ESAME DI FONDAMENTI DI INFORMATICA T-2 del 13/1/2023
Proff. E. Denti – R. Calegari – A. Molesini
Tempo a disposizione: 3h30

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)
NOME JAR DA CONSEGNARE: CognomeNome-matricola.jar (es. RossiMario-0000123456.jar)

Si devono consegnare DUE FILE: *l'intero progetto Eclipse* e *il JAR eseguibile*

Si ricorda che compiti *non compilabili* o *palesemente lontani da 18/30* NON SARANNO CORRETTI e causeranno la verbalizzazione del giudizio "RESPINTO"

Per incentivare gli studenti a procedere con celerità, nell'università SpeedCollege i voti ottenuti nei vari esami subiscono un *processo di decadimento* al passare del tempo: la legge con cui tale decadimento avviene è personalizzabile dal singolo corso di studio. A tutela dello studente è tuttavia stabilito che nessun voto positivo possa scendere sotto quota 18/30, che costituisce quindi la soglia minima, raggiunta la quale il decadimento si arresta e il voto rimane costante.

Si vuole sviluppare un'applicazione che consenta di seguire la carriera degli studenti al passare del tempo, *calcolando e mostrando i voti e la media pesata validi in un dato giorno*, oltre ai crediti globalmente acquisiti.

DESCRIZIONE DEL DOMINIO DEL PROBLEMA

Un'***attività formativa*** rappresenta un insegnamento universitario caratterizzato da *numero di codice* (univoco), *denominazione* (che può contenere spazi) e *numero di crediti* (non necessariamente intero).

Quando, **in una certa *data***, lo studente sostiene un ***esame***, all'attività formativa viene associato un ***voto iniziale,*** che può essere o un *numero fra 18 e 30* (quest'ultimo eventualmente con lode) o un giudizio di *idoneità*; in caso di esito negativo, viene attribuito uno dei giudizi *ritirato* o *respinto*. Ogni esame superato comporta l'acquisizione dei relativi *crediti*. È possibile ri-sostenere un esame solo se in precedenza l'esito è stato negativo, mentre non è consentito ripetere un esame già sostenuto con esito positivo. Al termine della carriera, lo studente deve sostenere la ***prova finale***: essa può quindi essere sostenuta solo *in data successiva* ad ogni altro esame.

In qualunque momento la carriera dello studente è quindi caratterizzata da:
- **Lista degli esami sostenuti** (sia con esito positivo che con esito negativo)
- **Crediti acquisiti** (concorrono all'acquisizione dei crediti solo gli esami superati con *esito positivo*)
- **Media pesata** (concorrono all'acquisizione dei crediti solo gli esami con voto, superati con *esito positivo*)

La media pesata **al giorno d** si calcola secondo la formula $MP(d) = \frac{\sum v_i(d) * p_i}{\sum p_i}$, essendo $v_i(d)$ **i voti ricalcolati al giorno d**, e p_i i pesi (crediti) degli esami.
La legge con cui i voti vengono ricalcolati (diminuendoli) al passare del tempo è personalizzabile: il voto così calcolato può anche essere non intero (es. 25.5/30), per evitare eccessive distorsioni.

Una serie di file di testo (i cui nomi non sono specificati né rilevanti) descrive possibili carriere di studenti**,** nel formato descritto più oltre.

