ESAME DI FONDAMENTI DI INFORMATICA T-2 del 13/06/2017
Proff. Enrico Denti – Gabriele 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 e CARTELLA : CognomeNome-matricola (es. RossiMario-0000123456)
NOME ZIP DA CONSEGNARE : CognomeNome-matricola.zip (es. RossiMario-0000123456.zip)

La nota azienda "ED Fitness & Health" desidera offrire ai suoi clienti l'applicazione "MyFitnessDiary" che permetta di:

- inserire giornalmente l'insieme dei propri ***allenamenti***;
- produrre a video un ***report giornaliero*** degli allenamenti contenente l'elenco delle varie attività sportive con indicazione dei minuti e delle calorie bruciate, nonché il totale giornaliero dei minuti di allenamento e delle calorie bruciate in quel giorno;
- produrre su file un ***report settimanale*** (ReportSettimanale.txt, nel formato dettagliato più oltre) che mostri i totali della settimana e i valori medi giornalieri per minuti di allenamento e calorie bruciate.

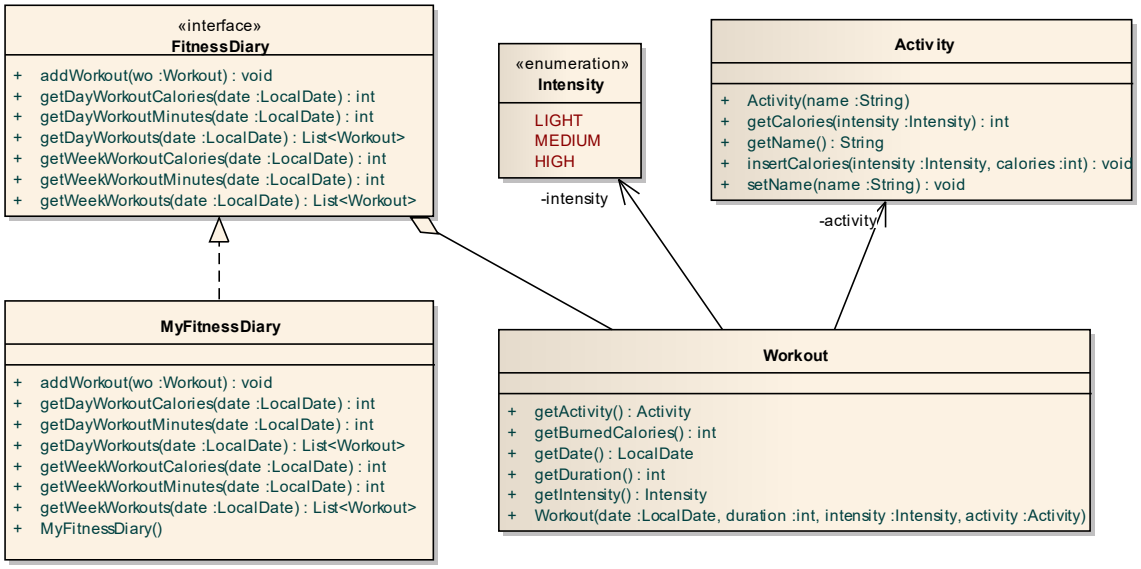
DESCRIZIONE DEL DOMINIO DEL PROBLEMA
L'utente di "MyFitnessDiary" deve poter inserire nell'applicazione l'insieme degli *allenamenti* che svolge giornalmente. Con ***allenamento*** si intende ogni tipo di ***attività sportiva*** (es: corsa, jogging, running, GAG, aerobica, fitness, cardio fitness, pesi, pilates, yoga, judo, karate, body building, etc.) con associata la relativa ***durata*** dell'attività e l'***intensità*** (es: leggera, media, elevata), che determina le calorie bruciate.

Il file di testo Attivita.txt contiene l'elenco di tutte le possibili attività sportive, con relativa indicazione delle calorie bruciate per ogni minuto di allenamento per i vari gradi di intensità.

