ESAME DI FONDAMENTI DI INFORMATICA T-2 del 13/09/2011
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 *EZCasa*, attiva sul mercato immobiliare, ha richiesto un'applicazione per la ricerca di case in vendita o affitto, con caratteristiche moderne e innovative.

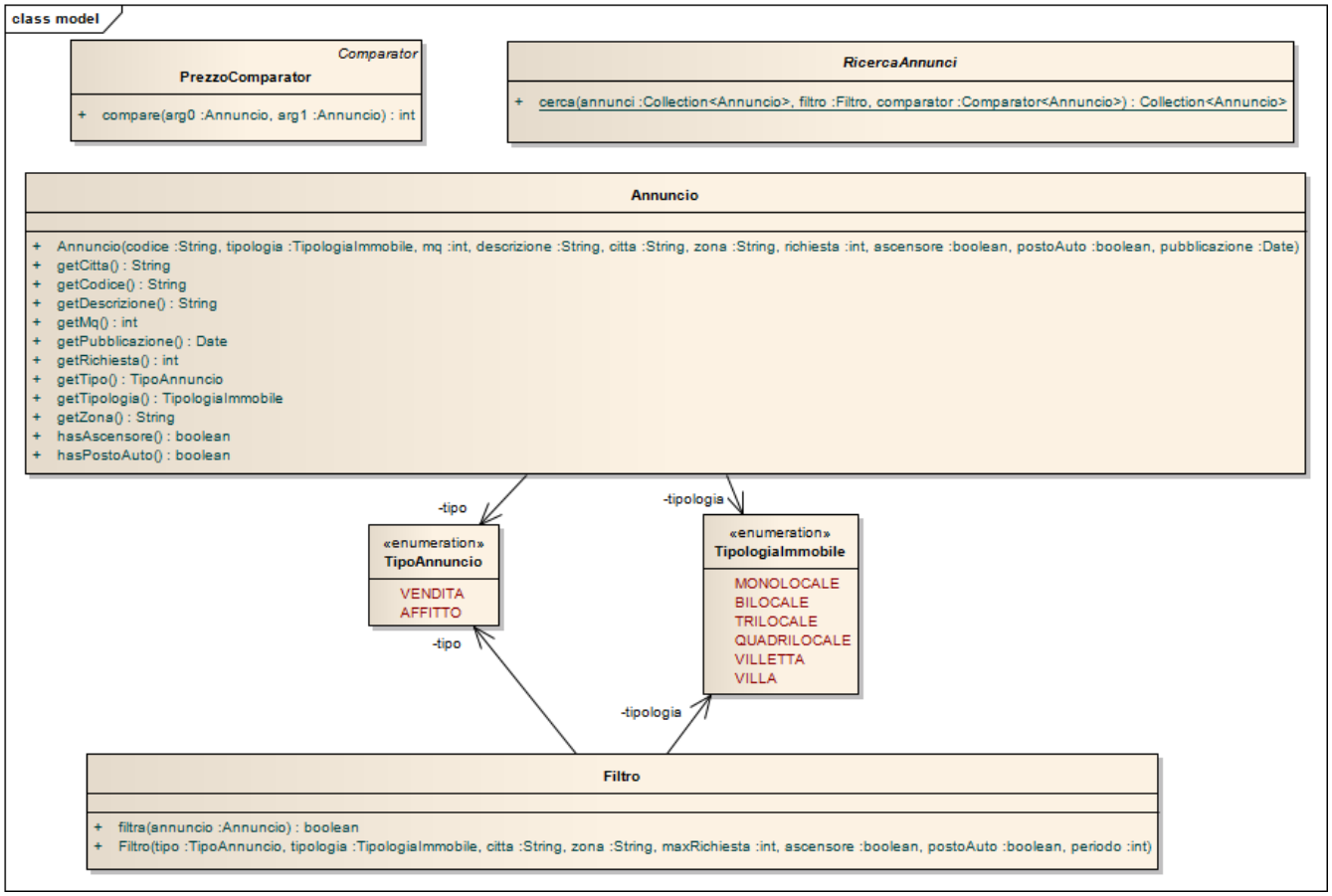
DESCRIZIONE DEL DOMINIO DEL PROBLEMA.
Un *annuncio immobiliare* è costituito da un insieme di dati (codice univoco dell'immobile, tipologia dell'immobile, metratura, descrizione, zona, città, prezzo richiesto, presenza di ascensore, presenza di posto auto) ed è pubblicato in un dato giorno; gli *annunci di vendita* differiscono dagli *annunci di affitto* solo per la forma del codice univoco (che inizia per 'A' nel caso degli affitti e per 'V' nel caso delle vendite) e per l'interpretazione del prezzo richiesto, che in un caso è il prezzo di vendita e nell'altro il canone mensile.
La *tipologia di immobile* può essere villa, villetta, quadrilocale, trilocale, bilocale, monolocale.