TEMPO STIMATO PER SVOLGERE L'INTERO COMPITO: 2h15 – 3h

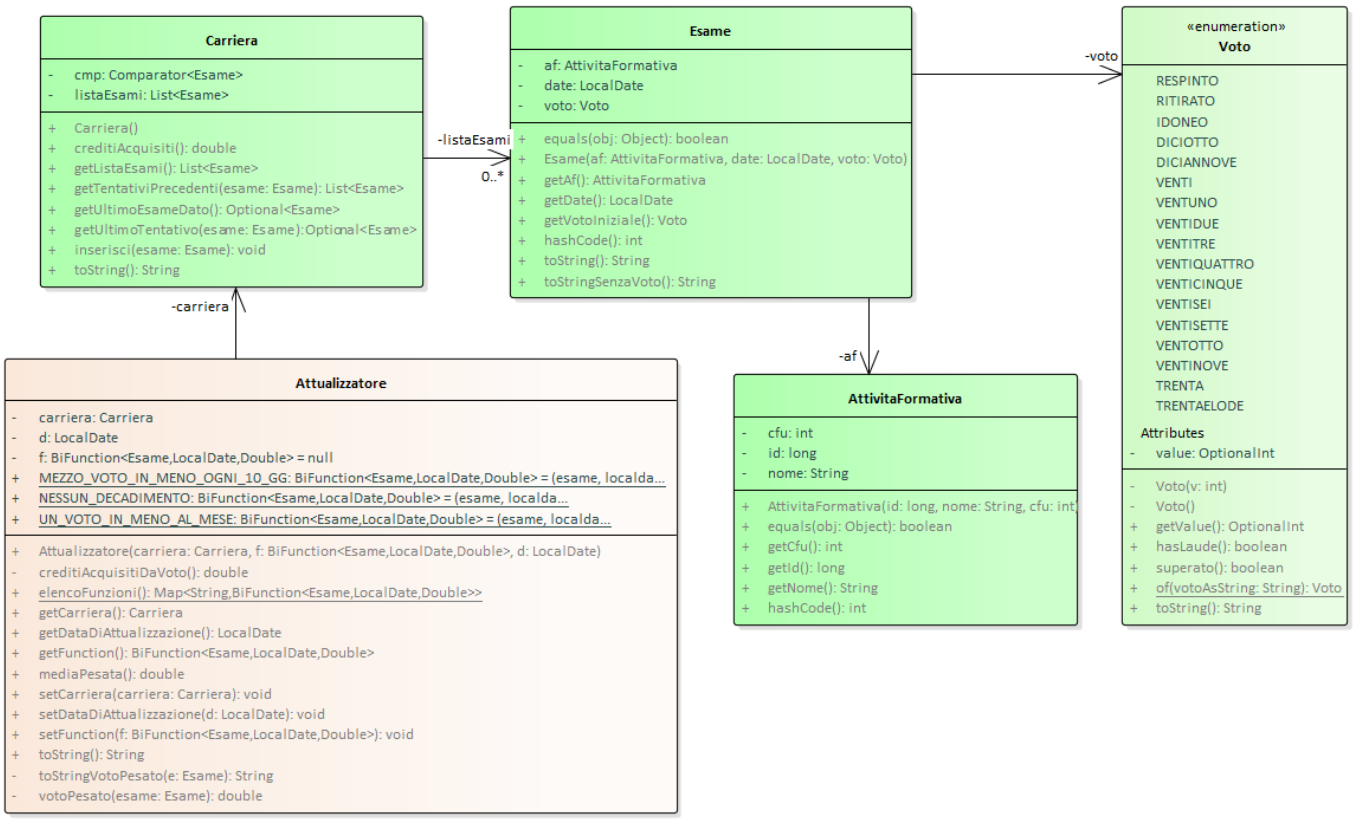
PARTE 1 – Modello dei dati: Punti 13 [TEMPO STIMATO: 60-70 minuti]
PARTE 2 – Persistenza: Punti 9 [TEMPO STIMATO: 40-60 minuti]
PARTE 3 – Grafica: Punti 8 [TEMPO STIMATO: 35-50 minuti]

JAVAFX – Parametri run configuration nei LAB

```
--module-path "C:\applicativi\moduli\javafx-sdk-17.0.2\lib"
--add-modules javafx.controls
```

Parte 1 – Modello dei dati (punti: 13)

