Tema d'esame - Edizione 1 - Turni A B C - 3 febbraio 2014

AVVERTENZA. *Non è ammesso l'uso delle classi del package* `prog.io` *allegato al libro di testo del corso.*

Agenda

Lo scopo è modellare e realizzare un'agenda di appuntamenti molto semplificata. Le classi da realizzare sono le seguenti:

- `Persona`, rappresenta una persona;
- `Appuntamento`, classe astratta che rappresenta un appuntamento generico;
- `AppuntamentoLavoro`, sottoclasse di `Appuntamento` che rappresenta un appuntamento di lavoro, in un luogo;
- `AppuntamentoPersonale`, sottoclasse di `Appuntamento` che rappresenta un appuntamento privato, con una persona;
- `Agenda`, rappresenta un'agenda in cui sono registrati gli appuntamenti in un determinato anno.

Tali classi dovranno esporre i metodi specificati nelle sezioni seguenti. Si noti che:

- Le classi, oltre ai metodi specificati, possono definire altri metodi che si ritengono utili (es., metodo `toString`, metodi set&get per accedere o modificare le informazioni, ecc.).
- Le classi e i metodi definiti nella specifica devono essere **pubblici**, i campi delle classi devono essere **privati**.
- L'intestazione di alcuni metodi è incompleta (vanno aggiunte le definizioni dei tipi).

Persona

Classe che descrive una persona di cui si specificano *nome* e *cognome* di tipo String, creare metodi get&set opportuni.

(abstract) Appuntamento

Classe astratta che descrive un appuntamento di cui si specifica la descrizione (una stringa), il giorno, l'ora di inizio. Per semplicità si assume che:

- La data è una stringa nel formato mm-gg, ad esempio `"02-03"` per "3 febbraio", `"10-15"` per "15 ottobre"; l'anno dell'appuntamento è sottointeso.
- L'ora è un intero compreso fra 0 e 23.
- Un appuntamento dura 2 ore.

Devono essere sollevate eccezioni se la data non è nel formato mm-gg, se mm non è un mese valido e se gg è maggiore di 31 o minore di 1 (usare una delle eccezioni già definite, esempio `IllegalArgumentException`).

La classe deve esporre il metodo

- `inConflitto(Appuntamento altro)`

  Restituisce true se l'appuntamento passato come parametro si sovrappone cronologicamente a quello che invoca il metodo, false altrimenti.

AppuntamentoLavoro

Sottoclasse concreta di `Appuntamento` che descrive un appuntamento di lavoro. La classe deve specificare il luogo dove avviene l'appuntamento (una stringa).

AppuntamentoPersonale

Sottoclasse concreta di `Appuntamento` che descrive un appuntamento con una persona. La classe deve specificare la persona con cui avviene l'appuntamento.

Agenda

La classe è caratterizzata dalla persona titolare dell'agenda e deve contenere l'elenco degli appuntamenti fissati. L'agenda non può contenere appuntamenti che si sovrappongono. Deve avere un costruttore che riceve come parametro la persona titolare dell'agenda e deve esporre i seguenti metodi pubblici:

- `aggiungiAppuntamento(Appuntamento a)`

  Se l'appuntamento `a` non si sovrappone con gli appuntamenti già nell'agenda, `a` viene aggiungo all'agenda.

- `statistiche(String tipo)`

  Restituisce la percentuale di appuntamenti di lavoro o quella di appuntamenti personali, a seconda della stringa passata come parametro ("`lavoro`" o "`personale`").

- `ordina()`

  Ordina gli appuntamenti dell'agenda in ordine cronologico (ordine crescente per data e, a parità di data, ordine crescente per ora).

- (FACOLTATIVO) `cancellaAppuntamento(Appuntamento a)`

  Se l'appuntamento `a` è in agenda, viene cancellato. (Suggerimento: utilizzare *opportunamnente* il metodo `contains` per verificare se l'appuntamento è in agenda).

Esercizio 1

Scrivere una classe tester `AgendaTester` che crea una agenda per Mario Rossi e inserisce uno di seguito all'altro i seguenti appuntamenti (gli appuntamenti vanno scritti nel codice, la classe non deve effettuare operazioni di input):

```
appuntamento di lavoro "riunione",  il 12 marzo alle 13, in sala 1
appuntamento di lavoro "seminario", il 12 marzo alle 10, in aula magna
appuntamento di lavoro "riunione",  il 12 marzo alle 11, in sala 2
appuntamento personale "cena",      il  5 febbraio alle 20, con Paolo Bianchi
appuntamento personale "visita",    il  4 febbraio alle 9,  con Laura Ferrari
appuntamento di lavoro "colloquio", il  4 marzo alle 10, in ufficio 10
```

Successivamente l'agenda viene ordinata (metodo `ordina`) e vengono stampati gli appuntamenti. Controllare che gli appuntamenti siano stampati in ordine cronologico e che gli appuntamenti che si sovrappongono ad altri (es., la seconda riunione) non siano stati inseriti nell'agenda.

Esercizio 2

Scrivete una classe `Main`, contenente il metodo `main`, che permetta all'utente di creare e gestire un'agenda di appuntamenti, come segue.

Il programma accetta due parametri in ingresso dalla riga di comando, uno per il nome e uno per il cognome del titolare. Per esempio,

```
Luigi Verdi
```

Il programma deve gestire opportunamente il caso in cui il nome del titolare non venga specificato. (Facolativo: gestire il caso in cui nome e/o cognome siano composti da piú parole)

Il programma presenta poi all'utente un menu con le voci:

1. `Aggiungi appuntamento.`
2. `Cancella appuntamento.` (se non implementato, lasciare la voce nel menu con un commento)
3. `Ordina gli appuntamenti.`
4. `Stampa l'agenda.`
5. `Esci.`

Per ciascun caso il programma deve chiedere all'utente le informazioni per portare a termine l'operazione (es. descrizione, ecc. dell'appuntamento). Il programma esegue l'operazione richiesta, poi torna al menu, tranne per l'ultima opzione, che permette di uscire dal programma. Si veda la Figura 1 per un esempio di esecuzione.

```
3
1. Aggiungi appuntamento
2. Cancella appuntamento
3. Ordina gli appuntamenti
4. Stampa l'agenda
5. Esci
4
Agenda di Luigi Verdi
01-30 h: 9:00 riunione  - sala 1
01-30 h: 11:00 seminario  - aula magna
02-01 h: 20:00 cena  con Laura Del Re
02-01 h: 22:00 cinema  con Paolo Neri
Lavoro: 50.0%
Personali: 50.0%
```

Figura 1: Esempio di esecuzione per l'Esercizio.

Consegna

Ricordo che le classi devono essere tutte *public* e che vanno consegnati tutti i file *.java* prodotti. NON vanno consegnati i *.class*. Non vanno inoltre consegnati file *.java* che non compilano (un elaborato che non compila correttamente non verrà esaminato).

Per la consegna, eseguite l'upload dei singoli file sorgente dalla pagina web: http://upload.di.unimi.it.