Parte 1 (punti: 17)

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

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



SEMANTICA:

a) L'enumerativo ***Intensity*** (fornito nello Start kit) rappresenta i diversi tipi di intensità di un allenamento.

b) La classe ***Activity*** (fornita nello Start kit) rappresenta la generica attività sportiva: il metodo ***getName*** restituisce il nome dell'attività, mentre il metodo ***getCalories*** restituisce le calorie bruciate in ogni minuto di allenamento in base all'***Intensity*** ricevuta come argomento.

c) La classe ***Workout*** (fornita nello Start kit) rappresenta un allenamento rappresentato in termini di *data*, *durata*, *intensità* e *attività*. La classe deve prevedere opportuni metodi accessor (ma non di modifica: trattasi di oggetto non modificabile) nonché il metodo ***getBurnedCalories*** che calcoli e restituisca le calorie bruciate durante l'allenamento. Il costruttore deve controllare gli argomenti in ingresso, lanciando ***IllegalArgumentException*** in caso di errori.

d) L'interfaccia ***FitnessDiary*** (fornita nello Start kit) dichiara i metodi messi a disposizione dal diario.

e) La classe ***MyFitnessDiary*** (da realizzare) implementa l'interfaccia ***FitnessDiary***:

- ***addWorkout(Workout)*** aggiunge un workout al diario, restituendo un boolean indicante l'esito dell'operazione (successo=true, fallimento=false);
- ***getDayWorkouts(LocalDate date)*** restituisce la lista dei ***Workout*** relativi al giorno specificato;
- ***getWeekWorkouts(LocalDate date)*** restituisce la lista dei ***Workout*** relativi alla settimana (lunedì-domenica) che contiene la data specificata: ad esempio, se la data specificata è il 13/06/2017, il metodo deve restituire tutti i ***workout*** svolti nelle date da 12/06/17 al 18/06/2017, estremi inclusi;
- ***getWeekWorkoutCalories(LocalDate date)*** restituisce un intero che rappresenta le **calorie totali** dei workout della settimana contenente la data specificata;
- ***getDayWorkoutCalories(LocalDate date)*** restituisce un intero che rappresenta le **calorie totali** dei workout del giorno specificato;
- ***getWeekWorkoutMinutes(LocalDate date)*** restituisce un intero che rappresenta i **minuti totali** dei workout della settimana contenente la data specificata;
- ***getDayWorkoutMinutes(LocalDate date)*** restituisce un intero che rappresenta i **minuti totali** dei workout del giorno specificato;

Persistenza (myfitnessdiary.persistence) *(punti 9)*

Come già anticipato, il file di testo Attivita.txt contiene l'elenco di tutte le possibili attività sportive, con l'indicazione delle calorie bruciate per ogni minuto di allenamento per i vari gradi di intensità.

Più precisamente, Attivita.txt contiene una riga per ogni attività sportiva, formattata come segue:

```
Nome attività -- calL   calM   calH
```

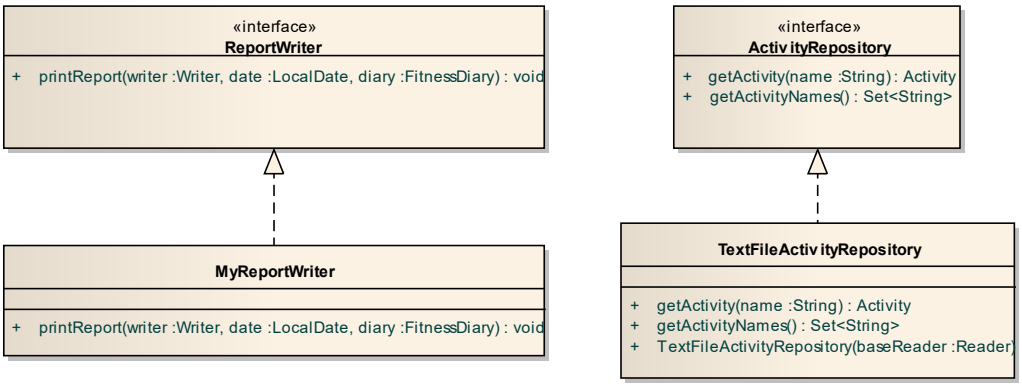
dove:

- *Nome attività* rappresenta il nome dell'attività sportiva (es: yoga, jogging, running, cardio fitness, yoga, pilates, body building, mixed martial arts, etc.): come tale, può contenere spazi;
- uno o più caratteri '-' separano il nome dell'attività dalle calorie bruciate
- `calL`, `calM`, `calH` sono tre numeri interi che rappresentano le calorie bruciate per ogni minuto di allenamento rispettivamente per intensità leggera (`calL`), media (`calM`) o elevata (`calH`): i tre valori sono separati tra loro da una o più tabulazioni o spazi.

Il report settimanale dovrà essere memorizzato nel file di testo ReportSettimanale.txt in forma analoga alla seguente:

```
Totali settimana del 31 maggio 2017
minuti totali allenamento 100, calorie totali bruciate 300
minuti medi di allenamento 14, calorie medie bruciate 42
```

Più esattamente, la prima riga specifica il giorno selezionato (stile LONG, locale ITALY) mentre le successive riportano rispettivamente, i minuti e le calorie totali (seconda riga) e medi (terza riga) per la settimana da lunedì a domenica che contiene il giorno selezionato.



