ESAME DI FONDAMENTI DI INFORMATICA T-2 del 12/07/2011
Proff. E. Denti – G. Zannoni
Tempo a disposizione: 4 ore MAX

NB: il candidato troverà nell'archivio ZIP scaricato da Esamix anche il software "Start Kit"

NOME PROGETTO ECLIPSE: CognomeNome-matricola (es. RossiMario-0000123456)

La compagnia *ED&GZ Weather*, attivissima sul mercato digitale, ha richiesto un'applicazione per la diffusione di informazioni meteo con caratteristiche particolarmente innovative.

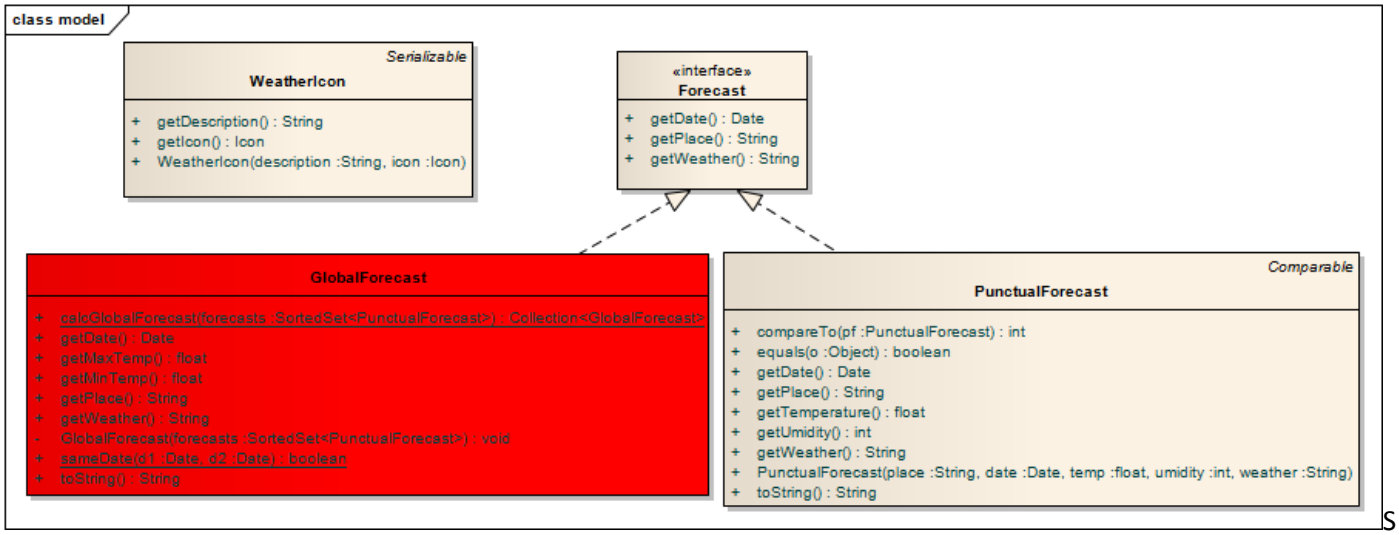
DESCRIZIONE DEL DOMINIO DEL PROBLEMA.
Una *previsione meteo puntuale* è costituita da un insieme di dati (orario di riferimento, temperatura, grado di umidità, precipitazioni) riferiti a una certa località a un dato orario per un dato giorno. Per praticità sono però spesso fornite *previsioni meteo sintetiche,* mirate a dare un'idea d'insieme del tempo previsto per l'intera giornata, che accorpano i dati puntuali riportando semplicemente, in forma aggregata, la temperatura minima, la temperatura massima, il grado di umidità medio e il tempo atteso per quella località; quest'ultimo è espresso sotto forma di icona (sereno, variabile, nuvoloso, pioggia, neve).
Poiché le previsioni puntuali sono tipicamente calcolate a intervalli di 3-6h (nel caso più frequente: alle ore 2, ore 5, ore 8, ore 11, ore 14, ore 17, ore 20, ore 23), le previsioni sintetiche estraggono i loro dati da un insieme di 4-8 previsioni puntuali: il tempo globalmente atteso è quindi ottenuto come media pesata nelle 24 ore dei tempi attesi puntuali, secondo opportuni coefficienti – ad esempio , una giornata serena per 9h, con pioggia per 3h e nuvole per 12h sarà probabilmente sintetizzata dall'icona "variabile".

