ESAME DI FONDAMENTI DI INFORMATICA T-2 del 04/07/2018

Proff. E. Denti – R. Calegari – G. Zannoni Tempo: 4 ore

NOME PROGETTO ECLIPSE: CognomeNome-matricola (es. RossiMario-0000123456)
NOME CARTELLA PROGETTO: CognomeNome-matricola (es. RossiMario-0000123456)
NOME ZIP DA CONSEGNARE: CognomeNome-matricola.zip (es. RossiMario-0000123456.zip)

NB: l'archivio ZIP da consegnare deve contenere l'intero progetto Eclipse

La società *CupidOnLine* vuole offrire un servizio di ricerca intelligente dell'anima gemella: sulla base delle caratteristiche proprie e del partner desiderato, l'applicazione deve proporre una serie di candidati/e.

DESCRIZIONE DEL DOMINIO DEL PROBLEMA.

Ogni **persona** in archivio è caratterizzata da una serie di elementi:

- nickname (identificativo univoco)
- sesso
- data di nascita (utile per il calcolo dell'età e del segno zodiacale)
- caratteristiche fisiche (colore occhi, colore capelli, altezza, peso)
- luogo di residenza (città, provincia e regione)

Una **preferenza** è un insieme di caratteristiche che il partner dovrebbe soddisfare: alcune devono essere specificate per forza, altre invece sono opzionali. Più precisamente:

- sesso (obbligatorio)
- range di età (obbligatorio)
- segno zodiacale (opzionale)
- caratteristiche fisiche (colore occhi, colore capelli, altezza, peso) (opzionali)
- luogo di residenza (città, provincia, regione) (obbligatorio)

Una **corrispondenza** è una possibile coppia di persone sicuramente compatibili per sesso e caratterizzata da un ***indice di compatibilità*** **≤ 100** calcolato sulla base della vicinanza fra la propria preferenza e le caratteristiche del potenziale partner, prendendo il *valore minimo* fra i seguenti sotto-indici:

- età: 100 se nel range, -5 per ogni anno fuori range (dal lato più vicino);
- luogo di residenza: 100 nella città, 90 nella provincia, 60 nella regione, 40 fuori regione;
- segno zodiacale: 100 se identico a quello richiesto o non specificato, 90 altrimenti;
- caratteristiche fisiche dimensionali: 100 se identico a quello richiesto o non specificato, altrimenti -1 per ogni cm o kg di differenza;
- caratteristiche fisiche adimensionali: 100 se identico a quello richiesto o non specificato, 95 altrimenti.

L'insieme dei possibili partner dev'essere ***ordinato in senso discendente per indice di compatibilità***, in modo da proporre per primi i partner più interessanti.

ESEMPIO

Persona : Roberto, 24 anni, Toro, capelli neri, occhi castani, 1.78, 61 kg, Bologna (BO, ER);
richiede partner donna , 20-25, Bilancia, capelli biondi, occhi -, 1.70, 58 kg, Bologna (BO, ER)
Candidate proposte:
- Anna, 22, Gemelli, cap. biondi, occhi azzurri, 1.70, 60, Imola (BO,ER)
- Ludovica, 26, Bilancia, cap. neri, occhi castani, 1.75, 51, Bologna (BO,ER)
- Elena, 19, Ariete, cap. neri, occhi neri, 1.65, 57, Modena (MO,ER)