Il file di testo annunci.txt contiene gli annunci pubblicati, nel formato più oltre specificato.

Parte 1 (punti: 16)

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

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



SEMANTICA:

a) il tipo enumerativo ***TipologiaImmobile*** (fornito) definisce le tipologie previste dal dominio;

b) il tipo enumerativo ***TipoAnnuncio*** (fornito) definisce i due tipi (vendite/affitti) di annunci immobiliari;

c) la classe ***Annuncio*** **(da realizzare)** è caratterizzata dall'insieme di proprietà sopra dettagliate; forniscono opportuni metodi accessor, una ***toString*** e una ***compareTo*** idonee, e quant'altro ritenuto necessario per

l'applicazione. È compito del costruttore identificare il tipo di annuncio a partire dal formato del codice univoco, impostando conseguentemente la corrispondente proprietà.

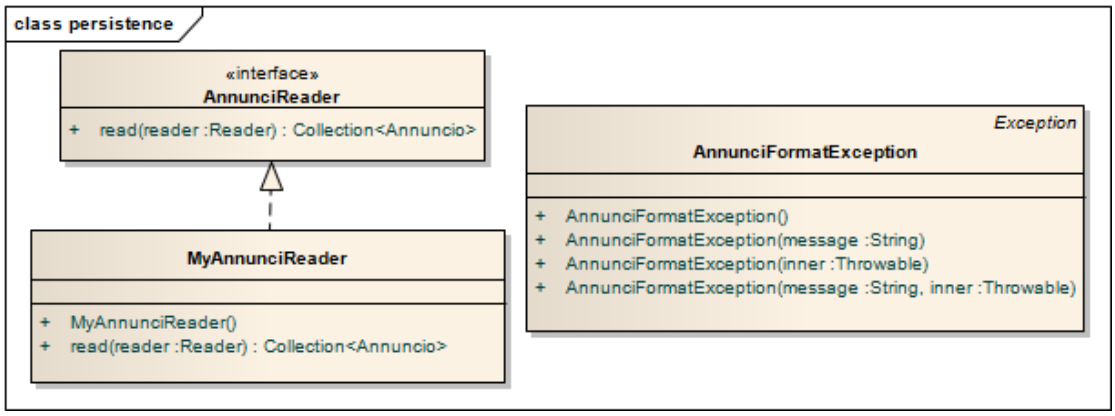
d) la classe ***Filtro*** **(da realizzare)** rappresenta l'insieme di dati su cui può essere parametrizzata la ricerca: il costruttore riceve in ingresso nell'ordine un valore di ***TipoAnnuncio***, un valore di ***TipologiaImmobile***, due stringhe che rappresentano rispettivamente la città e la zona di interesse, un prezzo *da intendersi come massimo* che si è disposti a pagare (-1 = qualunque prezzo), due valori ***boolean*** che rappresentano rispettivamente la richiesta di ascensore e posto auto (true=richiesto, false=indifferente) e un ***int*** che rappresenta il periodo di interesse (ovvero, quanti giorni indietro rispetto alla data attuale cercare annunci; -1 = indifferente). Il primo argomento dev'essere necessariamente istanziato: tutti gli altri possono essere impostati a valori indifferenti (*null* per gli oggetti, -1 per gli interi, false per i boolean) per indicare che quei parametri non devono intervenire nel filtro.

   La classe espone un unico metodo ***filtra*** che, dato un ***Annuncio***, restituisce un boolean che esprime se l'annuncio soddisfa le condizioni del filtro.

e) la classe ***RicercaAnnunci*** **(da realizzare)** riassume in sé le funzionalità di ricerca, incapsulate nel metodo ***cerca***: esso prende in ingresso una collection di ***Annuncio***, un ***Filtro*** e un ***Comparator<Annuncio>***, e restituisce un'altra collection ordinata di ***Annuncio*** che contiene i soli annunci che soddisfano il filtro, ordinati in senso ascendente secondo il criterio di ordinamento espresso dal comparatore.