Package: speedycollege.model **[TEMPO STIMATO: 60-70 minuti]**



SEMANTICA:

a) la classe ***AttivitaFormativa*** (fornito) descrive un'attività formativa come da dominio del problema

b) l'enumerativo ***Voto*** (fornito) rappresenta i possibili valori di voto o giudizio, come da dominio del problema; in particolare, sono definite sia costanti enumerative per ogni voto fra 18 e 30L, associate al corrispondente valore numerico (per 30L il valore è 30), nonché costanti enumerative per i tre giudizi respinto, ritirato e idoneo, non associate ad alcun valore numerico. Il metodo *getValue* estrae il valore intero associato, se esistente, mentre i due metodi *hasLaude* e *superato* sono veri rispettivamente nel solo caso del 30L, e per i voti positivi, ossia diversi da *ritirato* o *respinto*. Il factory method statico *of* restituisce la giusta istanza dell'enumerativo in base alla stringa ricevuta, che può essere o il valore numerico fra "18" e "30", oppure una delle stringhe "30L", "ID", "RT", "RE". Dualmente, un'apposita *toString* emette la stringa appropriata per ogni valore dell'enumerativo.

c) La classe ***Esame*** (fornito) rappresenta un esame, che associa a un'attività formativa un voto ottenuto in una ben precisa data. Il costruttore riceve tali dati, che sono poi recuperabili tramite gli appositi *accessor*; idonee *toString*, *equals*, *hashcode* completano l'implementazione. Da segnalare il metodo di utilità *toStringSenzaVoto*, che emette una stringa analoga a quella di *toString* ma priva del voto finale (utile per generare facilmente, in altra situazione, la carriera attualizzata, in cui il voto dovrà essere diverso)

d) la classe ***Carriera*** (fornita) mantiene al proprio interno la lista degli esami sostenuti, ordinata per data crescente e in subordine per codice univoco dell'attività formativa. Il costruttore crea una carriera inizialmente vuota, che verrà poi riempita via via tramite il metodo *inserisci*. Più precisamente:

- il metodo *inserisci* inserisce un esame in carriera, a condizione che un esame per la stessa attività formativa non sia già stato sostenuto in precedenza con esito positivo (altrimenti, deve lanciare ***IllegalArgumentException*** con opportuno messaggio d'errore); deve inoltre, ovviamente, verificare che l'argomento ricevuto non sia nullo, lanciando in tal caso ***IllegalArgumentException*** con

apposito messaggio d'errore. Nel solo caso in cui l'esame da inserire sia la prova finale (aspetto verificabile unicamente dalla descrizione, che in tal caso conterrà la dizione "PROVA FINALE"), deve altresì *verificare che la data della stessa sia posteriore a quella di ogni altro esame in carriera* – anche in questo caso, lanciando ***IllegalArgumentException*** con adeguato messaggio

- il metodo *getListaEsami* restituisce semplicemente la lista ordinata degli esami attualmente presenti in carriera
- il metodo *getTentativiPrecedenti* restituisce la lista ordinata dei soli esami (con esito ovviamente negativo) sostenuti in precedenza per la stessa attività formativa dell'esame ricevuto come argomento (se non ce ne sono, restituisce una lista vuota)
- il metodo *getUltimoEsameDato* restituisce, se esiste (per questo il tipo di ritorno è un ***Optional***), il più recente degli esami sostenuti
- il metodo *getUltimoTentativo* restituisce, se esiste (per questo il tipo di ritorno è un ***Optional***), il più recente degli esami (con esito negativo) sostenuti in precedenza per la stessa attività formativa dell'esame ricevuto come argomento
- il metodo *creditiAcquisiti* restituisce la somma dei crediti acquisiti da esami superati con esito positivo (sia con voto numerico, sia con giudizio di idoneità)
- un'apposita *toString* emette una stringa corrispondente alla lista di tutti gli esami (superati o meno), separati da "a capo", rispettando l'ordine di lista.

e) la classe ***Attualizzatore*** **(da completare)** attualizza una ***Carriera:*** il costruttore riceve tre argomenti, che costituiscono lo stato iniziale dell'attualizzatore (la carriera da attualizzare, la legge di decadimento desiderata e la data a cui effettuare l'attualizzazione) e le memorizza nello stato interno. Tale stato è però accessibile e modificabile in qualsiasi momento tramite le coppie di accessor *setCarriera* / *getCarriera*, *setFunction* / *getFunction*, *setDataDiAttualizzazione* / *getDataDiAttualizzazione*. Inoltre:

