Lab 14 – Oroscopi
Tempo a disposizione: 3,5 ore MAX

La rivista di astrologia "Vedo, prevedo, stravedo" ha richiesto lo sviluppo di un'applicazione per la generazione automatica di oroscopi, in modo da sollevare gli astrologi dall'inutile fatica di doverli comporre personalmente.

DESCRIZIONE DEL DOMINIO DEL PROBLEMA

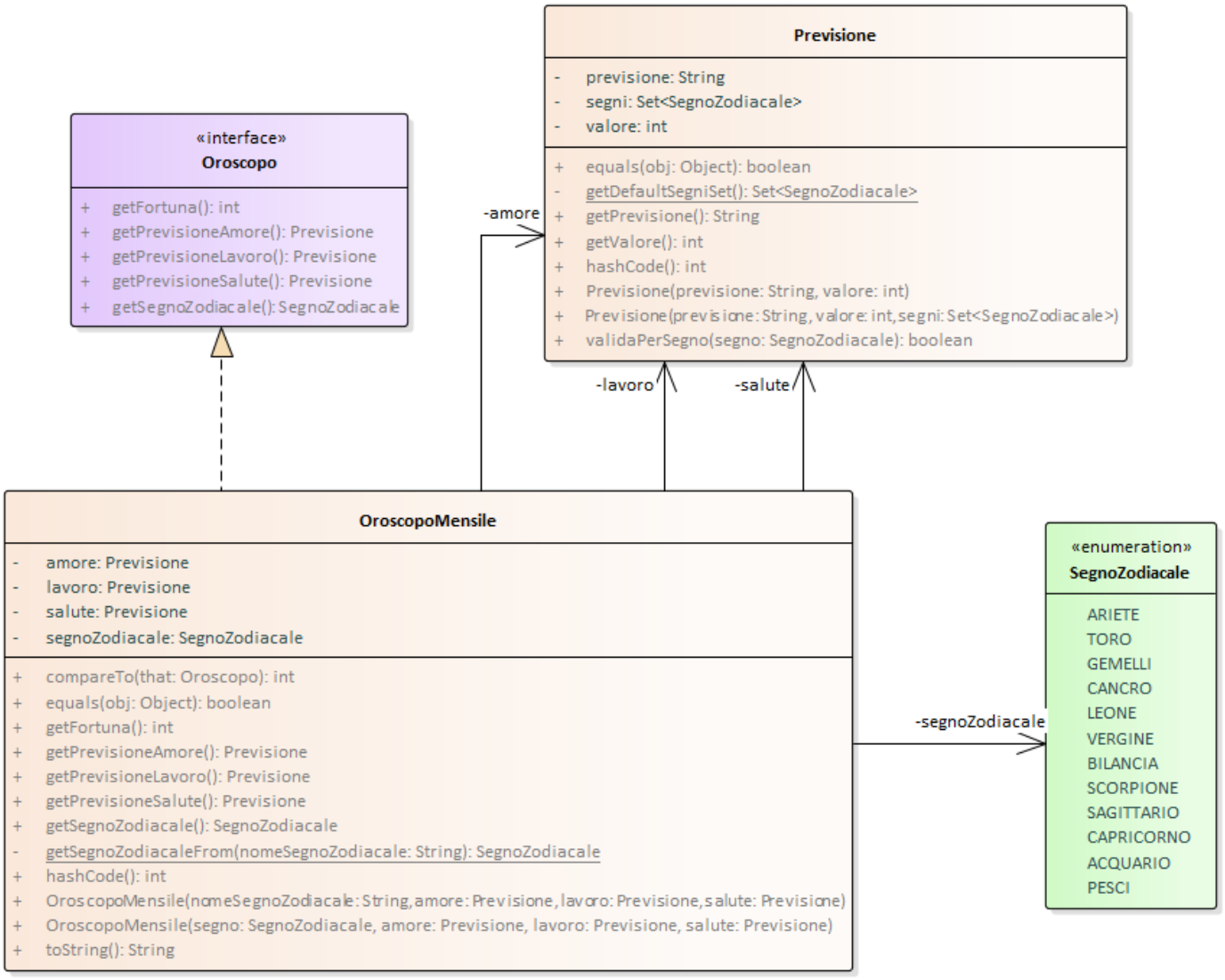
Ogni *oroscopo mensile* si riferisce ad un *Segno Zodiacale* ed è composto da tre *previsioni* riferite a tre settori (amore, lavoro, salute) nonché da una indicazione riassuntiva di fortuna, espressa in una scala da 1 (minimo) a 10 (massimo).