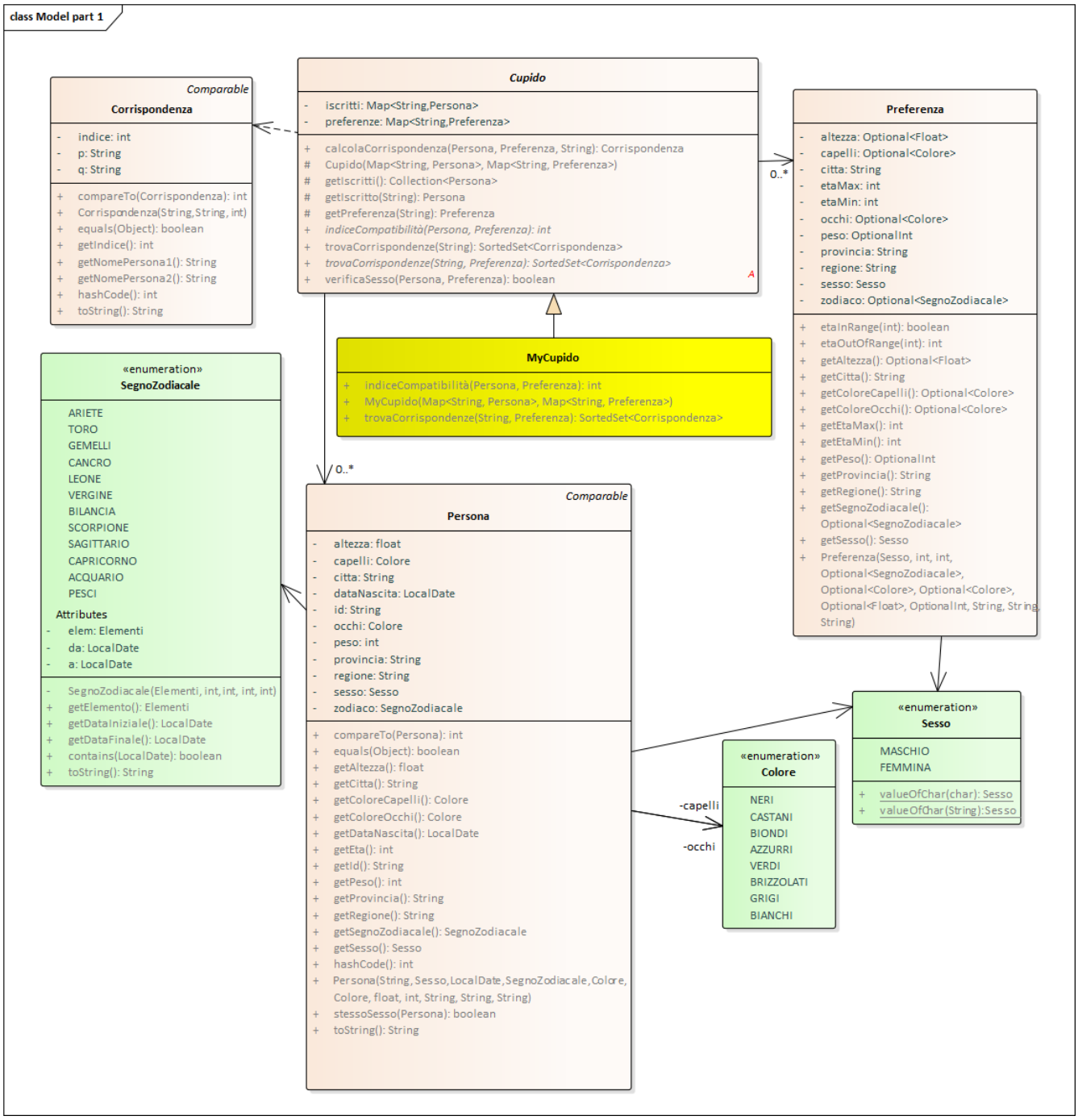
Calcolo:	*nickname*	*età*	*luogo*	*zod.*	*alt.*	*peso*	*capelli*	*MIN*	*classifica*
	Anna	100	90	**90**	100	98	100	**90**	**2°**
	Ludovica	95	100	100	95	**93**	95	**93**	**1°**
	Elena	95	**60**	90	95	99	95	**60**	**3°**

I due file di testo Iscritti.txt e Preferenze.txt contengono rispettivamente l'elenco degli iscritti e delle preferenze, nel formato più oltre specificato.

Parte 1 (punti: 21)

Dati (namespace cupidonline.model) *(punti: 12)*

Il modello dei dati deve essere organizzato secondo il diagramma UML di seguito riportato:



SEMANTICA:

a) l'enumerativo ***Colore*** (fornito) definisce i colori leciti per capelli e occhi;

b) l'enumerativo ***Sesso*** (fornito) definisce i due sessi: i due metodi factory ausiliari ***valueOfChar*** restituiscono il valore dell'enumerativo corrispondente al carattere 'M' o 'F' (o stringa unitaria) passato come argomento;

c) l'enumerativo ***SegnoZodiacale*** (fornito) definisce i dodici segni con le rispettive date di inizio e fine: il metodo ausiliario ***contains*** verifica se una certa data è compresa nel range di un dato segno zodiacale (NB: internamente viene definito e usato un ulteriore tipo enumerativo ***Elemento*** per esprimere l'elemento *terra,acqua,aria,fuoco* di ogni segno: ciò non traspare però all'esterno, quindi è irrilevante ai fini di questo compito);