- il metodo *mediaPesata* **(da fare)** restituisce la media pesata dei voti (numerici) fin qui ottenuti, attualizzata alla data specificata per mezzo della funzione di decadimento specificata
  (NB: possono essere utili alcuni metodi privati forniti…)
- un'apposita *toString* emette una stringa corrispondente alla lista di tutti gli esami (superati o meno), separati da "a capo", rispettando l'ordine di lista

La classe definisce altresì, sotto forma di metodi statici, alcune funzioni di decadimento prestabilite, recuperabili tramite l'apposito metodo *elencoFunzioni,* che restituisce una mappa che associa un nome convenzionale (stringa) a ogni funzione (una *BiFunction<Esame,LocalDate,Double>*). **Al momento ne sono previste tre, di cui due già fornite e una da realizzare.**

- la funzione statica *UN_VOTO_IN_MENO_AL_MESE* **(da fare)** attualizza l'esame dato alla data fornita come argomento, restituendo il risultato sotto forma di double, secondo la logica di sottrarre un voto per ogni mese di distanza dalla data dell'esame (ad esempio, un esame sostenuto il 22/3/2020 con voto 23 vedrà il voto calare a 22 dal 22/4/2020; e così via), garantendo comunque che nessun voto possa mai scendere sotto quota 18 e che i voti non numerici restino inalterati (NB: può essere utile ispirarsi alle altre due fornite… 😊)

Parte 2 – Persistenza (punti: 9)

Package: speedycollege.persistence **[TEMPO STIMATO: 40-60 minuti]**

Sono forniti un certo numero di file di testo, tutti strutturati secondo il medesimo formato (possono contenere righe vuote). L'applicazione principale provvede da sé a recuperarne l'elenco dalla directory corrente e richiamarne ciclicamente la lettura: pertanto il candidato deve occuparsi unicamente di implementare il classico reader per un singolo file.

Ogni riga è organizzata in una serie di elementi separati da una o più tabulazioni. Due elementi ci sono sempre:

- codice identificativo dell'attività formativa (un intero long)
- denominazione dell'attività formativa (che può contenere spazi e ogni altro carattere diverso da tabulazione), seguita – dopo almeno uno spazio – dal numero di crediti nella forma "(cfu:*N*)", essendo *N* un numero intero positivo (NB: la sigla "cfu:" è sempre minuscola)

Solo se l'esame è stato sostenuto, sono presenti due ulteriori elementi:

- data in cui l'esame è stato sostenuto, nel formato italiano *short* GG/MM/AA
- voto (un numero fra 18 e 30, oppure una delle sigle 30L, ID, RT, RE rispettivamente per trenta e lode, idoneo, ritirato, respinto)

ESEMPIO:

