Prova di Laboratorio di Programmazione — Edizione 2

20 febbraio 2014

ATTENZIONE: Non è possibile usare le classi del package `prog.io` del libro di testo

Lo scopo è modellare implementare un "time sheet" molto semplificato che elenchi le ore lavorative dedicate a diversi progetti di una o più persone.

Oltre ai metodi richiesti in ciascuna classe, è opportuno implementare altri metodi che si ritengono utili per svolgere l'esercizio nel modo più corretto ed elegante, per esempio il metodo `toString` e i metodi di accesso ai campi.

Il codice esterno ad una classe non deve mai accedere direttamente ai campi della classe.

Le operazioni di stampa vanno effettuate esclusivamente nelle classi che definiscono il metodo `main`.

Esercizio 1

Si implementino le seguenti classi.

Classe `Entry`.
Classe **astratta** che descrive un'attività della durata fissa di un'ora caratterizzata da: giorno del mese, ora di inizio, nome del progetto associato all'attività.

Classe `Work`.
Sottoclasse di `Entry` le cui istanze rappresentano un'attività consistente in un'ora di lavoro individuale.

Classe `Meeting`.
Sottoclasse di `Entry` le cui istanze rappresentano un'attività consistente in una riunione di un'ora caratterizzata da una lista di partecipanti.

La classe `Meeting` definisce il metodo

- `isInMeeting(String persona)`
  Restituisce `true` se la persona in argomento è nella lista dei partecipanti al meeting.

Scrivere un programma `Es1` che legge da **standard input** una sequenza di linee con i seguenti formati:

```
W,giorno,ora,progetto
M,giorno,ora,progetto,persona1,persona2,...
PW
PM
```

Le linee che iniziano con `W` richiedono la creazione e memorizzazione di un'attività di tipo `Work`. Le linee che iniziano con `M` richiedono la creazione e memorizzazione di un'attività di tipo `Meeting` (cioè una riunione) avente come partecipanti un numero variabile di persone. Le linee che iniziano con `PW` richiedono la stampa **in ordine cronologico** delle sole attività di tipo `Work` memorizzate fino a quel momento. Le linee che iniziano con `PM` richiedono la stampa **in ordine cronologico** delle sole attività di tipo `Meeting` memorizzate fino a quel momento.

Un'attività viene memorizzata se e solo se non inizia allo stesso giorno e ora di un'attività precedentemente memorizzata.

Non è necessario fare controlli sulla correttezza dell'input. Non vanno fatte assunzioni sul numero di linee in input. L'input va letto da **standard input**, possibilmente utilizzando la redirezione fornita da shell (si veda l'esempio sotto).

Esempio

Se il file `in1.txt` contiene le linee seguenti

```
M,10,3,Alfa,Giorgio,Anna
W,10,2,Beta
W,10,2,Alfa
M,9,12,Gamma,Luca,Giorgio
W,9,8,Gamma
M,1,5,Beta,Luca,Anna
M,10,3,Beta,Matteo,Luca
PM
PW
```

allora il comando `java Es1 < in1.txt` deve stampare

```
Giorno 1 ore 5: incontro per il progetto Beta
Giorno 9 ore 12: incontro per il progetto Gamma
Giorno 10 ore 3: incontro per il progetto Alfa
Giorno 9 ore 8: lavoro sul progetto Gamma
Giorno 10 ore 2: lavoro sul progetto Beta
```

Esercizio 2

Si implementi la classe seguente:

Classe `Timesheet`.
Un'istanza di questa classe rappresenta una sequenza di attività (di tipo `Entry`) create dal proprietario del timesheet.

La classe `Timesheet` definisce i metodi:

- `addEntry(Entry a)`
  Aggiunge l'attività fornita in argomento. Se l'attività inizia allo stesso giorno e ora di un'attività precedentemente memorizzata nel timesheet che esegue il metodo allora non viene svolta nessuna operazione.
- `lookup(String name)`
  Restituisce il numero di riunioni contenute nel timesheet in cui partecipa la persona in argomento.

Scrivere un programma `Es2` che legge da **standard input** una sequenza di linee con i seguenti formati:

```
C,proprietario
W,proprietario,giorno,ora,progetto
M,proprietario,giorno,ora,progetto,persona1,persona2,...
V,persona
```

Le linee che iniziano con `C` creano un timesheet il cui proprietario è quello specificato. Le linee che iniziano con `W` e `M` memorizzano attività di tipo `Work` e `Meeting` in modo simile all'Esercizio 1, con la differenza che l'attività viene memorizzata nel timesheet del proprietario specificato (si noti infatti che i comandi ora includono anche il nome del proprietario del timesheet). Per convenzione, assumiamo che nel comando `M` il proprietario del timesheet è colui che presiede la riunione e quindi non compaia mai fra i partecipanti. Le linee che iniziano con `V` richiedono di stampare: 1) il numero di progetti distinti per cui la persona specificata ha memorizzato almeno un'attività di tipo `Work` o ha presieduto almeno una riunione; 2) indicare il numero di riunioni (`Meeting`) in cui la persona specificata appare come partecipante (vedi esempio qui sotto).

Non è necessario fare controlli sulla correttezza dell'input. In particolare, si assuma che i comandi `W` e `M` si riferiscano sempre a timesheet precedentemente creati con un comando `C`. Non vanno fatte assunzioni sul numero di linee in input. L'input va letto da **standard input**, possibilmente utilizzando la redirezione fornita da shell (si veda l'esempio sotto).

Esempio

Se il file `in2.txt` contiene le linee seguenti

```
C,Luca
C,Giorgio
C,Anna
C,Matteo
M,Luca,10,3,Alfa,Giorgio,Anna
M,Anna,9,12,Gamma,Giorgio
W,Anna,10,2,Beta
W,Giorgio,10,2,Alfa
M,Anna,9,12,Gamma,Luca,Giorgio
W,Luca,9,8,Gamma
M,Giorgio,1,5,Beta,Luca,Anna
M,Matteo,10,3,Beta,Anna,Luca
W,Luca,10,10,Gamma
V,Luca
V,Anna
V,Matteo
```

allora il comando `java Es2 < in2.txt` deve stampare

```
Luca partecipa a 2 progetti diversi
Luca partecipa a 1 riunioni di Giorgio
Luca partecipa a 0 riunioni di Anna
Luca partecipa a 1 riunioni di Matteo
Anna partecipa a 2 progetti diversi
Anna partecipa a 1 riunioni di Luca
Anna partecipa a 1 riunioni di Giorgio
Anna partecipa a 1 riunioni di Matteo
Matteo partecipa a 1 progetti diversi
Matteo partecipa a 0 riunioni di Luca
Matteo partecipa a 0 riunioni di Giorgio
Matteo partecipa a 0 riunioni di Anna
```

Istruzioni per la consegna

Consegnare soltanto i file seguenti (**non** i file `.class`):

```
Entry.java                              Timesheet.java
Work.java                               Es1.java
Meeting.java                            Es2.java
```

come un unico file compresso creato col comando

```
> zip provalab Entry.java Work.java Meeting.java Timesheet.java Es1.java Es2.java
```

Eseguire l'upload del file all'indirizzo `upload.di.unimi.it`.

Non consegnare file che non compilano.