Il file di testo Forecasts.txt contiene le previsioni puntuali per alcuni giorni per serie di località, nel formato più oltre specificato. Il file binario Icons.dat contiene invece una serie di icone (istanze di ***WeatherIcon***) utili a rappresentare il tempo atteso, in forma di mappa: la chiave di ogni entry è una stringa univoca con il nome descrittivo dell'icona ("sereno", "variabile", "nuvoloso", "pioggia", "neve", "tempesta", "nebbia"), il cui valore è l'icona corrispondente.

Parte 1 (punti: 16)

Dati (namespace weather.model) *(punti: 8)*

Il modello dei dati relativo all'esame deve essere organizzato secondo il diagramma UML riportato a pagina 2.



S

EMANTICA:

a) la classe ***WeatherIcon*** (fornita nello Start Kit) rappresenta una icona del tipo memorizzato nella mappa fornita nel file binario

b) l'interfaccia ***Forecast*** (fornita nello Start Kit) è implementata dalle due classi ***PunctualForecast*** e ***GlobalForecast***;

c) la classe ***PunctualForecast*** (fornita nello Start Kit) rappresenta una previsione meteo puntuale per un dato giorno e orario in una data località, ed è caratterizzata dalle proprietà sopra elencate; fornisce opportuni metodi accessor, una ***toString*** e una ***compareTo*** idonee, e quant'altro ritenuto necessario per l'applicazione;

d) la classe ***GlobalForecast*** **(da realizzare)** rappresenta una previsione meteo globale per un dato giorno in una data località, ed è sintetizzata a partire dalle previsioni puntuali per tale località nel giorno dato. **Le istanze di questa classe sono create solo ed esclusivamente da un opportuno metodo di fabbrica *calcGlobalForecasts*: pertanto, l'unico costruttore deve imperativamente essere privato.**

Il metodo di fabbrica ***calcGlobalForecasts***, data una collezione di ***PunctualForecast*** relativa a più giorni ma ad una sola città, crea un'opportuna collezione di ***GlobalForecast*** contenente esattamente una e una sola istanza di ***GlobalForecast*** per ogni giorno per il quale esiste almeno una ***PunctualForecast***. Il costruttore privato a sua volta riceve come argomento un'opportuna collezione di ***PunctualForecast*** relativa a un unico giorno e, tramite opportuni calcoli, sintetizza le proprietà globali richieste.

In particolare, il tempo globale riferito alla giornata è calcolato calcolando la media, pesata in base al numero delle ore corrispondenti, del tempo atteso nelle diverse fasce orarie, arrotondata al valore più prossimo, utilizzando i seguenti coefficienti: *1=neve, 2=tempesta, 3=pioggia, 4=nebbia, 5=nuvoloso, 6=variabile, 7= sereno*.

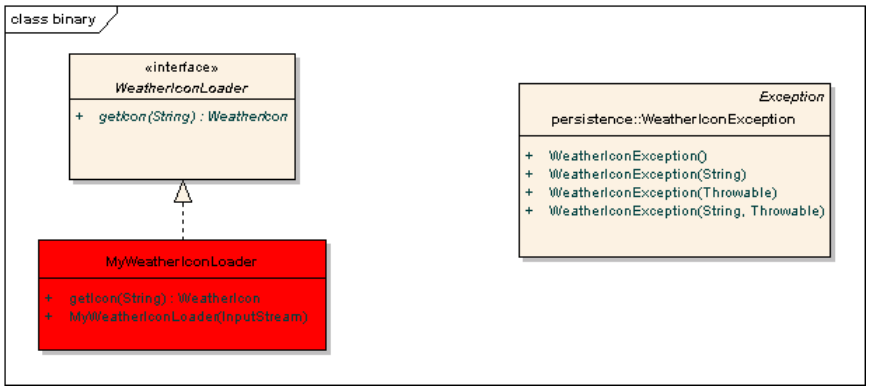
[Ad esempio, se nell'arco della giornata sono previste 3h di pioggia, 3h di nuvole, 6h di sereno, 6h di variabile, e altre 6h di sereno nella notte, il calcolo sarà (3x3+3x5+6x6+12x7)/24 = 6 → variabile.]

Attenzione: potrebbero non esistere previsioni puntuali sufficienti a coprire tutto l'intero arco della giornata, nel qual caso la media dovrà essere calcolata sulle sole ore per cui esistono previsioni. Ad esempio, se la prima previsione puntuale è alle 5 (di mattina), mentre altre ne seguono fino verso a sera, la previsione globale andrà calcolata su 19 ore (dalle 5 alle 24). Va da sé che l'ultima previsione puntuale copre l'arco di tempo tra l'orario indicato e la mezzanotte: così, se ultima previsione è alle 20, s'intende che essa copre le 4h dalle 20 alle 24.

