PROVA IN ITINERE N. 1 DI PROGRAMMAZIONE AD OGGETTI
CORSO DI LAUREA IN INGEGNERIA INFORMATICA, A.A. 2020/2021

UNIVERSITÀ DEGLI STUDI DI SALERNO
6 NOVEMBRE 2020

Si chiede di implementare il seguente insieme di classi ed interfacce necessarie alla realizzazione di un'applicazione per la gestione di sistemi di elaborazione:

Interface (fornite)	Enum (fornite)	Classi
`Filterable`	**`CPUType`**	**`Device (abstract)`**
`DeviceFilter`	**`MobileCPUType`**	**`Notebook`**
	`NBScreenType`	**`Smartphone`**
		`DeviceStore`
		`NotebookFilter`
		`HighStorageDeviceFilter`
		`DeviceReleaseYearComparator`
		`DeviceInsertionException`

Le classi da implementare devono essere inserite tutte nel seguente package:

`it.unisa.diem.oop.developed.groupxx`

dove XX deve essere sostituito con il numero del proprio gruppo espresso su due cifre.

Le relazioni tra le classi ed interfacce da implementare sono indicate nel diagramma delle classi in Fig.1 (per facilitare la leggibilità nella figura sono visualizzate prevalentemente le relazioni di tipo "*is a*").

Al candidato sono fornite le classi presenti nel package **`it.unisa.diem.oop.test`** per effettuare il test del codice scritto. Si consulti l'ultima pagina del presente documento per un esempio di esecuzione in caso di corretta implementazione di tutte le classi ed interfacce.