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

Persistenza (namespace ezc.persistence) *(punti 8)*



Come già anticipato, il file di testo annunci.txt contiene gli annunci immobiliari per una serie di località. Ogni riga contiene un singolo annuncio, costituito da una serie di campi *case insensitive* separati da ":" e precisamente: data di pubblicazione dell'annuncio, codice univoco dell'immobile (alfanumerico), tipologia dell'immobile ("`villa`", "`villetta`", "`quadrilocale`", "`trilocale`", "`bilocale`", "`monolocale`"), metratura, descrizione (stringa libera priva di ":"), zona, città, prezzo, ed eventualmente presenza di ascensore ("`ASC`") e/o posto auto ("`PA`").

ESEMPI

```
11/09/2011 : V25883634: trilocale: 90: tinello, cucinotto, due camere, bagno, due balconi: saragozza: bologna: 230000
10/09/2011 : V25883639: trilocale: 85: 4 piano, luminoso, ristrutturato, ingr. Su soggiorno con ang. Cottura, 2 camere, bagno, cantina: saragozza: bologna: 290000 : ASC
02/09/2011 : V5489612: trilocale: 90: trilocale con cucina, soggiorno, due camere, bagno: ospizio: reggio emilia: 120000
10/09/2011 : V5489014: bilocale: 70: app. composto da soggiorno con angolo cottura, camera, bagno: centro: reggio emilia: 100000
08/09/2011 : A5422014: bilocale: 70: soggiorno con angolo cottura, camera a due letti, bagno, spese escluse: centro: reggio emilia: 480
```

L'interfaccia ***AnnunciReader*** (fornita) dichiara i metodi utili a leggere un insieme di ***Annunci***, implementata poi da ***MyAnnunciReader*** (**da realizzare**); quest'ultima dovrà lanciare ***IOException*** in caso di problemi di accesso al file, o ***AnnunciFormatException*** (fornita) in caso di problemi di formato sul file. Il contratto stabilito da ***AnnunciReader*** prevede un metodo ***read*** che, dato un ***Reader***, restituisce una lista di ***Annunci***.

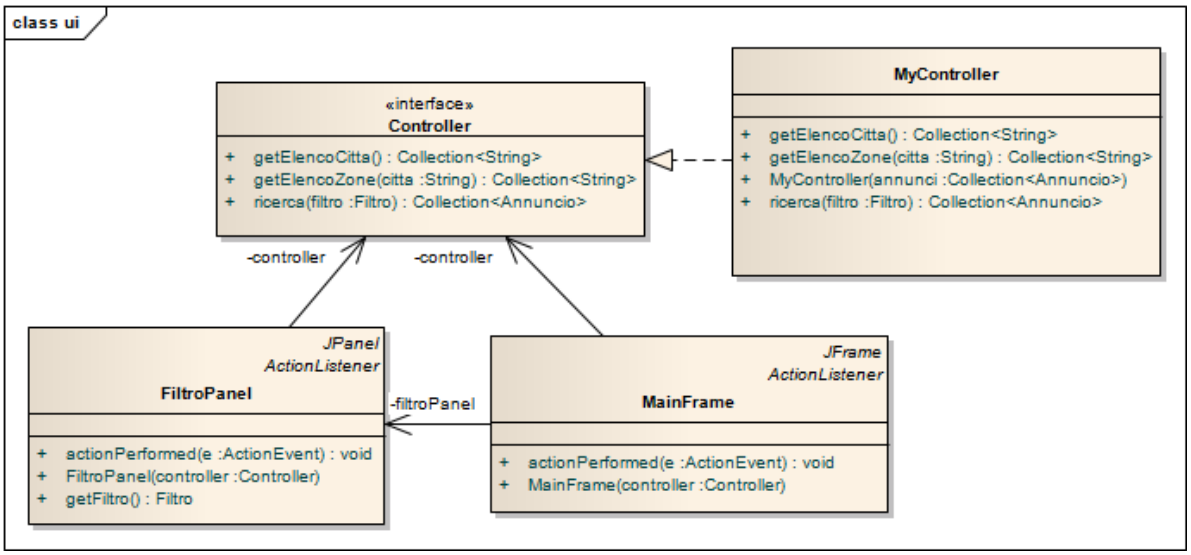
Lo Start Kit contiene anche i test (da includere nel progetto) per verificare il funzionamento di questa classe.