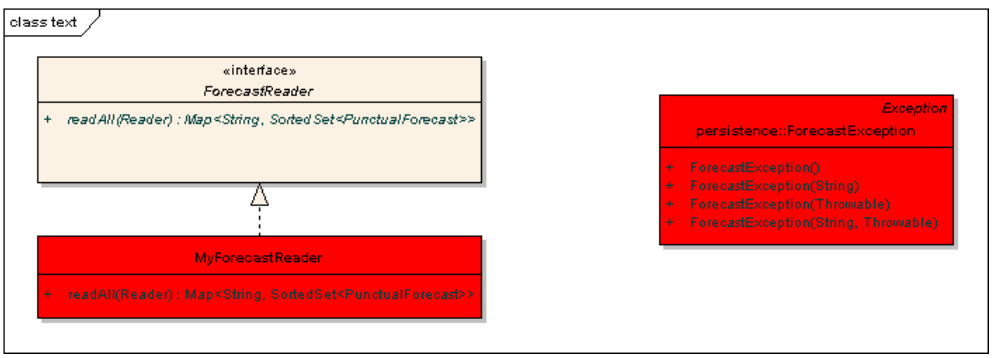
Le proprietà della previsione così calcolata sono accessibili tramite i metodi ***getMinTemp***, ***getMaxTemp*** e ***getWeather***: quest'ultimo restituisce una stringa descrittiva del tempo globale del giorno (Es: "sereno" o "nebbia").

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

Persistenza (namespace weather.persistence.binary/text) *(punti 8)*



1) Il file binario Icons.dat contiene serializzata una mappa che permette di recuperare l'opportuna istanza della classe ***WeatherIcon*** usando come chiave di accesso una opportuna stringa (che distingue fra maiuscole e minuscole). In particolare l'interfaccia ***WeatherIconLoader*** (fornita nello Start Kit) definisce il metodo getIcon che permette di recuperare la ***WeatherIcon*** corrispondente alla stringa data (ad esempio, getIcon("sereno") restituirà l'icona del sole). ***WeatherIconLoader*** dev'essere implementata da ***MyWeatherIconReader*** (**da realizzare**): il costruttore dovrà leggere la mappa dal file, lanciando ***IOException*** in caso di problemi di accesso al file o ***WeatherIconException*** (fornita nello Start Kit) in caso di problemi di formato sul file.



2) Come già anticipato, il file di testo Forecasts.txt contiene le previsioni puntuali dei prossimi giorni per una serie di località. Ogni riga contiene una previsione puntuale, sotto forma di elenco, *separato da "punti e virgola"*, dei seguenti dati: *nome della città, data del giorno a cui la previsione si riferisce (nella forma* `gg/mm/aaaa`*), orario a cui la previsione si*

riferisce (nella forma hh.mm), temperatura prevista (numero e simbolo di grado "°"), *grado di umidità (numero e simbolo di percentuale, "%")*, tempo previsto (una stringa "sereno", "variabile", "nuvoloso", "nebbia", "pioggia", "tempesta", "neve"); maiuscole e minuscole sono considerate differenti.