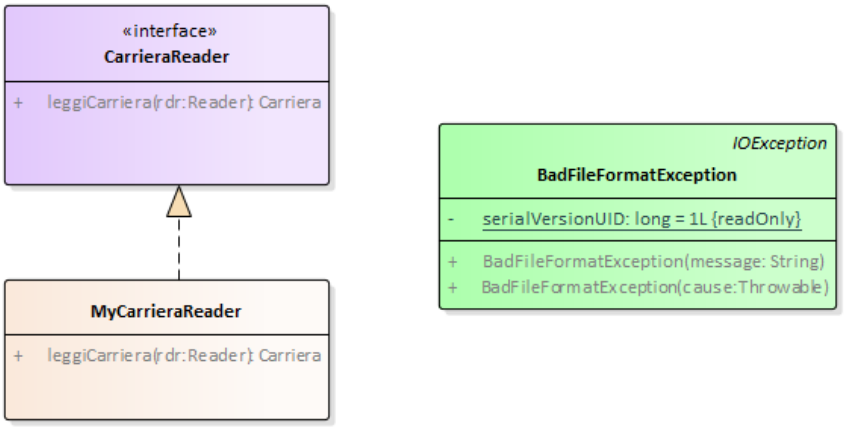
```
27991    ANALISI MATEMATICA T-1 (cfu:9)            12/01/20          RT
27991    ANALISI MATEMATICA T-1 (cfu:9)            10/02/20          22
28004    FONDAMENTI DI INFORMATICA T-1 (cfu:12)    13/02/20          24
29228    GEOMETRIA E ALGEBRA T (cfu:6)             18/01/20          26
26337    LINGUA INGLESE B-2 (cfu:6)                18/06/20          ID
27993    ANALISI MATEMATICA T-2 (cfu:6)            10/06/20          RE
27993    ANALISI MATEMATICA T-2 (cfu:6)            02/07/20          RT
27993    ANALISI MATEMATICA T-2 (cfu:6)            22/07/20          23
28006    FONDAMENTI DI INFORMATICA T-2 (cfu:12)    21/07/20          27
28011    RETI LOGICHE T (cfu:6)

28012    CALCOLATORI ELETTRONICI T (cfu:6)         22/01/21          RT
28012    CALCOLATORI ELETTRONICI T (cfu:6)         22/02/21          20
30780    FISICA GENERALE T (cfu:9)                 12/02/21          25
28032    MATEMATICA APPLICATA T (cfu:6)            02/02/21          24
28027    SISTEMI INFORMATIVI T (cfu:9)             03/06/21          28
28030    ECONOMIA E ORG. AZIENDALE T (cfu:6)       02/07/21          RE
28030    ECONOMIA E ORG. AZIENDALE T (cfu:6)       22/07/21          24
28029    ELETTROTECNICA T (cfu:6)                  02/09/21          26
28014    FOND. DI TELECOMUNICAZIONI T (cfu:9)      15/09/21          30
28020    SISTEMI OPERATIVI T (cfu:9)               12/01/22          24

28015    CONTROLLI AUTOMATICI T (cfu:9)            13/01/21          25
28016    ELETTRONICA T (cfu:6)                     10/02/21          22
28024    RETI DI CALCOLATORI T (cfu:9)             05/02/21          23
28659    TECNOLOGIE WEB T (cfu:9)                  12/06/21          25
28021    INGEGNERIA DEL SOFTWARE T (cfu:9)         24/06/21          27
17268    PROVA FINALE (cfu:3)

28074    TIROCINIO T (cfu:6)                       27/09/21          ID

88324    AMMINISTRAZIONE DI SISTEMI T (cfu:6)      13/07/21          29
88325    LAB. SICUREZZA INFORMATICA T (cfu:6)      07/06/21          30L
```



SEMANTICA:

a) L'eccezione ***BadFileFormatException*** (fornita) esprime l'idea di file formattato in modo scorretto

b) L'interfaccia ***CarrieraReader*** (fornita) dichiara il metodo *leggiCarriera,* che legge da un *Reader* (ricevuto come argomento) i dati di una carriera, configurando e restituendo l'opportuno oggetto ***Carriera.***

IMPORTANTE: poiché la ***Carriera*** è composta di ***Esami***, le righe contenenti la sola descrizione di attività formative, ossia quelle senza data e voto, devono essere comunque verificate a livello di formato ma ignorate per quanto riguarda l'inserimento in carriera, in quanto non descrivono un esame sostenuto.

c) La classe ***MyCarrieraReader*** (**da realizzare**) implementa ***CarrieraReader***: non prevede costruttori, si limita a implementare il metodo *leggiCarriera* come sopra specificato. In caso di problemi di I/O deve essere propagata l'opportuna ***IOException,*** mentre in caso di ***Reader*** nullo o altri problemi di formato dei file deve essere lanciata una opportuna ***BadFileFormatException***, il cui messaggio dettagli l'accaduto.
In particolare, il reader deve verificare, lanciando ***BadFileFormatException*** in caso contrario, che:

i. la riga contenga due o quattro elementi
ii. il primo elemento sia un intero long
iii. i crediti siano presenti nel formato sopra specificato, ovvero "(cfu:*N*)"