d) la classe ***Persona*** (fornita) rappresenta un iscritto dell'agenzia, con tutte le sue proprietà; in particolare il nickname (stringa) è garantito essere un identificativo univoco [quindi può essere usato come chiave nelle mappe..]; ***Persona*** è anche ***Comparable*** in senso alfabetico crescente per nickname;

e) la classe ***Preferenza*** (fornita) rappresenta una preferenza, con tutte le sue proprietà, *molte delle quali opzionali*; oltre agli accessor, i metodi di utilità ***etaInRange*** e ***etaOutOfRange*** rispettivamente verificano se l'età data è nel range ammesso da quella preferenza, o di quanti anni sia lontana dal range ammesso dalla preferenza;

f) la classe ***Corrispondenza*** (fornita) esprime la corrispondenza fra due persone, identificate dal loro nickname, con relativo indice di compatibilità; è ***Comparable*** in senso decrescente per indice di compatibilità;

g) la classe astratta ***Cupido*** (fornita) incapsula i dati (iscritti e preferenze) e definisce la maggior parte dei metodi, lasciando astratti solo i due che dipendono da specifici algoritmi. Più esattamente:

- il costruttore riceve gli iscritti e le preferenze, sotto forma di mappe (prodotte dai reader) rispettivamente di tipo ***Map<String, Persona>*** per gli iscritti, e ***Map<String, Preferenza>*** per le preferenze; in entrambi i casi, la chiave è il nickname della persona;
- il metodo ***trovaCorrispondenze(String nickname)*** trova le corrispondenze per l'iscritto identificato dal ***nickname***, utilizzando la sua stessa preferenza: costituisce un caso particolare del metodo successivo (più generale): restituisce un insieme *ordinato in senso decrescente per indice di compatibilità;*
- il metodo ***trovaCorrispondenze(String nickname, Preferenza pref)*** è astratto e descritto sotto al punto h); il suo obiettivo è cercare in generale corrispondenze fra un iscritto e una preferenza, in modo da gestire sia il caso in cui un iscritto cerchi un partner fra gli altri iscritti (e allora la preferenza sarà la sua), sia il caso in cui un *non iscritto* voglia cercare fra i già iscritti dell'agenzia (e allora la preferenza del non iscritto sarà stata ottenuta separamente, magari compilando un questionario sull'interfaccia grafica)
- il metodo ***verificaSesso(Persona q, Preferenza pref)*** verifica che la persona sia di sesso uguale a quello richiesto dalla preferenza, se quest'ultima non è nulla (altrimenti è comunque falso: non lancia eccezioni);
- il metodo ***calcolaCorrispondenza(Persona q, Preferenza pref, String nickname)*** calcola la ***Corrispondenza*** fra la persona ***q*** e una preferenza ***pref***, associando quest'ultima al ***nickname*** fornito: in questo modo è possibile calcolare corrispondenze anche fra un iscritto ***q*** e un non iscritto di cui sia nota la preferenza;
- il metodo ***indiceCompatibilità(Persona q, Preferenza pref)*** è astratto e descritto sotto al punto h); il suo scopo è calcolare l'indice di compatibilità fra una persona e una preferenza, secondo un certo criterio.

h) la classe **My*Cupido* (da realizzare)** estende ***Cupido*** implementando per la ricerca corrispondenze e il calcolo dell'indice di compatibilità gli specifici algoritmi descritti nel *Dominio del Problema*. Più precisamente:

- il costruttore, che riceve gli iscritti e le preferenze, delega interamente la costruzione alla classe base.
- il metodo ***trovaCorrispondenze(String nickname, Preferenza pref)*** trova tutte le corrispondenze di una data preferenza ***pref***, associandole poi nella ***Corrispondenza*** all'identificativo ***nickname***: ciò rende possibile cercare corrispondenze anche per persone non iscritte, di cui sia nota la preferenza. Se la preferenza è nulla si intende che non esistono partner corrispondenti (tale aspetto è già verificato nel metodo ***verificaSesso***). Il metodo restituisce un insieme *ordinato in senso decrescente per indice di compatibilità.*
- il metodo ***indiceCompatibilità(Persona q, Preferenza pref)*** calcola l'indice di compatibilità, compreso fra 0 e 100, fra la persona ***q*** e la preferenza ***pref***, secondo l'algoritmo descritto nel *Dominio del Problema*.