L' *oroscopo annuale* di un dato segno è costituito da 12 oroscopi mensili, ognuno della forma sopra specificata.

Parte 1 (punti: 15)

Dati (package oroscopo.model) *(punti: 6)*

Il modello dei dati deve essere organizzato secondo il diagramma UML più sotto riportato.



SEMANTICA:

a) L'enumerativo ***SegnoZodiacale*** (fornito nello start kit) rappresenta i dodici segni dello Zodiaco

b) l'interfaccia ***Oroscopo*** (fornita nello start kit), rappresenta un oroscopo con le proprietà descritte sopra;

c) La classe ***Previsione*** (fornita nello start kit) modella una singola previsione ed è caratterizzata dal testo della previsione e da un valore numerico che rappresenta il "grado di fortuna" associato alla previsione stessa. Il costruttore principale accetta un `Set<SegnoZodiacale>`, che rappresenta l'insieme dei segni per i quali la ***Previsione*** è valida; è disponibile anche un costruttore senza la specifica dei segni, che costruisce una ***Previsione*** valida per tutti i segni. La classe espone il metodo ***validaPerSegno*** che consente di verificare se una ***Previsione*** è valida per il segno passato come argomento.

d) La classe ***OroscopoMensile*** **(da realizzare)** rappresenta un oroscopo mensile, implementa le interfacce ***Oroscopo*** e ***Comparable<Oroscopo>*** (confronto di un ***OroscopoMensile*** con un ***Oroscopo***!) e prevede due costruttori:

- ***OroscopoMensile(SegnoZodiacale, Previsione, Previsione, Previsione)*** che costruisce un'istanza valida per il segno passato come primo argomento, usando le tre previsioni ricevute come secondo, terzo e quarto argomento rispettivamente per le sezioni amore, lavoro e salute dell'oroscopo;
- ***OroscopoMensile(String, Previsione, Previsione, Previsione)*** che opera come sopra ma ricevendo il segno zodiacale come stringa anziché come istanza di ***SegnoZodiacale***.

È compito dei costruttori verificare:

a) che i parametri passati non siano **null**;

b) che le previsioni con cui viene costruito l'***OroscopoMensile*** siano valide per il ***SegnoZodiacale*** associato;

c) nel caso del solo costruttore con parametro ***String***, che la stringa passata rappresenti effettivamente un valore valido dell'enumerativo ***SegnoZodiacale***.

In tutti i casi in cui un parametro non sia valido occorre lanciare una ***IllegalArgumentException.***

La classe espone inoltre i seguenti metodi:

- ***getFortuna*** che restituisce *il grado di fortuna medio,* definito come media del grado di fortuna delle tre previsioni relative alle sezioni (amore, lavoro, salute) arrotondata all'intero più vicino;
- ***compareTo*** che ordina gli oroscopi seguendo l'ordine dell'enumerativo ***SegnoZodiacale*** (per questo è sufficiente un ***Oroscopo*** come argomento);
- ***toString*** che restituisce il risultato della concatenazione del valore restituito dal metodo ***getPrevisione*** delle tre previsioni.

Persistenza (package oroscopo.persistence) *(punti 9)*

Il file di testo FrasiOroscopo.txt contiene le frasi da usare per comporre le previsioni. Tali frasi sono raggruppate per settore (amore, salute, lavoro): più precisamente, ogni blocco ha come prima riga il nome del settore (ad esempio, "SALUTE") e termina con la riga "FINE". Il file può contenere righe vuote che vanno saltate.