SEMANTICA:

a) L'interfaccia ***ActivityRepository*** (fornita nello Start kit) dichiara i metodi messi a disposizione dal repository delle attività sportive.

b) La classe ***TextFileActivityRepository*** (**da realizzare**) implementa ***ActivityRepository*** e riceve in ingresso nel costruttore il Reader da cui leggere i dati relativi alle ***Activity***. Tale classe dovrà quindi recuperare tutte le attività sportive e memorizzarle in una opportuna struttura dati che permetta una ricerca **veloce** delle ***Activity*** per **nome**. In caso di problemi nella lettura del file, deve essere lanciata una ***BadFileFormatException***.

f) La classe deve inoltre mettere a disposizione i seguenti metodi:

- ***getActivity(String name)*** che restituisce una ***Activity*** ricevendo in ingresso il nome dell'attività;
- ***getActivityNames*** che restituisce un ***Set<String>*** contenente i nomi di tutte le attività sportive.

c) L'interfaccia ***ReportWriter*** (fornita nello Start kit) dichiara il metodo per la scrittura del report.

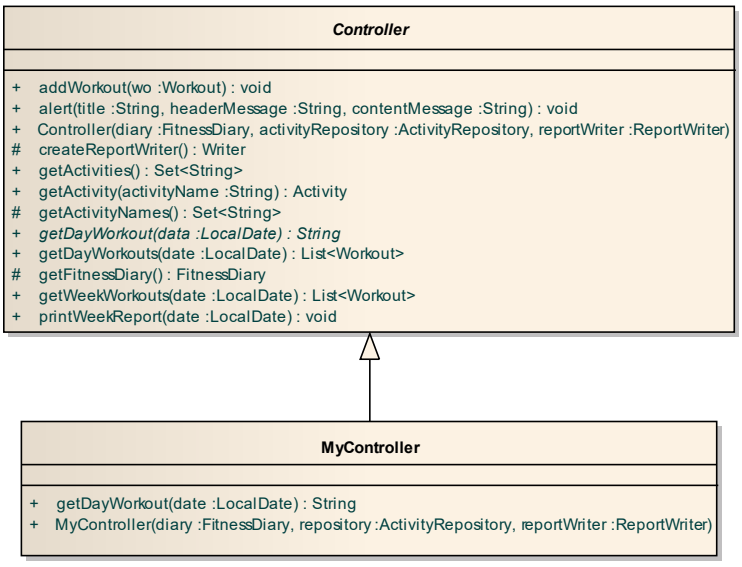
d) La classe ***MyReportWriter*** (**da realizzare**) implementa ***ReportWriter***. La classe offre solo il seguente metodo:

- ***printReport(Writer writer, LocalDate date, FitnessDiary diary)*** genera il report secondo il formato sopra specificato: riceve in ingresso un ***Writer*** su cui scrivere, una ***LocalDate*** che rappresenta uno dei giorni all'interno della settimana di cui si vuole stampare il report e il ***FitnessDiary*** da usare per ottenere i valori da scrivere.

Parte 2 (punti: 13)

Controller (myfitnessdiary.controller) *(punti 3)*

Il Controller deve essere organizzato secondo il diagramma UML più sotto riportato.



SEMANTICA:

La classe astratta ***Controller*** (fornita) dichiara l'interfaccia del controller e ne implementa lo scheletro. **Rinviando al codice incluso nello Start kit per i dettagli dei molti metodi forniti**, si evidenzia che:

- il costruttore ha tre argomenti, ***FitnessDiary***, ***ActivitiesRepository*** e ***ReportWriter***
- il metodo ***addWorkout*** riceve in ingresso un ***Workout*** e lo inserisce nel ***FitnessDiary***;
- il metodo **astratto** ***getDayWorkout(LocalDate data)*** restituisce una stringa che elenca gli allenamenti (nome attività e relativa indicazione di minuti di allenamento e calorie bruciate) relativi alla data specificata, nonché i minuti totali di allenamento e le calorie totali bruciate.