Prendere il *valore minimo* fra i seguenti sotto-indici:

- età: 100 se nel range, -5 per ogni anno fuori range (dal lato più vicino);
- luogo di residenza: 100 nella città, 90 nella provincia, 60 nella regione, 40 fuori regione;
- segno zodiacale: 100 se identico a quello richiesto o non specificato, 90 altrimenti;
- caratteristiche fisiche dimensionali: 100 se identico a quello richiesto o non specificato, altrimenti -1 per ogni cm o kg di differenza;
- caratteristiche fisiche adimensionali: 100 se identico a quello richiesto o non specificato, 95 altrimenti.

Persistenza (namespace cupidonline.persistence) *(punti 9)*

Come già anticipato, i due file Iscritti.txt e Preferenze.txt contengono rispettivamente l'elenco degli iscritti e delle preferenze, uno/a per riga: ogni riga contiene una serie di dati separati fra loro da virgole.

Sebbene il formato dei due file sia simile, i dati degli iscritti sono sempre completi, mentre **le preferenze possono contenere una lineetta ("-") al posto delle caratteristiche opzionali non specificate.** In particolare:

- per gli iscritti si specificano identificativo univoco, sesso, data di nascita (formato ISO), colore capelli, colore occhi (preceduti da apposita parola chiave "capelli" o "occhi", rispettivamente), altezza in metri (numero reale), peso in kg (numero intero), e infine città, provincia e regione di residenza;
- per le preferenze si specificano invece l'identificativo univoco della persona a cui la preferenza è associata e l'insieme delle caratteristiche richieste al potenziale partner, alcune delle quali però sono opzionali e quindi possono essere sostituite da una lineetta (vedere esempio sotto). Più precisamente:
  - **SONO SEMPRE PRESENTI** sesso, *range di età* (nel formato "*nn-mm*", dove *nn* e *mm* rappresentano l'età minima e massima), città, provincia e regione di residenza;
  - **SONO INVECE OPZIONALI** segno zodiacale (non necessariamente tutto maiuscolo), colore capelli e occhi (la parola chiave "capelli" o "occhi" è però comunque presente), altezza, peso.

  Da notare che sesso e range di età sono a inizio riga, mentre le informazioni sulla residenza sono in fondo alla riga: i dati opzionali sono quindi al centro fra questi.

ESEMPIO DEL FILE Iscritti.txt

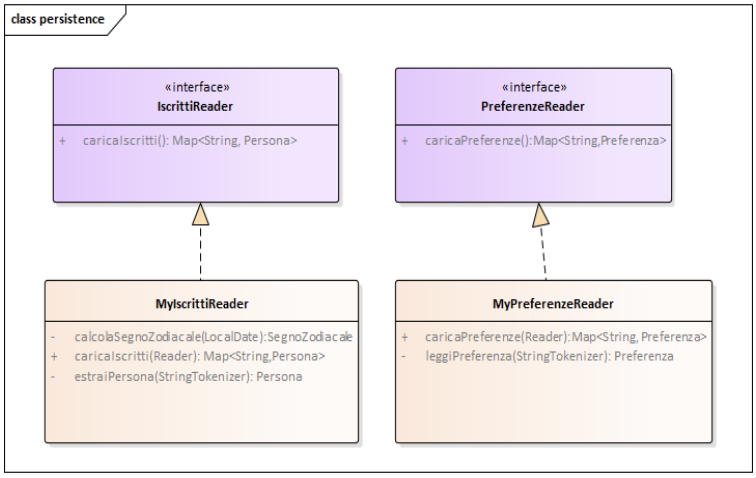
```
Roberto, M, 1994-04-24, capelli neri, occhi castani, 1.78, 61, Bologna, BO, Emilia-Romagna
Armando, M, 1997-08-01, capelli castani, occhi castani, 1.71, 65, Parma, PR, Emilia-Romagna
Eufrasio, M, 1993-10-30, capelli biondi, occhi azzurri, 1.82, 66, Firenze, FI, Toscana
Anna, F, 1996-06-18, capelli biondi, occhi azzurri, 1.70, 60, Imola, BO, Emilia-Romagna
Ludovica, F, 1992-10-14, capelli neri, occhi castani,1.75, 51, Bologna, BO, Emilia-Romagna
Elena, F, 1999-04-10, capelli neri, occhi neri,      1.65, 57, Modena, MO, Emilia-Romagna
```