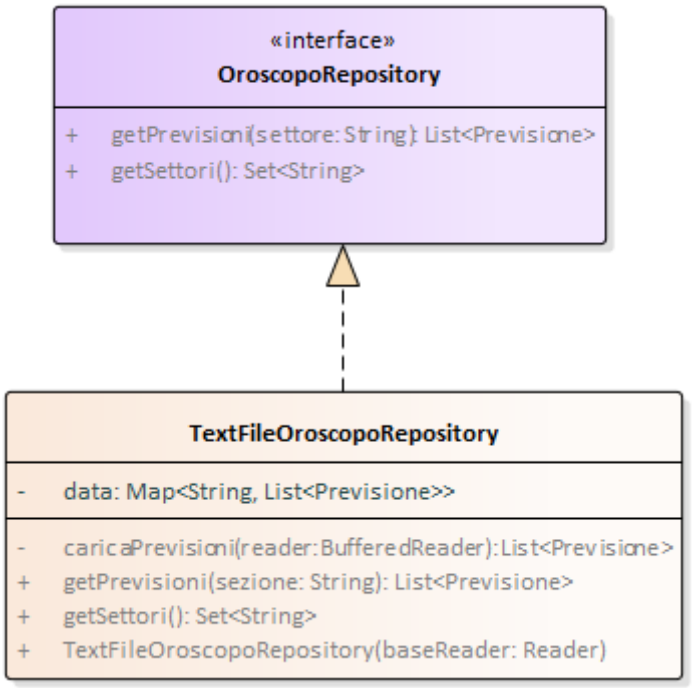
Ogni frase rappresenta una singola previsione ed è strutturata con i seguenti campi, separati da tabulazioni (carattere '\t'):

- testo della frase
- gradi di fortuna (intero maggiore o uguale a zero)
- **opzionalmente,** i segni per i quali la previsione è valida (elenco separato da virgole, senza spazi intermedi).

Esempio:

```
AMORE
avrai la testa un po' altrove          4      ARIETE,TORO,GEMELLI
divertimento nello stare insieme       8
FINE
LAVORO
sopravvalutate chi vi è intorno        4
FINE
SALUTE
meglio mangiare    modo salutare       6
FINE
```

È opportuno sottolineare che i settori sono presenti sequenzialmente nel file **senza un ordine prestabilito**: la lettura dovrà quindi evitare ipotesi sulla sequenza in cui le sezioni compaiono. (Come conseguenza, è potenzialmente possibile che non siano fornite frasi per un dato settore.)



a) L'interfaccia ***OroscopoRepository*** (fornita nello start kit) dichiara due metodi:
   - ***getPrevisioni(String)*** che recupera le previsioni del settore ricevuto come argomento;
   - ***getSettori*** che restituisce l'insieme dei settori (tipicamente tutti, ma potenzialmente anche non tutti) per i quali sono presenti frasi nel repository.

b) La classe ***TextFileOroscopoRepository*** **(da realizzare)** implementa tale interfaccia lanciando ***BadFileFormatException*** (fornita) in caso di errore di lettura nel formato del file.
   - Il **costruttore** della classe riceve un ***Reader*** da utilizzare per la lettura ed effettua la lettura lanciando ***IllegalArgumentException*** nel caso in cui tale il reader ricevuto sia ***null*** o ***BadFileFormatException*** in caso di formato del file errato (segnalando opportunamente l'errore). I dati letti da file sono memorizzati in una opportuna struttura dati (ad esempio una mappa contente per ogni Settore la lista di Previsioni associata)
   - Il metodo ***getPrevisioni*** deve funzionare in modo case insensitive, in altre parole, il parametro stringa che rappresenta il nome della sezione può contenere indifferentemente lettere minuscole e/o maiuscole, quindi, ad esempio, usare tale metodo con in input "Lavoro" è equivalente ad usarlo con in input "LAVORO".

Lo Start Kit contiene anche i test (da includere nel progetto) per verificare il funzionamento di questa classe.

Parte 2 (punti: 15)

IMPORTANTE: per consentire il test della GUI anche nel caso di *reader* non funzionanti, è fornita la classe **OroscopoApplicationMock** che replica **OroscopoApplication** utilizzando però dati fissi pre-cablati al posto di quelli letti da file.

L'applicazione deve permettere all'utente-astrologo di scegliere il segno zodiacale e il mese di interesse e generare il corrispondente oroscopo mensile, mostrandolo in un'area di testo. Inoltre, premendo l'apposito pulsante, l'applicazione deve generare sul file di testo OroscopoAnnuale.txt l'oroscopo annuale per il segno selezionato: in tal caso, vi è l'ulteriore vincolo che il grado di fortuna medio *sull'intero anno* sia superiore ad una soglia prestabilita (non impostabile dall'utente).