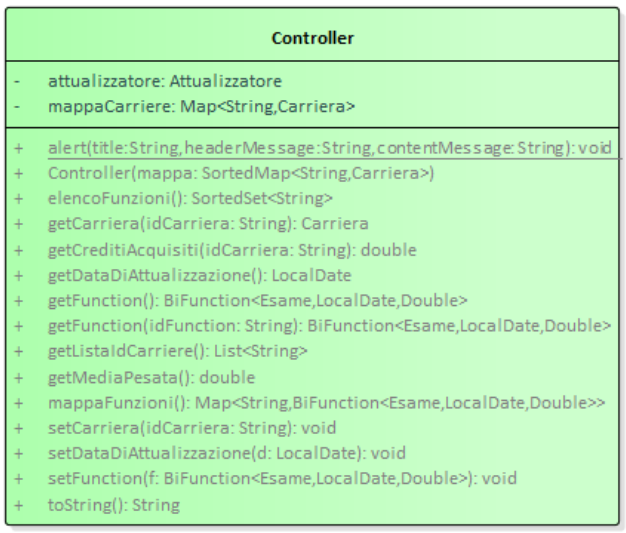
nonché, dove siano presenti anche gli altri due elementi:

iv. la data sia correttente formattata secondo la struttura GG/MM/AA
v. il voto sia un numero intero compreso fra 18 e 30, oppure una delle sigle sopra specificate

Parte 3 (punti: 7)

Package: speedycollege.controller *(punti 0)*

Il **Controller** (fornito) è organizzato secondo il diagramma UML nella figura seguente: esso mantiene internamente una mappa <stringa, carriera> che associa a ogni ***Carriera*** una opportuna stringa identificativa univoca.



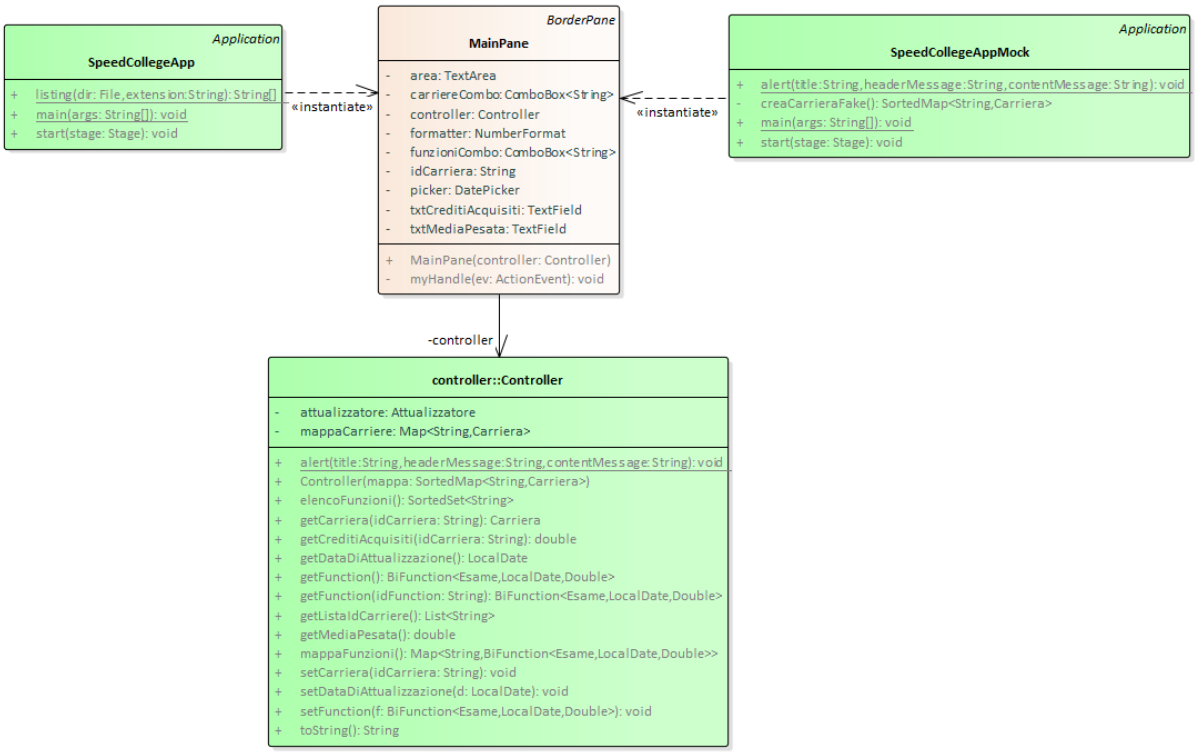
SEMANTICA:

- il costruttore riceve la mappa <stringa,carriera> su cui opererà
- un'ampia serie di metodi fa da ponte con i quasi omonimi metodi dell'**Attualizzatore**
- la classe contiene anche il **metodo statico ausiliario *alert***, utile per mostrare avvisi all'utente.