Parte 2 (punti: 14)

Controller (namespace ezc.ui) *(punti 6)*

La classe ***MyController*** (**da realizzare**) implementa l'interfaccia ***Controller*** (fornita): in particolare il costruttore prende in ingresso la collection di tutti gli ***Annuncio*** mentre il metodo ***ricerca*** prende in ingresso un filtro e restituisce la collection di ***Annuncio*** che soddisfano il filtro. Altri metodi degni di nota sono ***getElencoCitta*** e ***getElencoZone*** che restituiscono rispettivamente la collezione (ordinata alfabeticamente) della città per le quali esistono annunci, e la collezione (ordinata alfabeticamente) delle zone della città specificata.

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



Interfaccia utente (namespace ezc.ui) *(punti 8)*

L'interfaccia utente deve essere simile (non necessariamente identica) all'esempio mostrato nella figura che segue. La classe ***MainFrame*** (**da realizzare**) realizza la finestra principale, ed è articolata in tre parti principali: un pannello (***FiltroPanel***) contenente i controlli per la realizzazione del filtro, un pulsante che consente di effettuare la ricerca, un'area di testo dove mostrare i risultati e che consente di cercare annunci di affitto o vendita in base a uno o più dei dati presenti negli annunci stessi. A tal fine, occorre preliminarmente scegliere fra affitti e vendite dalla combobox in alto: tutti gli altri criteri sono invece singolarmente attivabili/disattivabili tramite una checkbox (Fig. 1).

La selezione di una città produce il ripopolamento della lista delle zone: all'inizio tale combobox è popolata a partire dalla prima città della combobox precedente.

Il pannello ***FiltroPanel*** (**da realizzare**) deve essere realizzato come da diagramma UML: il metodo ***getFiltro*** restituisce ogni volta un'istanza di Filtro configurata in coerenza con i controlli attivati in quel momento.

La ricerca è attivata dalla pressione del pulsante CERCA: il risultato, ottenuto tramite il metodo ***ricerca*** del controller (che prende in ingresso il ***Filtro*** restituito da ***FiltroPanel***), è mostrato nell'apposita area di testo. Ovviamente, se nessun criterio è attivato, vengono mostrati tutti gli annunci (Fig. 2); altrimenti, vengono considerati solo i criteri attivati (Figg. 3 e 4). In particolare, se è specificato un numero di giorni, vengono considerati solo gli annunci non più vecchi di *tot* giorni rispetto ad oggi (Fig. 4).

Eventuali valori negativi o nulli nel campo "Max Richiesta" devono essere ignorati, come se esso non fosse stato compilato (caso non mostrato nelle figure).