ESEMPI

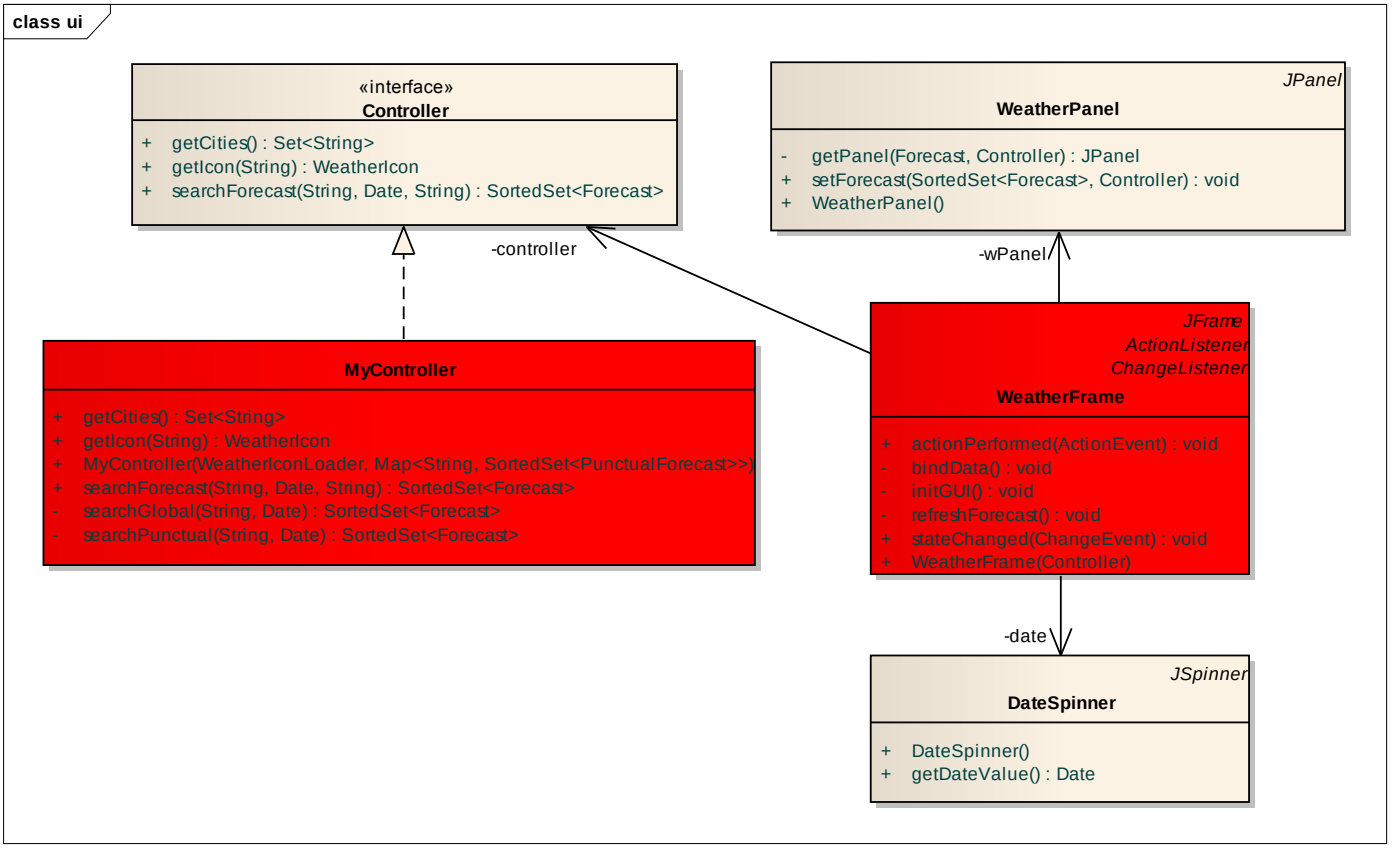
```
Bologna; 13/11/2011; 02.00; 18°; 91%; pioggia
Bologna; 13/11/2011; 05.00; 16°; 92%; nuvoloso
Bologna; 13/11/2011; 08.00; 20°; 87%; sereno
Bologna; 13/11/2011; 11.00; 24°; 64%; sereno
Bologna; 13/11/2011; 14.00; 27°; 58%; variabile
...
```

L'interfaccia ***ForecastReader*** (fornita nello Start Kit) dichiara i metodi utili per leggere un insieme di ***Punctual-Forecast*** e dev'essere implementata da ***MyForecastReader*** (**da realizzare**); quest'ultima dovrà lanciare ***IOException*** in caso di problemi di accesso al file, o ***ForecastException*** (**da realizzare**) in caso di problemi di formato sul file. Il contratto stabilito da ***ForecastReader*** prevede che venga restituita una mappa avente per chiave il nome della città e come valore la lista di ***PunctualForecast*** di tale città, ordinata per giorno e orario in senso ascendente.

Lo Start Kit contiene la classe di utilità utils.Month per la corretta impostazione dei mesi
Lo Start Kit contiene anche i test relativi alle classi MyForecastReader e MyWeatherIconLoader.

Parte 2 (punti: 14)

Controller (namespace weather.ui) *(punti 7)*



La classe ***MyController*** (**da realizzare**) implementa l'interfaccia ***Controller*** (fornita nello Start Kit); in particolare il metodo ***getCities*** restituisce l'elenco delle città, mentre ***getIcon*** restituisce l'icona corrispondente alla stringa passata come argomento: entrambi sfruttano a tal fine la mappa caricata dal file binario tramite il ***WeatherIconLoader*** ricevuto dal costruttore.

Il metodo ***searchForecast*** cerca e restituisce le previsioni (puntuali o globale) per la città e la data indicate, sotto forma di collezione di ***Forecast***; nel caso della previsione globale, la collezione restituita contiene un solo elemento (istanza di ***GlobalForecast***).

