LAB11b – EDFitness

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

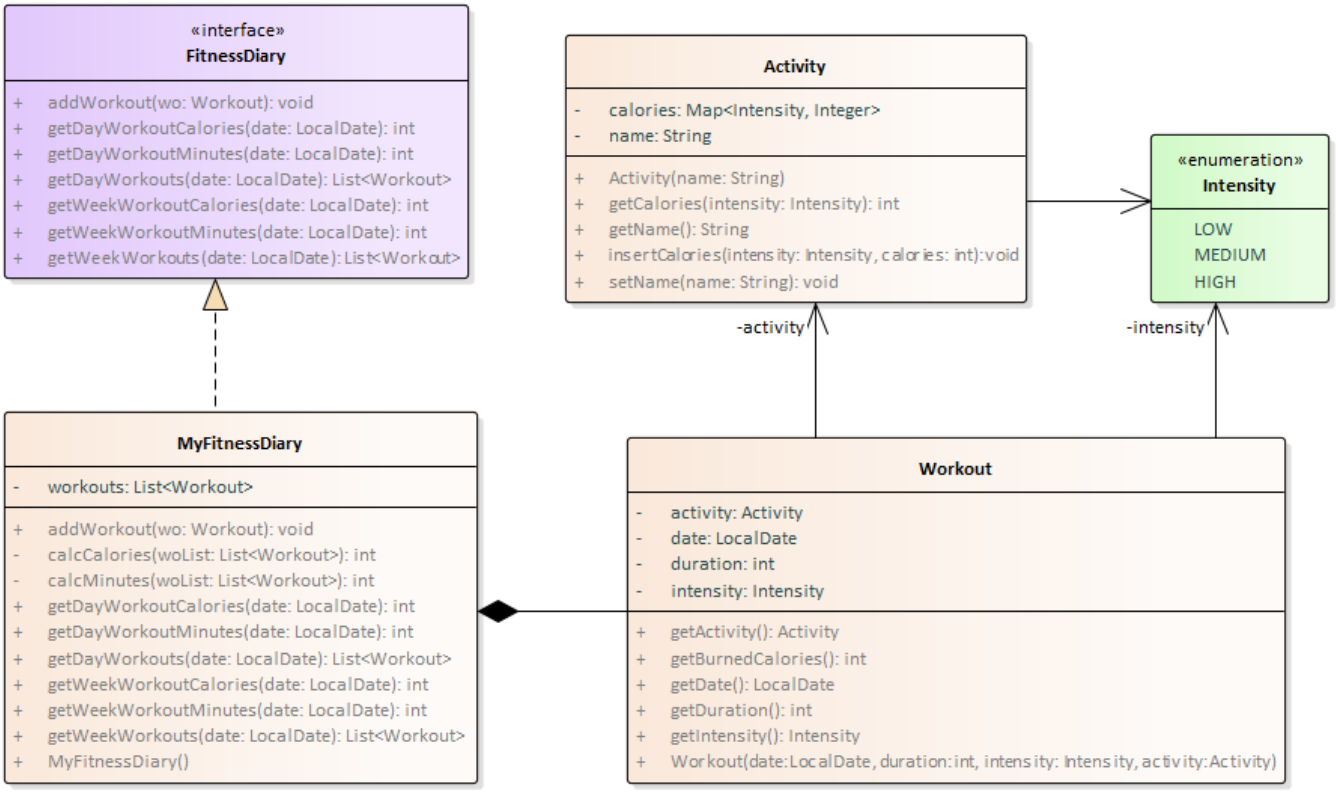
L'utente di "MyFitnessDiary" deve poter inserire nell'applicazione l'insieme degli *allenamenti* che svolge giornalmente. Con ***allenamento*** si intende ogni tipo di ***attività sportiva*** (es: corsa, jogging, running, GAG, aerobica, fitness, cardio fitness, pesi, pilates, yoga, judo, karate, body 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*** (da realizzare) 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) vedi UML per il dettaglio;
- il metodo ***getBurnedCalories*** che calcola e restituisce 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;
- ***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 05/05/2020, il metodo deve restituire tutti i ***workout*** svolti nelle date da 04/05/2020 al 10/05/2020, 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)*

In questa esercitazione la persistenza è fornita già svolta nello Start kit

Parte 2 (punti: 13)

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

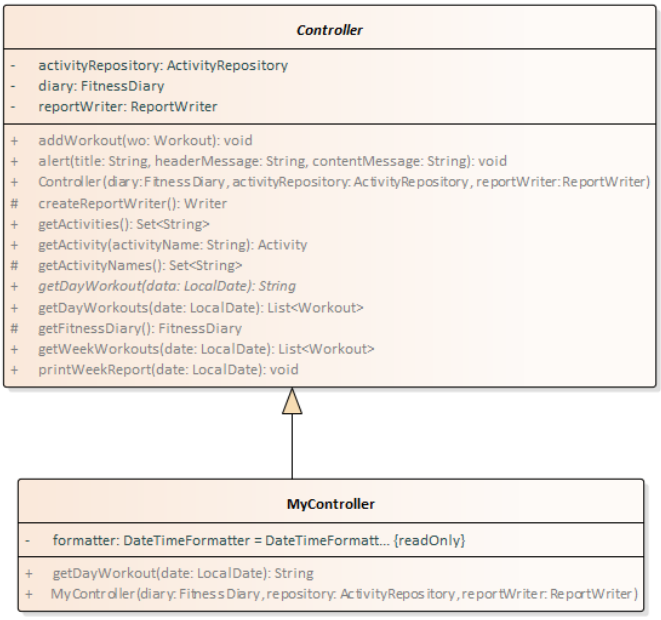
Il Controller deve essere organizzato secondo il diagramma UML più sotto riportato.

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 05 maggio 2020
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)*

In questa esercitazione la grafica è fornita già svolta nello Start kit