La classe ***MyController*** (**da realizzare**) completa l'implementazione realizzando ***getDayWorkout*** formattando la stringa come segue:

```
Allenamento di mercoledì 31 maggio 2017
Body building minuti: 10  calorie bruciate: 40
Calcio minuti: 60  calorie bruciate: 600
Nuoto rana minuti: 5  calorie bruciate: 35

Minuti totali allenamento: 75
Calorie totali bruciate: 675
```

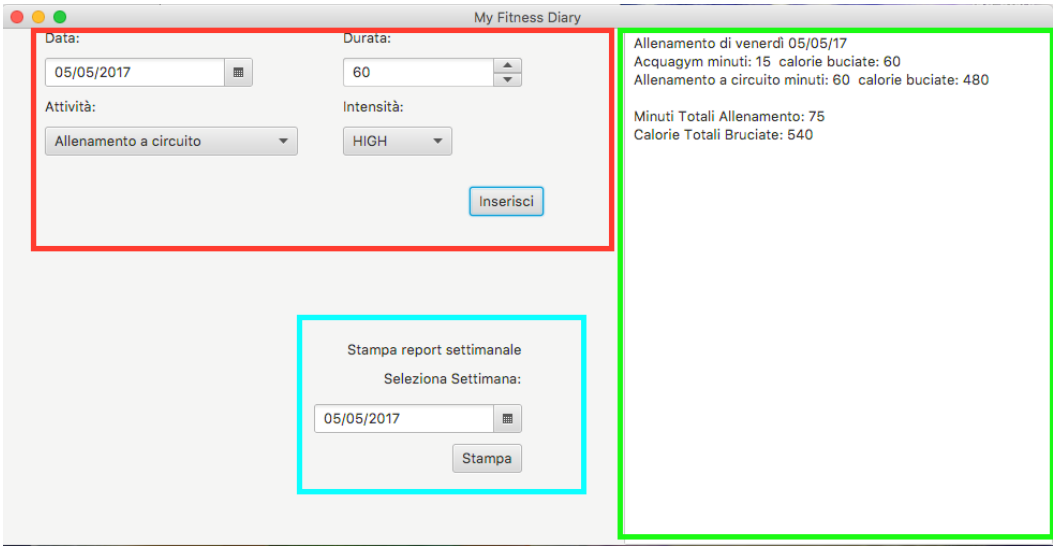
Interfaccia Utente (myfitnessdiary.ui) (punti 10)

Studenti A.A. 2016/17 – versione con JavaFX

Importante: nel caso di svolgimento della versione JavaFX, eliminare dallo start kit le classi MyMain e GUITest.

La classe ***MyFitnessDiaryApplication*** (fornita) costituisce l'applicazione JavaFX che si occupa di aprire il file, il controller e incorporare il ***MyFitnessPane*** (**da realizzare**). Per consentire di collaudare la GUI anche in assenza della parte di persistenza, è possibile avviare l'applicazione mediante la classe ***MockFitnessDiaryApplication***.

L'interfaccia utente deve essere simile (non necessariamente identica) all'esempio mostrato nella figura seguente:



La classe concreta ***MyFitnessPane*** (**da realizzare**) deve estendere ***BorderPane***.

La GUI è organizzata in tre sezioni:

1) Sezione inserimento allenamento (parte in alto a sinistra, evidenziata in rosso):
   - un ***DatePicker*** che indicata la data (valore iniziale: data odierna)
   - uno ***Spinner*** per la durata dell'allenamento (minimo 0, massimo 1000, valore iniziale 0, passo 5)

- una ***ComboBox*** pre-popolata con l'elenco di tutte le attività sportive
- una ***ComboBox*** pre-popolata con i diversi valori di intensità
- un bottone di inserimento che, quando premuto, permette di inserire un nuovo allenamento e di conseguenza aggiorna la ***TextArea*** indicata nel rettangolo verde (dettagli sotto).

Se uno qualunque dei dati selezionati è errato (ossia tale da non consentire la creazione del ***Workout:*** data errata, durate o intensità negative, attività inesistenti) occorre avvisare l'utente facendo apparire una finestra di dialogo di errore tramite il metodo ***alert*** del ***Controller***. Ovviamente in tal caso non si effettua alcun inserimento.