Lo Start Kit contiene i test relativi alla classe MyController

Interfaccia utente (namespace weather.ui) *(punti 7)*

L'interfaccia utente deve essere simile (non necessariamente identica) all'esempio mostrato nella figura che segue. La classe ***WeatherFrame*** (**da realizzare**) realizza la finestra principale, che consente di cercare la previsione puntuale o globale per una data città in una certa data. A tal fine, una lista a discesa (casella combinata) presenta le città disponibili, mentre un componente ***DateSpinner*** (fornito nello Start Kit) consente di scegliere la data richiesta (Fig. 1);

poiché per non tutte le città potrebbero essere disponibili previsioni per tutte le date, l'applicazione deve segnalare all'utente eventuali casi in cui sia impossibile soddisfare la richiesta, mediante idoneo messaggio (Fig. 2). Una coppia di radiobutton consente di scegliere fra previsioni puntuali o previsione globale, che vengono mostrate ciascuna nel modo più adatto (Fig. 3 / Fig. 4).

Non appena di seleziona il radiobutton, l'applicazione mostra nell'apposita area (un pannello di tipo ***WeatherPanel*** fornito nello Start Kit) le previsioni richieste, tramite il metodo ***searchForecast*** del controller.

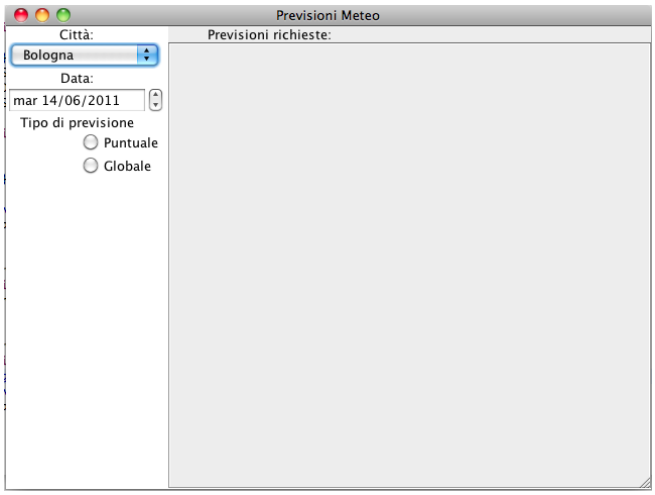


Fig. 1

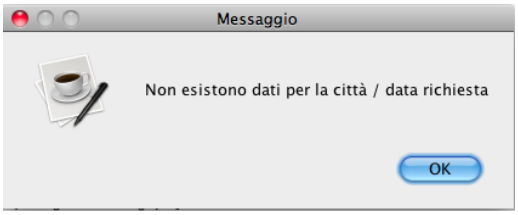


Fig. 2

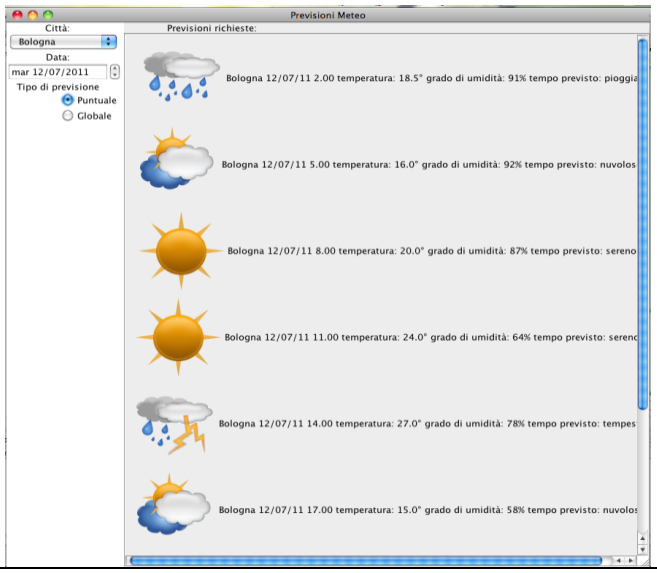


Fig. 3

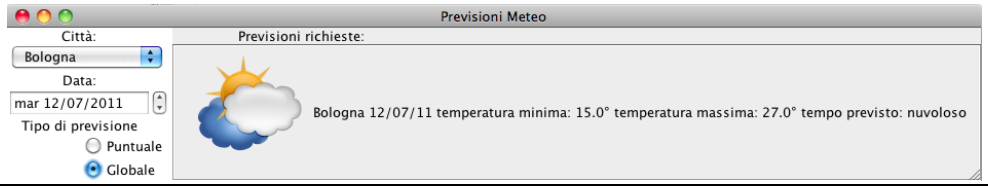


Fig. 4