

Programmazione 2

23 Aprile 2021 –Primo Compitino

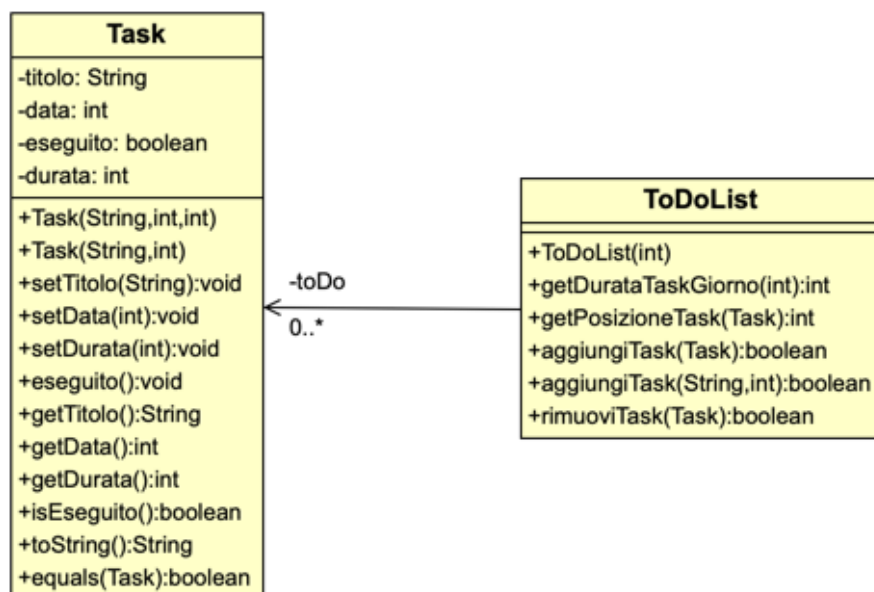
Testo parte di pratica

Si consideri un programma per gestire le liste delle cose da fare (todo list). Un'attività da svolgere (o task) è caratterizzata da alcune informazioni tra cui una data e una durata. La data indica quando il task deve essere e la durata indica quante ore durerà il task. Un task può essere inserito nella lista delle cose da fare solo se la data in cui deve essere svolto è quella odierna o una data nel futuro (non si possono inserire task nel passato) e se nella data indicata per il task la somma delle durate di tutti i task pianificati (incluso quello che si vorrebbe inserire) non supera le 8 ore (una giornata ha 8 ore 'lavorative').

Implementare le classi esattamente come rappresentate dal seguente diagramma UML. Il diagramma include tutti e i soli metodi richiesti, compresi quelli di incapsulamento.

Viene fornita la classe **DataUtil** che dispone del solo metodo statico che restituisce la data odierna nel formato AAAAMMGG: **getDataDiOggi() : int**. Ad esempio, se invocato oggi restituirà 20210423.

Viene anche fornita la classe **TestToDoList** che contiene un insieme di casi di test che devono essere fatti girare di volta in volta in modo da verificare la corretta realizzazione del programma.



Classe Task:

- ✓ Rappresenta una attività da svolgere. È caratterizzata da un **titolo**, da una **data** in cui l'attività deve essere svolta, una **durata** in ore e dal fatto che sia stata fatta o meno (**eseguito**). Tutti gli attributi sono riscrivibili e leggibili dall'esterno.
- ✓ Di seguito le regole per la modifica degli attributi da codificare nei metodi di **setter**:
 1. **setTitolo(String titolo)**: imposta un titolo al task se il titolo passato in ingresso è diverso da null o dalla stringa vuota, altrimenti imposta il titolo a "Da specificare"
 2. **setData(int data)**: la data è intesa come un **int** nella forma AAAAMMGG. Ad esempio, la data di oggi è espressa come 20210423. Il metodo imposta la data al task se e solo se la data passata in ingresso è almeno pari alla data odierna o nel futuro (controllo semplice su interi)
 3. **setDurata(int durata)**: imposta la durata se il valore passato in ingresso è un intero compreso tra 1 e 8, altrimenti la imposta per default ad 1
 4. **eseguito()**: imposta l'attributo **eseguito** a **true**
- ✓ Definisce un costruttore che inizializza gli attributi rispettando le restrizioni sopra descritte. Il costruttore accetta nell'ordine il **titolo**, la **data** e la **durata** ed imposta a **false** l'attributo **eseguito**.

- ✓ Definisce un ulteriore costruttore che inizializza tutti i parametri come passati in ingresso, ed imposta la `data` a quella odierna.
- ✓ Due `Task` sono uguali se hanno lo stesso `titolo` (a prescindere dalla capitalizzazione delle lettere), la stessa `data` e la stessa `durata`
- ✓ Il metodo `toString()` restituisce una stringa che specifica il titolo, la data, la durata e il fatto che sia stato svolto o meno

Classe `ToDoList`:

- ✓ Una `to do list` contiene `task` che devono essere svolti (associazione `todo`).
- ✓ Definisce un costruttore che inizializza il numero massimo di `task` che possono essere inseriti. Se viene passato un numero minore od uguale a zero, la lista viene inizializzata in modo che contenga al massimo 5 `task`
- ✓ Il metodo `getDurataTaskGiorno(int data)` restituisce la somma di tutte le durate dei `task` allocati nella giornata passata in ingresso. Non viene richiesto il controllo sulla data passata in ingresso: si assume sia corretta.
- ✓ Il metodo `getPosizioneTask(Task task)` restituisce la posizione all'interno dell'array del `task` uguale al `task` passato in ingresso. Se non esiste un `task` uguale a quello passato in ingresso il metodo restituisce -1.
- ✓ Il metodo `aggiungiTask(Task task)` aggiunge il `task` passato in ingresso se:
 1. l'inserimento del `task` non fa sforare il numero massimo di ore complessive allocabili in una giornata per svolgere i `task` (max 8) rispetto alla giornata specificata nel `task` in ingresso
 2. il `task` non è già presente
 3. la lista non è piena
 Il metodo restituisce `true` se il `task` è stato inserito, `false` in caso contrario.
- ✓ Il metodo `aggiungiTask(String titolo, int durata)` aggiunge un nuovo `task` da svolgere nella giornata odierna se le condizioni specificate al punto sopra sono soddisfatte.
- ✓ Il metodo `rimuoviTask(Task task)` rimuove dalla lista il `task` uguale a quello passato in ingresso, se presente. Il metodo restituisce `true` se l'elemento è stato trovato e rimosso, `false` in caso contrario.