworkDevice.
FileServer rappresenta un file server che può essere connesso alla rete.
La classe rende disponibili i seguenti metodi:

- **`??? void process(int sourceAddress, String message)`** - *Elabora una richiesta di accesso al file, stampando un messaggio sullo schermo come nell’esempio seguente:*
  `Device 4: File Server: Su richiesta di 1: Accesso a hello.docx`
  e inviando una risposta al mittente della richiesta contenente il messaggio nel formato “Dati risposta: message” dove message è il messaggio ricevuto come parametro come da esempio:
  `Dati risposta: Accesso a hello.docx`
  Se non è connesso a un'interfaccia di rete, lancia un'eccezione. (Mantiene lo stesso modificatore di accesso della superclasse)
  - **`sourceAddress`** - indirizzo del mittente del richiesta
  - **`message`** - il testo della richiesta

- **`public String toString()`** - *Restituisce una stringa che rappresenta il file server, nel formato: "Device addr: File Server" dove addr è l'indirizzo del dispositivo.*

Printer

E’ una classe pubblica ed estende NetworkDevice.
Printer rappresenta una stampante di rete.
La classe rende disponibili i seguenti metodi:

- **`??? void process(int sourceAddress, String message)`** - *Elabora una richiesta di stampa remota, stampando un messaggio sullo schermo come nell’esempio seguente:*
  `Device 2: Printer: Su richiesta di 1: Stampa di hello.docx`
  (Mantiene lo stesso modificatore di accesso della superclasse)
  - **`sourceAddress`** - indirizzo del mittente del richiesta
  - **`message`** - il testo della richiesta

- **`public String toString()`** - *Restituisce una stringa che rappresenta la stampante, nel formato: "Device addr: Printer" dove addr è l'indirizzo del dispositivo.*

Esempio di esecuzione

Nel riquadro seguente è riportato l'output a video che dovrebbe essere prodotto dall'esecuzione della classe **oop2019.prova1.test.Main** nel caso di corretta implementazione dell'elaborato:

```
CONNESSIONI DI RETE REALIZZATE
Connessioni di: Hub H07P01
  Device 1: Computer Notebook Pasquale
  Device 2: Printer
  Hub H42P00
Connessioni di: Hub H42P00
  Device 3: Computer Desktop Gennaro
  Device 4: File Server
  Hub H07P01

------------------------------------------------
Hub H07P01: da 1 via Device 1: Computer Notebook Pasquale per 4: Accesso a hello.docx
Hub H42P00: da 1 via Hub H07P01 per 4: Accesso a hello.docx
Device 4: File Server: Su richiesta di 1: Accesso a hello.docx
Hub H42P00: da 4 via Device 4: File Server per 1: Dati risposta: Accesso a hello.docx
Hub H07P01: da 4 via Hub H42P00 per 1: Dati risposta: Accesso a hello.docx
Device 1: Computer Notebook Pasquale ha ricevuto da 4: Dati risposta: Accesso a hello.docx
------------------------------------------------
Hub H07P01: da 1 via Device 1: Computer Notebook Pasquale per 2: Stampa di hello.docx
Device 2: Printer: Su richiesta di 1: Stampa di hello.docx
Hub H42P00: da 1 via Hub H07P01 per 2: Stampa di hello.docx
------------------------------------------------
Hub H42P00: da 3 via Device 3: Computer Desktop Gennaro per 4: Accesso a budget.xlsx
Device 4: File Server: Su richiesta di 3: Accesso a budget.xlsx
Hub H42P00: da 4 via Device 4: File Server per 3: Dati risposta: Accesso a budget.xlsx
Device 3: Computer Desktop Gennaro ha ricevuto da 4: Dati risposta: Accesso a budget.xlsx
Hub H07P01: da 4 via Hub H42P00 per 3: Dati risposta: Accesso a budget.xlsx
Hub H07P01: da 3 via Hub H42P00 per 4: Accesso a budget.xlsx
------------------------------------------------
Hub H42P00: da 3 via Device 3: Computer Desktop Gennaro per 2: Stampa di budget.xlsx
Hub H07P01: da 3 via Hub H42P00 per 2: Stampa di budget.xlsx
Device 2: Printer: Su richiesta di 3: Stampa di budget.xlsx
```