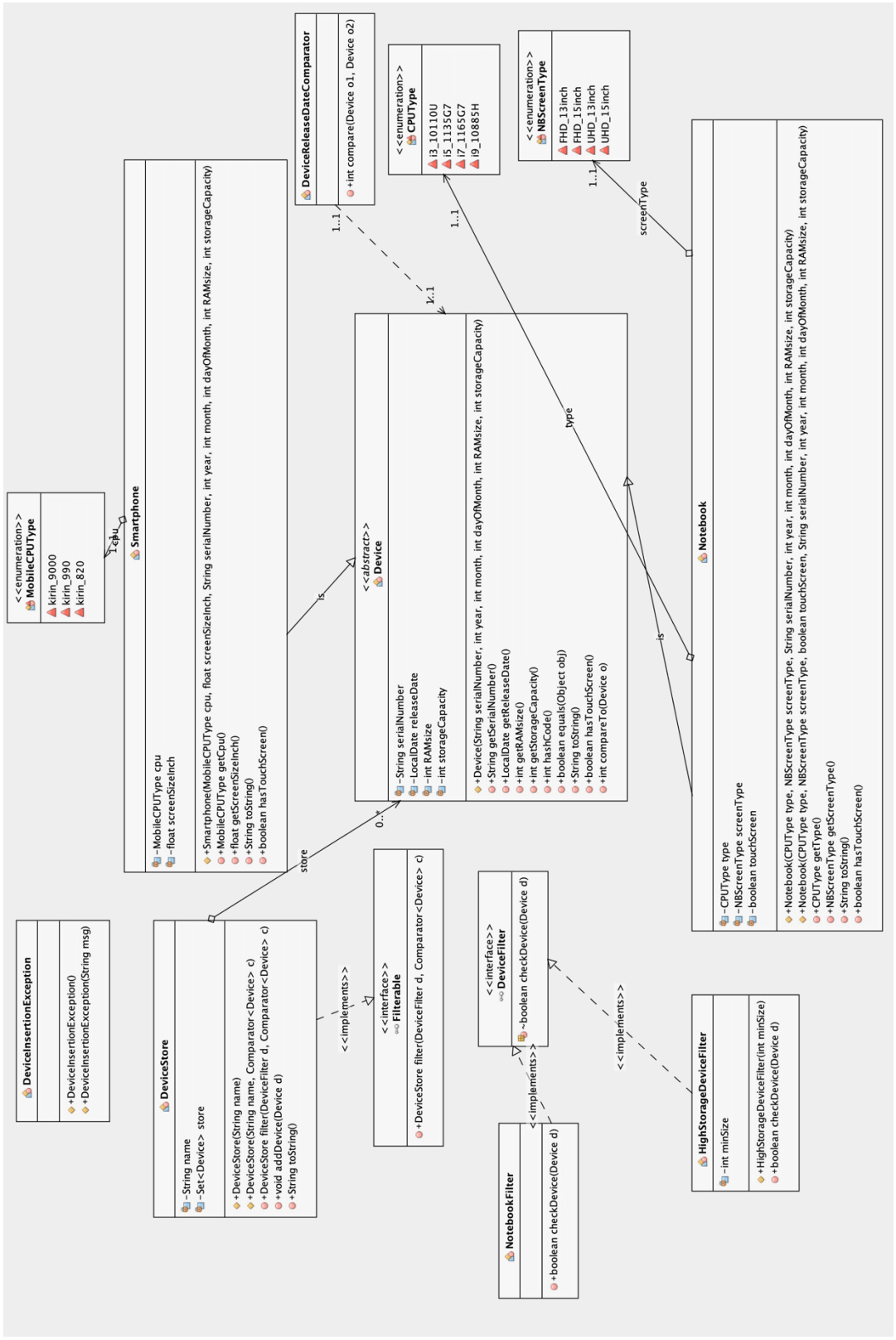

FIGURA 1 DIAGRAMMA DELLE CLASSI

DESCRIZIONE DELLE CLASSI DA IMPLEMENTARE

DEVICE

E' una classe astratta che definisce ed implementa l'entità **`Device`**, un sistema di elaborazione generico. Tiene traccia, mediante attributi in sola lettura, del numero seriale (**`String`**), della data di rilascio (**`LocalDate`**), della dimensione della memoria RAM in Gigabyte (**`int`**) e della dimensione della memoria di massa in GB (**`int`**). La classe implementa il comportamento specificato dall'interface **`Comparable<Device>`** definita dalle API di Java (**`java.lang.Comparable<T>`**). Due device si considerano uguali se hanno lo stesso numero di serie. La relazione d'ordine naturale è definita rispetto all'ordine lessicografico del numero di serie.

Gli attributi vengono inizializzati mediante opportuni parametri passati al costruttore, come specificato nel seguito.

La classe rende disponibili i seguenti metodi:

- **`public Device(String serialNumber, int year, int month, int dayOfMonth, int RAMsize, int storageCapacity)`** - costruttore
  - **`serialNumber`** – numero seriale del dispositivo
  - **`year`** – anno di rilascio
  - **`month`** – mese di rilascio
  - **`dayofMonth`** – giorno di rilascio
  - **`RAMsize`** – dimensione memoria RAM
  - **`storageCapacity`** – dimensione memoria di massa

- **`public boolean equals (Object other)`** - Verifica se il device è uguale ad 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 numero di serie (case insensitive).
  - **`other`** - l'altro oggetto

- **`public int hashCode()`** - Calcola l'hash code del device. L'hash code dipende solo dal numero di serie.

- **`public int compareTo(Device o)`** - Il metodo compareTo definisce la relazione d'ordine naturale sul tipo Device relativamente al numero seriale (ordine lessicografico) case *insensitive*; tale relazione deve essere preservata per tutte le eventuali classi derivate da Device facendo il modo che il metodo non possa essere sovrascritto nelle classi derivate.