ESEMPIO DEL FILE Preferenze.txt

```
Roberto, F, 20-25, Bilancia, capelli biondi, occhi -, 1.70, 58, Bologna, BO, Emilia-Romagna
Armando, F, 17-25, -, capelli -, occhi -, -, -, Parma, PR, Emilia-Romagna
Anna,      M, 20-29, Gemelli, capelli biondi, occhi azzurri, 1.70, 58, Imola, BO, Emilia-Romagna
Ludovica, M, 24-35, -, capelli -, occhi -, 1.80, -, Bologna, BO, Emilia-Romagna
Elena,     M, 18-23, -, capelli neri, occhi azzurri, 1.75, 58, Modena, MO, Emilia-Romagna
```

Le due interfacce ***IscrittiReader*** e ***PreferenzeReader*** (fornite) dichiarano i metodi ***caricaIscritti*** e ***caricaPreferenze*** che, dato un ***Reader***, restituiscono rispettivamente la mappa ***Map<String, Persona>*** degli iscritti e la mappa ***Map<String, Preferenza>*** delle preferenze, richieste dal costruttore di ***Cupido***.

Le due classi *MyIscrittiReader* (da realizzare) e ***MyPreferenzeReader (fornita)*** implementano tali interfacce effettuando i necessari controlli sul formato del file, lanciando ***BadFileFormatException*** (fornita) in caso di errori di formato, o propagando ***IOException*** in caso di errori di lettura con specifico messaggio d'errore.

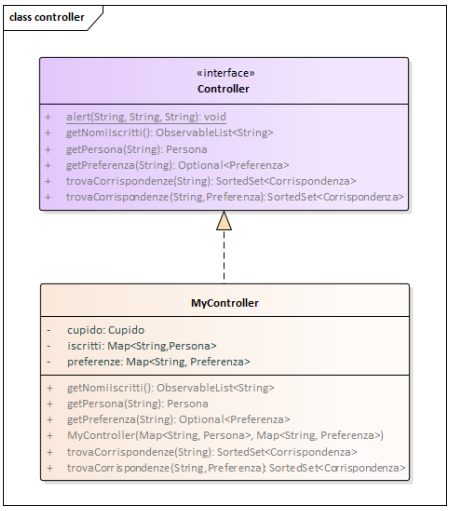


Parte 2 (punti: 9)

Controller (namespace cupido.ui.controller)

L'interfaccia ***Controller*** (fornita) dichiara i metodi ***getNomiIscritti***, ***getPersona***, ***getPreferenza*** e due versioni di ***trovaCorrispondenze***:

- ***getNomiIscritti***: restituisce la lista osservabile dei nomi degli iscritti;
- ***getPersona***: restituisce la ***Persona*** corrispondente al nickname fornito;
- ***getPreferenza***: restituisce, se esiste, la ***Preferenza*** della persona il cui nickname è specificato come argomento;
- ***trovaCorrispondenze*** (due versioni overloaded): richiama gli analoghi metodi di ***Cupido***.