Package: speedycollege.ui **[TEMPO STIMATO: 35-50 minuti]** *(punti 8)*

La classe ***SpeedyCollegeApp*** (fornita) costituisce l'applicazione JavaFX che si occupa di aprire i file, creare il controller e incorporare il ***MainPane***. Per consentire di collaudare la GUI anche in assenza / in caso di malfunzionamento della parte di persistenza, è possibile avviare l'applicazione mediante la classe ***SpeedyCollegeAppMock***.

L'interfaccia utente è illustrata nelle figure seguenti e segue il modello sotto illustrato*:*



L'interfaccia grafica si presenta come segue:

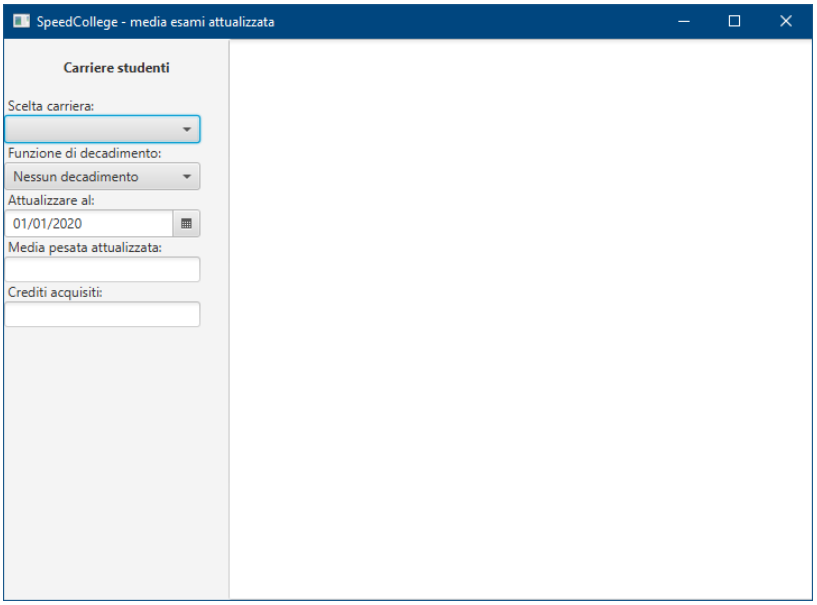
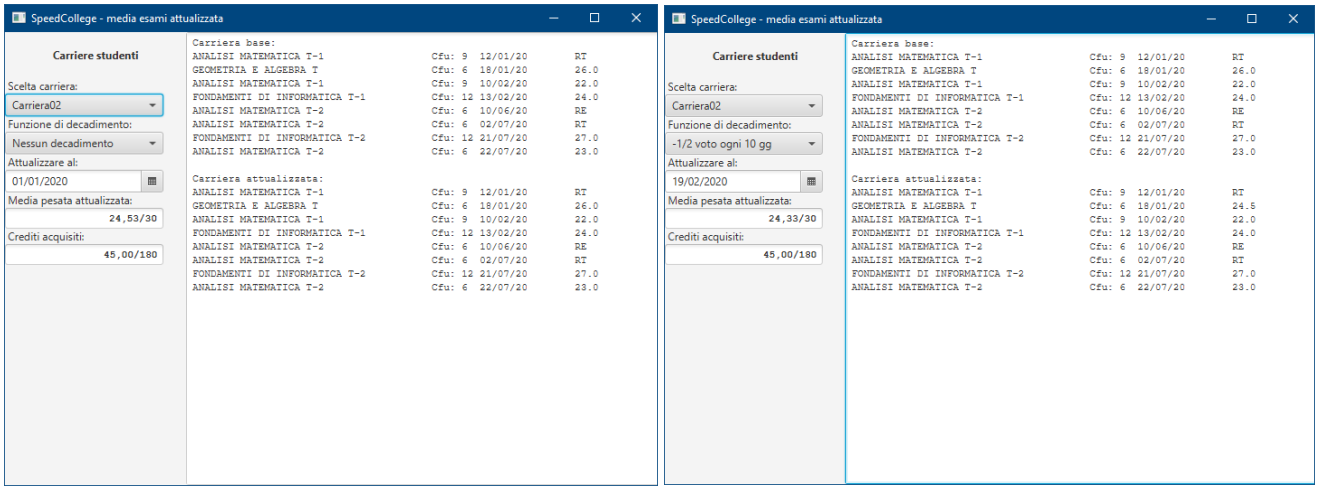