Controller (package oroscopo.controller) *(punti 9)*

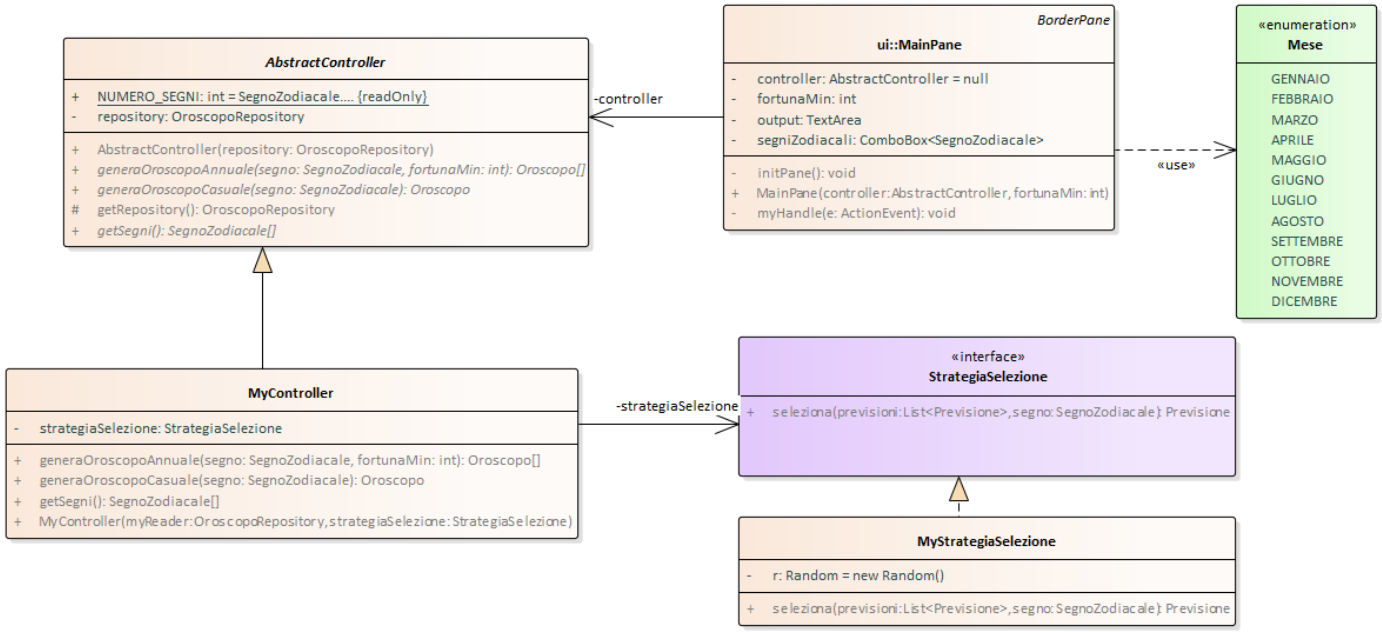
La classe ***AbstractController*** (fornita) implementa parzialmente il controller dell'applicazione

- il costruttore riceve come parametro un ***OroscopoRepository*** che viene memorizzato internamente in un campo privato il cui valore è recuperabile dalle sottoclassi mediante il metodo protetto ***getRepository***;
- il metodo ***getSegni*** restituisce l'elenco dei segni zodiacali;
- il metodo ***generaOroscopoCasuale*** produce un oroscopo casuale mensile per il segno specificato;
- il metodo ***generaOroscopoAnnuale*** produce e restituisce un array di 12 istanze di ***Oroscopo*** valide per il segno zodiacale indicato, con grado di fortuna medio superiore al parametro **fortunaMin**
  [SUGGERIMENTO: generare **iterativamente** un oroscopo annuale fino a che se ne trovi uno sopra soglia]

La classe ***MyController (da realizzare)*** concretizza ***AbstractController*** avvalendosi opportunamente del metodo protetto ereditato ***getRepository***. Essa prevede un unico costruttore a due argomenti – rispettivamente, un ***OroscopoRepository*** che fornisce l'accesso ai dati e una ***StrategiaSelezione*** che incapsula la scelta delle previsioni.