- **`public String getSerialNumber()`** – restituisce il numero seriale

- **`public LocalDate getReleaseDate()`** – restituisce la data di rilascio

- **`public int getRAMsize()`** – restituisce la dimensione della memoria RAM

- **`public int getStorageCapacity()`** – restituisce la dimensione della memoria di massa

- **`public abstract boolean hasTouchScreen()`** – è un metodo astratto che tiene traccia della disponibilità di uno schermo tattile

- **`public String toString()`** - Rappresenta il device con una stringa, come riportato di seguito (in corsivo sono riportati i valori dei rispettivi attributi):

```
serialNumber= CY2WPN23
releaseDate= 2018-1-1
RAMsize= 8GB
storageCapacity= 256GB
```

NOTEBOOK

E' una classe pubblica che concretizza la classe **`Device`**. Rappresenta il dispositivo Notebook, aggiungendo le seguenti proprietà in sola lettura: un modello di CPU (tipo di dato definito dall'enumerazione **`CPUType`** fornita) ed una tipologia di schermo (tipo di dato definito dall'enumerazione **`NBScreenType`** fornita). Può inoltre essere dotato di schermo tattile.

Gli attributi vengono inizializzati mediante opportuni parametri passati al costruttore, come specificato nel seguito.

La classe rende disponibili i seguenti metodi:

- **`public Notebook(CPUType type, NBScreenType screenType, String serialNumber, int year, int month, int dayOfMonth, int RAMsize, int storageCapacity)`** – costruttore che può essere usato per notebook privi di schermo tattile

- **`public Notebook(CPUType type, NBScreenType screenType, boolean touchScreen, String serialNumber, int year, int month, int dayOfMonth, int RAMsize, int storageCapacity)`**- costruttore

- **`public CPUType getType()`** – restituisce il tipo di CPU presente

- **`public NBScreenType getScreenType()`** – restituisce il tipo di schermo presente

- **`public String toString()`** - Rappresenta il notebook con una stringa, come riportato nei due casi seguenti, dove il primo si riferisce ad un dispositivo dotato di schermo tattile. (In corsivo sono riportati i valori dei rispettivi attributi):

```
Notebook
serialNumber= CY2WPN23
releaseDate= 2018-1-1
RAMsize= 8GB
storageCapacity= 256GB
CPU=i5_1135G7
screenType=FHD_15inch
touchScreen available

Notebook
serialNumber= CY2WWN23
releaseDate= 2017-1-1
RAMsize= 8GB
storageCapacity= 128GB
CPU=i3_10110U
screenType=FHD_13inch
```

- **`public boolean hasTouchScreen()`** – restituisce **`true`** se è presente uno schermo tattile

SMARTPHONE

E' una classe pubblica che concretizza la classe Device. Rappresenta il dispositivo Smartphone, aggiungendo le seguenti proprietà in sola lettura: un modello di CPU per dispositivi mobile (tipo di dato definito dall'enumerazione **`MobileCPUType`**), dimensione della diagonale dello schermo in pollici (**`float`**). Uno smartphone è sempre dotato di schermo tattile.

Gli attributi vengono inizializzati mediante opportuni parametri passati al costruttore, come specificato nel seguito.

La classe rende disponibili i seguenti metodi:

- **`public Smartphone(MobileCPUType cpu, float screenSizeInch, String serialNumber, int year, int month, int dayOfMonth, int RAMsize, int storageCapacity)`** – costruttore
- **`public MobileCPUType getCpu()`** - restituisce il tipo di CPU
- **`public boolean hasTouchScreen()`** – restituisce la disponibilità dello schermo tattile.
- **`public float getScreenSizeInch()`** – restituisce la dimensione della diagonale dello schermo in pollici
- **`public String toString()`** - Rappresenta lo smartphone con una stringa, come riportato nel caso seguente. (In corsivo sono riportati i valori dei rispettivi attributi)

```
Smartphone
serialNumber= DWW2HJ39
releaseDate= 2019-1-1
RAMsize= 4GB
storageCapacity= 64GB
CPU= kirin_9000
screenSizeInch= 6.7
```

NOTEBOOKFILTER

La classe NotebookFilter implementa l'interfaccia **`DeviceFilter`** (già fornita) attraverso il metodo **`boolean checkDevice(Device d)`**. Nello specifico, il metodo **`checkDevice(d)`** implementato dalla classe NotebookFilter restituisce **`true`** se d è un'istanza della classe **`Notebook`**, **`false`** altrimenti.