Fig. 1: la situazione iniziale della GUI, con combo vuota e nessuna selezione ancora effettuata

- a sinistra, due combo elencano una le carriere disponibili, l'altra le funzioni di decadimento predefinite; sotto, un caledarietto (***DatePicker***) consente di stabilire la data a cui attualizzare i voti (default: 1° gennaio 2020). Subito a seguire, due campi di testo (inizialmente vuoti e non scrivibili dall'utente) mostrano rispettivamente le media pesata attualizzata (che quindi può cambiare al variare della data scelta) e i crediti acquisiti relativi alla carriera selezionata (che invece restano, ovviamente, fissi al variare della data).
- a destra, un'area di testo (inizialmente vuota e non scrivibile dall'utente) mostra i dettagli della carriera selezionata, ossia la lista degli esami in essa contenuti.

L'utente può interagire selezionando o la carriera nella combo, o la funzione di decadimento, o la data di attualizzazione: in tutti i casi viene scatenato lo stesso gestore di eventi, che provvede a popolare i campi di testo sottostanti e l'area sulla destra con i dati corrispondenti.



Figg. 2 / 3: la GUI dopo la selezione di una carriera, prima senza decadimento (a sinistra), poi con una legge di decadimento (a destra), in quel caso riferia alla data del 19/2/2020.

Il MainPane è fornito *parzialmente realizzato*: è presente buona parte dell'impostazione strutturale, mentre sono da completare la configurazione di alcuni componenti e la gestione degli eventi.

La classe ***MainPane*** (**da completare**) estende ***BorderPane*** e prevede:

1) a sinistra, una ***VBox*** con le due combo, il picker e i due campi di testo, oltre alle opportune etichette
2) a destra, una ***VBox*** con la sola area di output.

La **parte da completare** riguarda:

1) la configurazione iniziale dei formattatori numerici
2) il popolamento delle due ***combo***
3) la predisposizione, con opportuna data di default al 1° gennaio 2020, del ***DatePicker***
4) l'aggancio dell'opportuno listener incapsulato nel metodo ausiliario *myHandler*
5) la logica di gestione dell'evento, incapsulata nel metodo privato *myHandler*

In particolare, la gestione dell'evento deve:

- recuperare la carriera, la funzione di decadimento e la data di attualizzazione selezionate
- utilizzare tali dati per popolare i vari campi di uscita, tramite gli appositi metodi del controller

Cose da ricordare

- salva costantemente il tuo lavoro: l'informatica a volte può essere "subdolamente ostile"..
- in particolare: se ora compila e stai per fare modifiche, salva la versione attuale (non si sa mai)

Checklist di consegna

- Hai fatto un **JAR eseguibile**, che contenga cioè l'indicazione del main?
- Hai controllato che **si compili** e **ci sia tutto**? [NB: non includere il PDF del testo]
- Hai **rinominato** IL PROGETTO, lo ZIP e il JAR esattamente come richiesto?
- Hai **chiamato** la cartella del progetto esattamente come richiesto?
- Hai fatto un **unico file ZIP (NON .7z, rar o altri formati)** contenente l'intero progetto? In particolare, ti sei assicurato di aver incluso tutti i file .java (e non solo i .class)?
- **Hai consegnato DUE file distinti, ossia lo ZIP col progetto e il JAR eseguibile?**
- Su EOL, hai **premuto** il tasto "CONFERMA" per inviare il tuo elaborato?