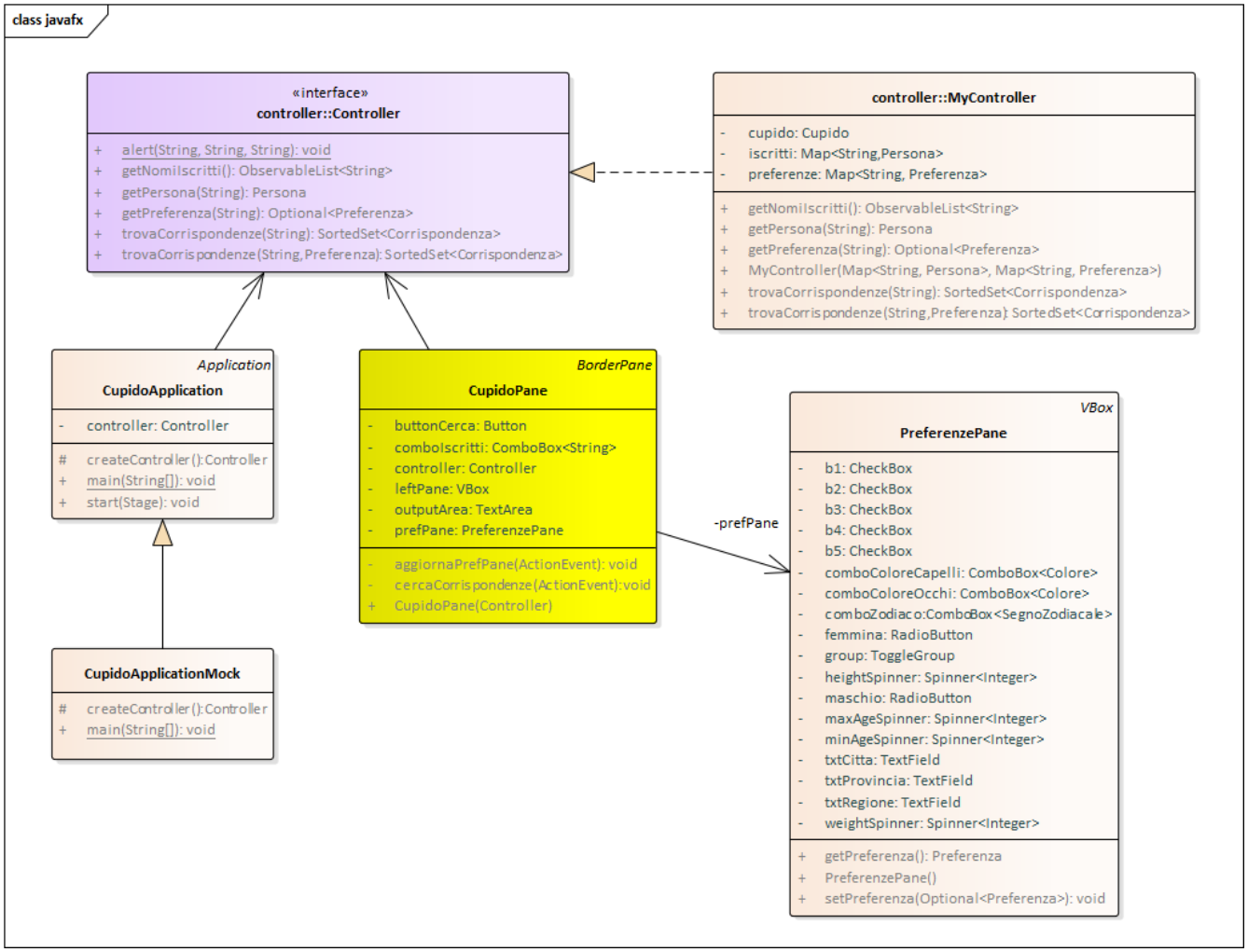
Essa fornisce inoltre il metodo statico ***alert*** che fa comparire una finestra di dialogo all'utente, utile per segnalare errori: i tre argomenti rappresentano il titolo della finestra, l'header e il testo del messaggio (v. Figg. 5 e 6).

La **classe *MyController* (pure fornita)** implementa ***Controller*** memorizzando internamente gli iscritti e le preferenze.

Interfaccia utente (namespace cupido.ui.javafx) *(punti 9)*

L'interfaccia utente deve essere simile (non necessariamente identica) all'esempio mostrato nelle figure seguenti.

La classe ***PreferenzePane*** **(fornita)**, offre **già pronto** un completo pannello per mostrare/leggere una preferenza utente, tramite i due metodi ***getPreferenza*** / ***setPreferenza*** (v. Figg. 1-4, parte sinistra).



La classe *CupidoPane* (da realizzare), che estende ***BorderPane***, deve prevedere (v. Fig. 1):

- una combo box per la selezione delle persone;
- un pulsante *Cerca* per scatenare la ricerca delle corrispondenze;
- sotto a questo, un pannello di tipo ***PreferenzePane;***
- sulla destra, una textarea in cui mostrare i risultati

Il formato delle stampe (v. figure) è quello prodotto dalla ***toString*** di ***Corrispondenza***, una per riga, completato da una frase finale che informa che non vi sono altre corrispondenze: tale frase è l'unica mostrata nel caso non vi sia alcuna corrispondenza.

All'inizio la combo è popolata con i nickname di tutte le persone disponibili, **più il valore "NON ISCRITTO" che dev'essere posto all'inizio** (Fig. 1). Se si sceglie una persona iscritta, il pannello preferenze sottostante ne mostra la preferenza (Fig. 2) [infatti, in questa modalità il pannello ***PreferenzePane*** viene usato solo come dispositivo di output]: quando l'utente preme il pulsante *cerca,* l'applicazione cerca le corrispondenze possibili e le mostra in ordine decrescente di indice di compatibilità (Fig. 3): se non ve ne sono, viene comunque mostrata la frase finale, per indicare che l'applicazione ha già operato e terminato (Fig. 4).