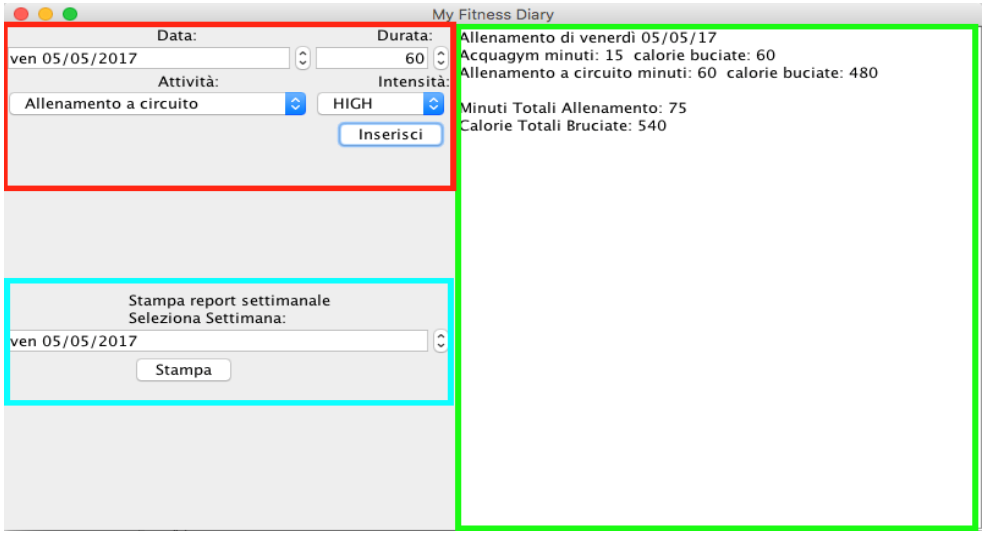
2) Sezione stampa report (parte in basso a sinistra, evidenziata in azzurro):
   - un ***DatePicker*** che indicata la data (valore iniziale: data odierna)
   - un bottone che, quando premuto, stampa il report settimanale (metodo ***printWeekReport*** del ***Controller***).
3) Sezione visualizza allenamenti nella giornata (parte a destra, evidenziata in verde)

   una ***TextArea*** che si aggiorna ad ogni inserimento di allenamento mostrando l'elenco degli allenamenti giornalieri con indicazione della durata e delle calorie bruciate, nonché il totale dei minuti di allenamento nella giornata e delle calorie bruciate (metodo ***getDayWorkout*** del ***Controller***) come mostrato in figura.

Studenti A.A. 2015/16 e precedenti – versione Swing

Importante: nel caso di svolgimento della versione Swing, eliminare dallo start kit le classi MyFitnessDiaryApplication e MockFitnessDiaryApplication.

L'interfaccia utente deve essere simile (non necessariamente identica) all'esempio mostrato nella figura seguente:



La classe ***MyMain*** (fornita) contiene il *main* di partenza dell'intera applicazione; tale *main* si occupa di aprire il file e creare tutto il necessario. Per consentire di collaudare la GUI anche in assenza della parte di persistenza, è possibile avviare l'applicazione mediante la classe ***GUITest***.

Il ***FitnessFrame*** (***da realizzare***) dovrà organizzare l'interfaccia in tre diverse sezioni:

1) Sezione inserimento allenamento (parte evidenziata in rosso):
   - un ***MyDateSpinner*** (fornito nello Start Kit) che indicata la data (valore iniziale: data odierna)
   - un ***JSpinner*** per la durata dell'allenamento; usare uno ***SpinnerNumberModel*** come segue:

```
SpinnerModel spinnerModel = new SpinnerNumberModel(0, 0, 1000, 5);
```

```
JSpinner durationSpinner = new JSpinner(spinnerModel);
…
int duration = (int) durationSpinner.getValue();      // estrazione del valore
```

- una ***JComboBox*** pre-popolata con l'elenco di tutte le attività sportive
- una ***JComboBox*** pre-popolata con i diversi valori di intensità
- un bottone di inserimento che quando premuto permette di inserire un nuovo allenamento e di conseguenza aggiorna la JTextArea indicata nel rettangolo verde

Se uno qualunque dei dati selezionati è errato (ossia tale da non consentire la creazione del ***Workout:*** data errata, durate o intensità negative, attività inesistenti) occorre avvisare l'utente facendo apparire una finestra di dialogo di errore: a tal fine basta invocare ***JOptionPane.showMessageDialog***(*null*, messaggio). Ovviamente in tal caso non si effettua alcun inserimento.

2) Sezione stampa report (parte evidenziata in azzurro):
   - un ***MyDateSpinner*** (fornito nello Start Kit) che indicata la data (valore iniziale: data odierna)
   - un bottone di stampa che quando premuto stampa il report settimanale (metodo ***printWeekReport*** del ***Controller***).
3) Sezione visualizza allenamenti nella giornata (parte evidenziata in verde)
   - una ***JTextArea*** che si aggiorna ad ogni inserimento di allenamento: mostra l'elenco degli allenamenti giornalieri con indicazione della durata e delle calorie bruciate, nonché il totale dei minuti di allenamento nella giornata e delle calorie bruciate (metodo ***getDayWorkout*** del ***Controller***) come mostrato in figura.