L'interfaccia ***StrategiaSelezione*** (fornita) dichiara un singolo metodo di selezione di una ***Previsione*** all'interno di una lista: per contratto, l'istanza restituita deve essere valida per il segno zodiacale ricevuto come argomento.

La classe ***MyStrategiaSelezione (da realizzare)*** implementa ***StrategiaSelezione*** adottando come strategia di scelta della ***Previsione*** da restituire l'estrazione a sorte fra le previsioni disponibili per il segno in questione. A tale scopo si faccia opportuno uso della classe ***Random*** e si iteri finché l'estrazione restituisce una previsione valida per il segno ricevuto come parametro.



Interfaccia utente (package oroscopo.ui) *(punti 6)*

L'interfaccia utente deve essere simile (non necessariamente identica) all'esempio mostrato di seguito.

Se il caricamento preliminare (realizzato nell'***OroscopoApplication*** fornito nello start kit, non mostrato nel diagramma UML) ha esito positivo, compare la finestra principale dell'applicazione (Fig. 1), costituita da un'istanza di ***MainPane***

(***da realizzare***): essa prevede un unico costruttore a due argomenti – un ***AbstractController*** ed un intero che rappresenta il grado di fortuna minimo degli oroscopi annuali da generare.

Funzionamento:

- in primo luogo, si deve scegliere il segno zodiacale da una Com*boBox<SegnoZodiacale>* (Fig. 1) (il mese è completamente irrilevante), ottenendo in risposta l'oroscopo mensile (Fig. 2) nell'area di testo;
- premendo invece il pulsante STAMPA ANNUALE, l'applicazione genera sul file di testo OroscopoAnnuale.txt (un esempio di tale file è riportato nello start kit) l'oroscopo annuale per il segno selezionato nella combo, garantendo un grado di fortuna medio *sull'intero anno* superiore alla soglia prestabilita (non impostabile dall'utente).

  Nota: a differenza del solito, al **solo scopo** di semplificare la struttura del sistema, la scrittura su file non è gestita da un opportuno writer tramite il controller ma è gestita direttamente nella UI.

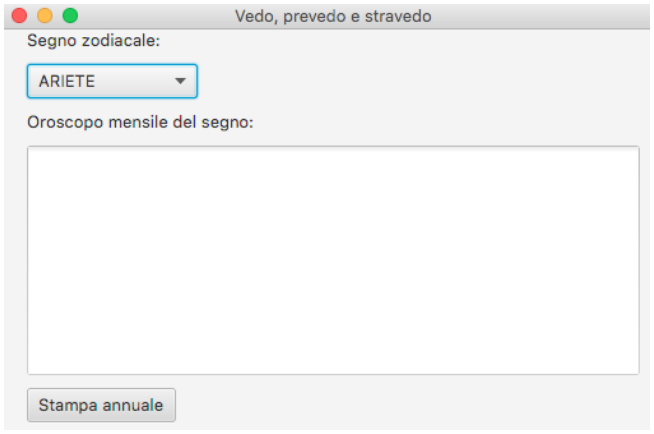


Fig. 1

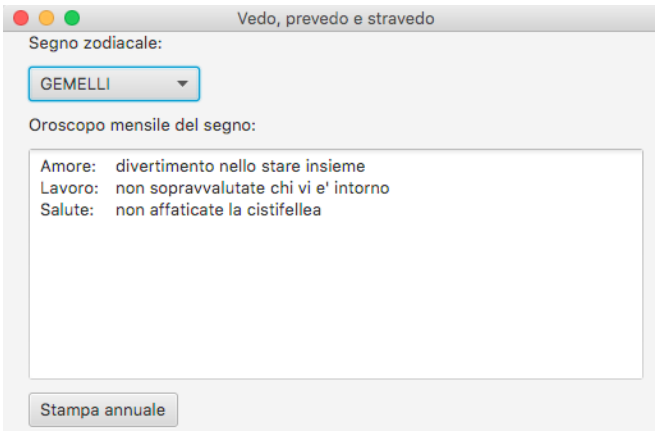


Fig. 2