HIGHSTORAGEDEVICEFILTER

La classe **`HighStorageDeviceFilter`** implementa l'interfaccia **`DeviceFilter`** (già fornita) attraverso il metodo boolean **`checkDevice(Device d)`**. Il metodo checkDevice implementato dalla classe HighStorageDeviceFilter restituisce **`true`** se la dimensione della memoria di massa è almeno pari al valore specificato attraverso il costruttore del filtro HighStorageDeviceFilter, **`false`** altrimenti.

L'inizializzazione del filtro in oggetto è realizzata mediante l'invocazione del seguente costruttore:

- **`public HighStorageDeviceFilter(int minSize)`**
  - **`minSize`** – dimensione minima della memoria di massa per il device da considerare

DeviceReleaseDateComparator

La classe **`DeviceReleaseDateComparator`** implementa l'interfaccia **`java.util.Comparator<T>`**, dove T è il tipo dell'oggetto da confrontare, che nel caso specifico è un Device.

L'obiettivo di questo comparatore è fornire un criterio di ordinamento per i Device (diverso da quello naturale) basato sulla data di rilascio in maniera crescente. Importante: si noti che per due dispositivi aventi la stessa data di rilascio va preservata la relazione d'ordine dei Device basata sul numero di serie.

La classe rende disponibile il seguente metodo:

- **`public int compare(Device o1, Device o2)`**

DeviceStore

La classe **`DeviceStore`** implementa l'interfaccia **`Filterable`** (già fornita) e definisce l'entità magazzino (store) come un insieme di dispositivi di elaborazione (Device). La struttura dati scelta per l'implementazione dello store dovrà consentire la visualizzazione degli elementi secondo la relazione d'ordine naturale prevista per i Device o secondo una relazione d'ordine specificata da un comparatore passato come parametro al costruttore.

La classe deve prevedere una proprietà **`name`** (String) che rappresenta il nome dello store, passato come parametro al costruttore.

La classe rende disponibili i seguenti metodi:

- **`public DeviceStore(String name)`** – costruttore

- **`public DeviceStore(String name, Comparator<Device> c)`** – costruttore che consente di imporre una relazione d'ordine diversa da quella naturale specificata mediante un comparatore **`c`**.

- **`public void addDevice(Device d)`** – consente di aggiungere un dispositivo allo store. Nel caso in cui il dispositivo sia già presente, solleva un'eccezione non controllata di tipo **`DeviceInsertionException`**.

- **`public String toString()-`** Rappresenta lo store con una stringa, come riportato nel caso seguente. Si noti che "MyStore" rappresenta il nome dello store.

```
MyStore contains 6 items.
Printing:
*****
Notebook
serialNumber= CY2WPN23
releaseDate= 2018-1-1
RAMsize= 8GB
storageCapacity= 256GB
CPU=i5_1135G7
screenType=FHD_15inch
touchScreen available
```

```
*****
Notebook
serialNumber= CY2WWN23
releaseDate= 2017-1-1
RAMsize= 8GB
storageCapacity= 128GB
CPU=i3_10110U
screenType=FHD_13inch
*****
```

Continua…

- **`public DeviceStore filter(DeviceFilter d, Comparator<Device> c)`** - consente di ottenere un nuovo store costituito da un sottoinsieme degli elementi presenti nello store di partenza, selezionati in accordo al filtro specificato dal parametro **`d`** e con una relazione d'ordine imposta dal comparatore **`c`**. Nel caso in cui **`c==null`** viene preservata la relazione d'ordine naturale degli elementi.

DeviceInsertionException

Eccezione non controllata che viene sollevata in caso di errori relativi all'inserimento di un dispositivo già presente.