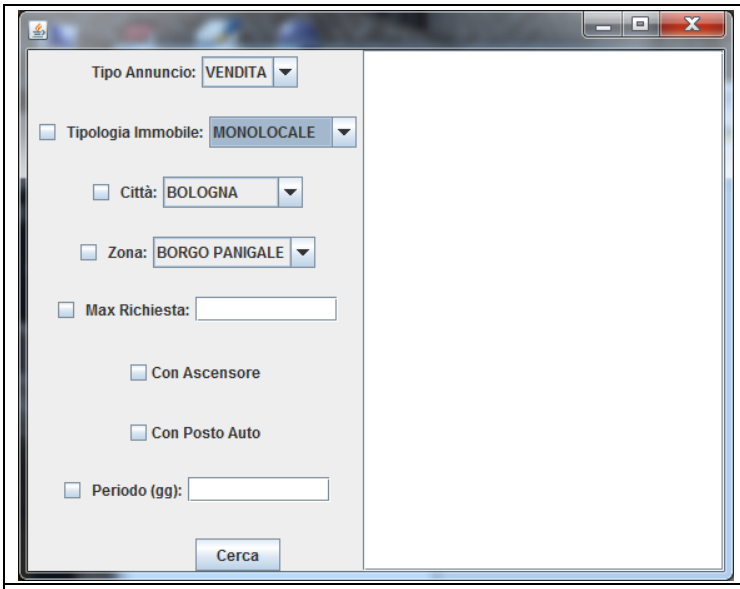


Fig. 1

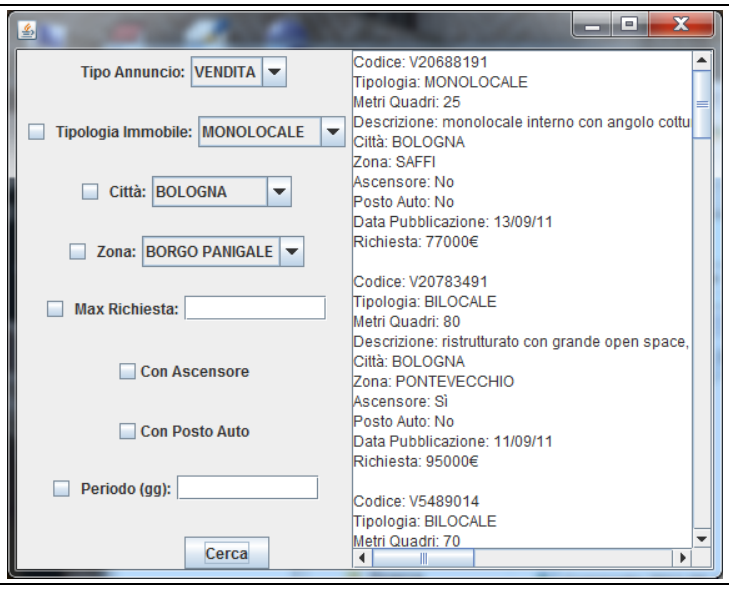


Fig. 2

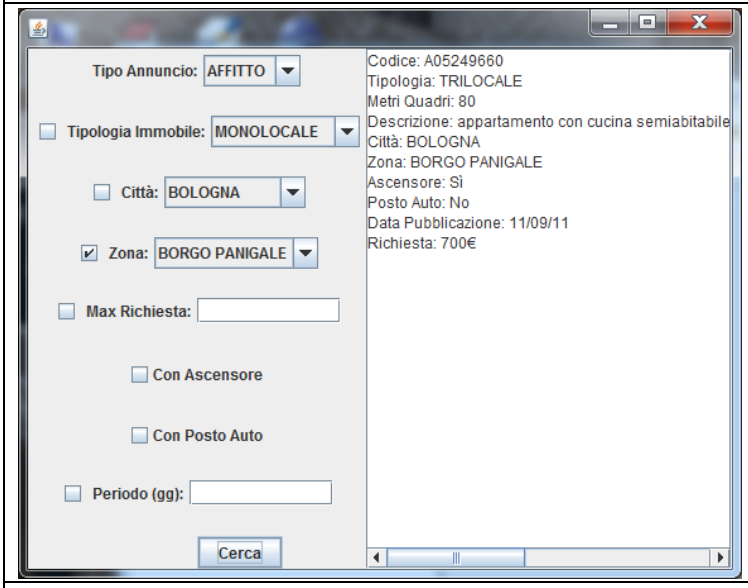


Fig. 3

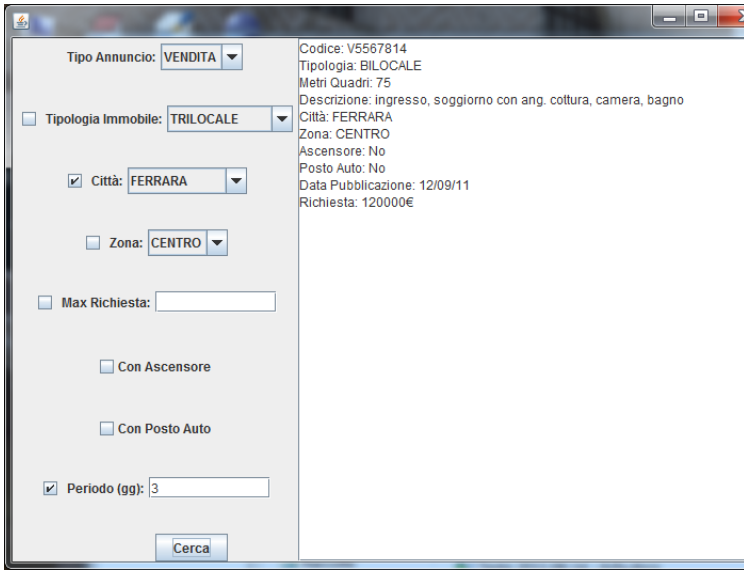


Fig. 4