In alternativa, si può cercare corrispondenze per un **NON ISCRITTO**: in tal caso l'applicazione utilizzerà l'insieme di preferenze specificato dall'utente nel pannello ***PreferenzePane***, inserendo in ogni campo i valori desiderati (Fig. 5): da notare che *i campi opzionali devono essere singolarmente abilitati*, altrimenti non saranno inclusi nella preferenza e quindi neppure nel successivo calcolo delle corrispondenze.

In caso di problemi (mancanza di uno o dell'altro file, errori di lettura, problemi di formato, etc.) l'applicazione deve mostrare opportuni dialoghi, sfruttando il metodo statico ***Controller.alert*** (Fig. 6).

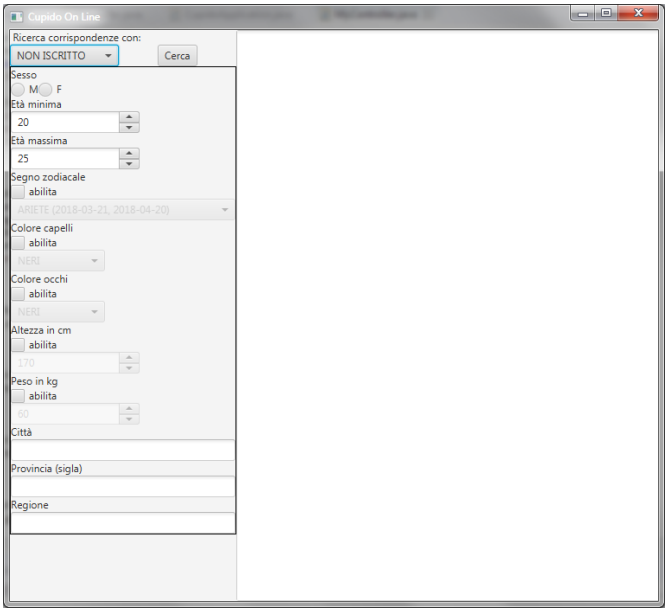


Figura 1

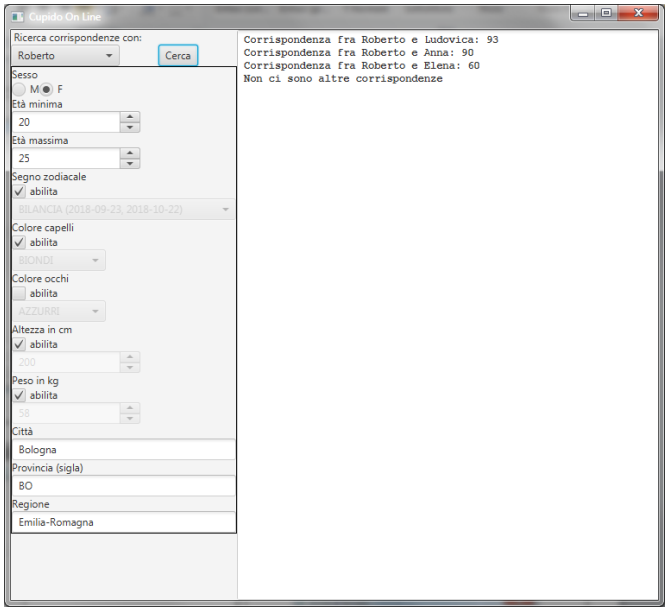


Figura 2

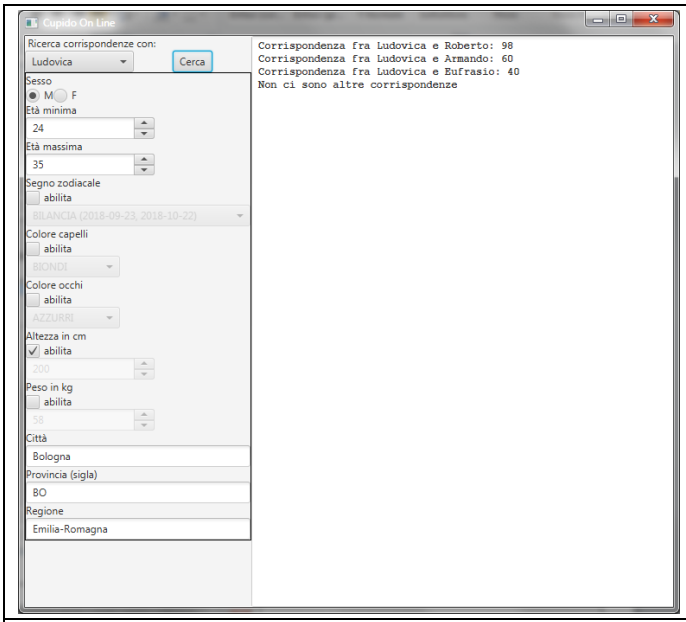


Figura 3

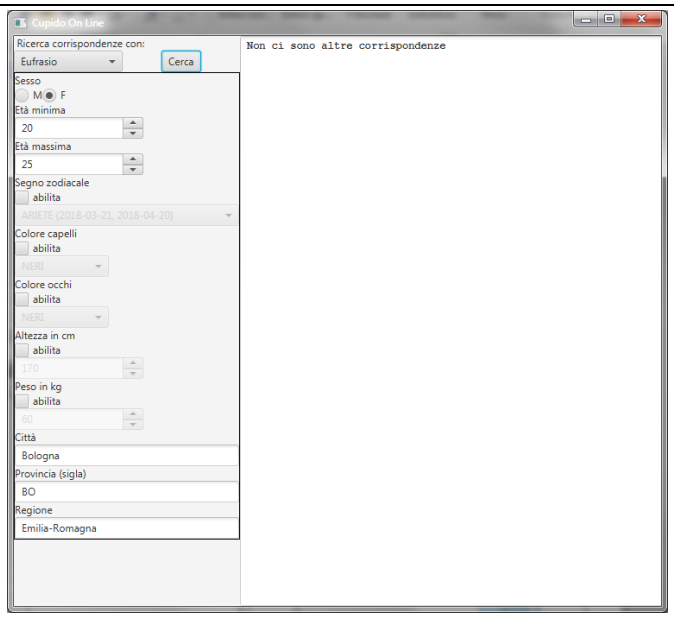


Figura 4

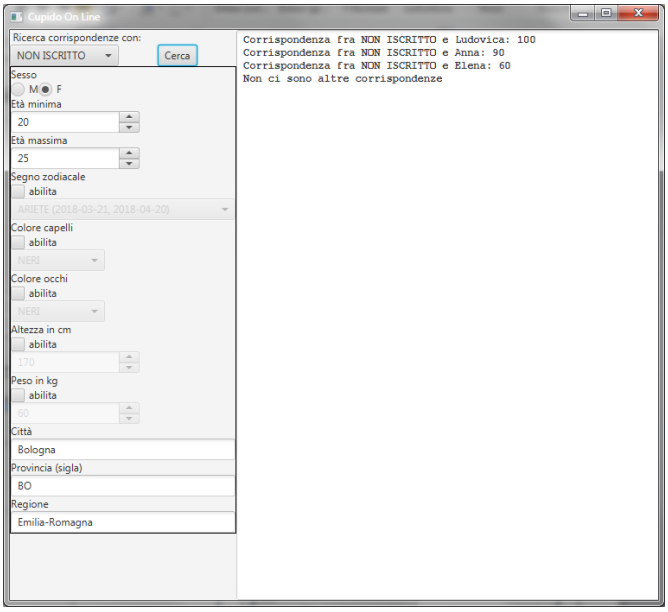


Figura 5

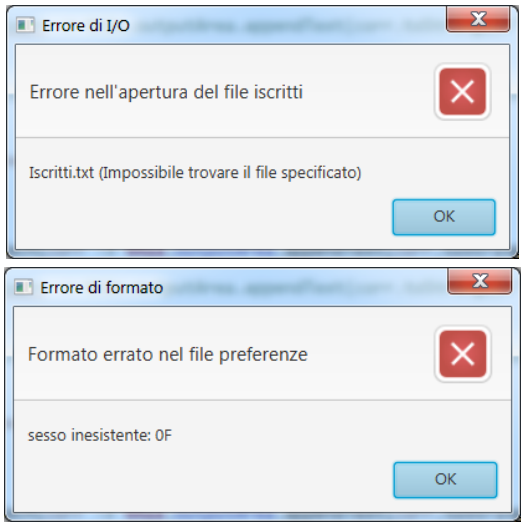


Figura 6