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

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

  - **`errorMessage`** - il messaggio di errore

TestStore.java

```
MyStore contains 6 items.
Printing:
*****
Notebook
serialNumber= CY2WPN23
releaseDate= 2018-07-03
RAMsize= 8GB
storageCapacity= 256GB
CPU=i5_1135G7
screenType=FHD_15inch
touchScreen available
*****
Notebook
serialNumber= CY2WWN23
releaseDate= 2017-06-30
RAMsize= 8GB
storageCapacity= 128GB
CPU=i3_10110U
screenType=FHD_13inch
*****
Notebook
serialNumber= CZ2WTN40
releaseDate= 2019-06-30
RAMsize= 16GB
storageCapacity= 512GB
CPU=i7_1165G7
screenType=UHD_13inch
*****
Notebook
serialNumber= CZ3WTN40
releaseDate= 2020-07-20
RAMsize= 16GB
storageCapacity= 512GB
CPU=i9_10885H
screenType=UHD_15inch
touchScreen available
*****
Smartphone
serialNumber= DWW2HJ39
releaseDate= 2019-05-20
RAMsize= 4GB
storageCapacity= 64GB
CPU= kirin_9000
screenSizeInch= 6.7
*****
Smartphone
serialNumber= DWW2HJ40
releaseDate= 2018-06-15
RAMsize= 2GB
storageCapacity= 64GB
CPU= kirin_820
screenSizeInch= 6.7
```

TestFilters.java

```
****TESTING NotebookFilter with ReleaseDate ascendent sorting***
MyStore custom view contains 5 items.
Printing:
*****
Notebook
serialNumber= CY2WWN23
releaseDate= 2017-06-30
RAMsize= 8GB
storageCapacity= 128GB
CPU=i3_10110U
screenType=FHD_13inch
*****
Notebook
serialNumber= CY2WPN23
releaseDate= 2018-07-03
RAMsize= 8GB
storageCapacity= 256GB
CPU=i5_1135G7
screenType=FHD_15inch
touchScreen available
*****
Notebook
serialNumber= CZ2WTN40
releaseDate= 2019-06-30
RAMsize= 16GB
storageCapacity= 512GB
CPU=i7_1165G7
screenType=UHD_13inch
*****
Notebook
serialNumber= CZ3WTN40
releaseDate= 2020-07-20
RAMsize= 16GB
storageCapacity= 512GB
CPU=i9_10885H
screenType=UHD_15inch
touchScreen available
*****
Notebook
serialNumber= CZ3WTN42
releaseDate= 2020-07-20
RAMsize= 16GB
storageCapacity= 512GB
CPU=i9_10885H
screenType=UHD_15inch
touchScreen available

****TESTING HighStorage Filter***
MyStore custom view contains 5 items.
Printing:
*****
Notebook
serialNumber= CY2WPN23
releaseDate= 2018-07-03
RAMsize= 8GB
storageCapacity= 256GB
CPU=i5_1135G7
screenType=FHD_15inch
touchScreen available
*****
Notebook
serialNumber= CY2WWN23
releaseDate= 2017-06-30
RAMsize= 8GB
storageCapacity= 128GB
CPU=i3_10110U
screenType=FHD_13inch
*****
Notebook
serialNumber= CZ2WTN40
releaseDate= 2019-06-30
RAMsize= 16GB
storageCapacity= 512GB
CPU=i7_1165G7
screenType=UHD_13inch
*****
Notebook
serialNumber= CZ3WTN40
releaseDate= 2020-07-20
RAMsize= 16GB
storageCapacity= 512GB
CPU=i9_10885H
screenType=UHD_15inch
touchScreen available
*****
Notebook
serialNumber= CZ3WTN42
releaseDate= 2020-07-20
RAMsize= 16GB
storageCapacity= 512GB
CPU=i9_10885H
screenType=UHD_15inch
touchScreen available
```