ESAME DI FONDAMENTI DI INFORMATICA T-2 del 9/2/2025
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 che *non passino almeno 2/3 dei test* o siano *palesemente lontani da 18/30* NON SARANNO CORRETTI e causeranno la verbalizzazione del giudizio "RESPINTO".

DOMINIO DEL PROBLEMA: Il personale che lavora su progetti finanziati è tenuto a compilare i cosiddetti *timesheet,* ossia dei *rendiconti mensili* nei quali vengono riportate, per ogni giorno del mese, le ore lavorate su ogni progetto; i rendiconti mensili sono poi raggruppati in un unico *rendiconto annuale*. Un esempio di rendiconto mensile cartaceo è illustrato sotto: come si può vedere, per ogni giorno del mese sono riportate le ore lavorate (nel formato HH:MM) su ogni progetto: opportune righe/colonne di sintesi forniscono i subtotali giornalieri (in verticale) e per progetto (in orizzontale), nonché il totale generale delle ore lavorate nel mese (in basso a destra).

Timesheet del mese di MARZO 2024										
	1	2	3		27	28	29	30	31	**tot**
Progetto A	01:30	00:00	00:00		01:00	02:00	00:00	00:00	00:00	**15:30**
Progetto B	01:00	00:00	00:00		01:00	00:00	03:30	00:00	00:00	**24:00**
Progetto C	00:00	00:00	00:00		01:00	01:30	01:30	00:00	00:00	**18:30**
…	…	…	…		…	…	…	…	…	…
TOTALE	***02:30***	***00:00***	***00:00***		***03:00***	***03:00***	***03:00***	***00:00***	***00:00***	**88:00**

Naturalmente, i giorni riportati sono tutti e soli quelli previsti dallo specifico mese, ossia 30 per i mesi di 30 giorni, 29 o 28 per febbraio (a seconda se l'anno sia bisestile o meno), 31 per gli altri mesi.

Poiché in molte realtà non è consentito al personale di lavorare di sabato e/o di domenica (come nella figura sopra), le ore riportate per quelle giornate, in tal caso, dovranno necessariamente essere zero. Spesso, inoltre, i contratti di lavoro stabiliscono un *numero massimo* di ore giornaliere lavorabili, oltre le quali non si può andare (tipicamente 8): naturalmente, nessuno può indicare in un dato giorno più di tale numero di ore.

OBIETTIVO: È richiesto di sviluppare un'applicazione che sostituisca i rendiconti cartacei, consentendo di inserire le ore lavorate tramite un'opportuna interfaccia grafica. L'app dovrà effettuare automaticamente i controlli sull'eventuale lavoro di sabato e festivo (ossia, dovrà consentirlo solo se le impostazioni dell'applicazione lo permettono) e sulle ore giornaliere lavorate (che non potranno eccedere il massimo prestabilito), nonché fornire la sintesi delle ore totali lavorate (per mese, per progetto, annuali) o il loro dettaglio, come da figure sotto riportate.

Il file di testo Projects.txt specifica, nel formato descritto più oltre, i progetti attivi con il numero totale di ore previste per ciascuno. L'applicazione dovrà:

a) leggere tali file e caricare i dati nelle opportune strutture-dati interne
b) permettere all'utente, tramite la GUI, di introdurre via via i dati necessari e avere sempre sott'occhio la situazione, prevenendo eventuali azioni errate (es. inserimento di ore lavorative di domenica, orari errati, etc.) tramite appositi dialoghi.

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

PARTE 1 – Modello dei dati: Punti 11 [TEMPO STIMATO: 50-70 minuti]

PARTE 2 – Persistenza: Punti 8 [TEMPO STIMATO: 35-40 minuti]

PARTE 3 – Grafica: Punti 11 [TEMPO STIMATO: 50-70 minuti]

NUMERO MINIMO DI TEST CON SUCCESSO PERCHÉ IL COMPITO SIA CORRETTO

40 su 58

JAVAFX – Parametri run configuration nei LAB

```
--module-path "C:\applicativi\moduli\javafx-sdk-21.0.2\lib"
--add-modules javafx.controls
```

Cose da ricordare

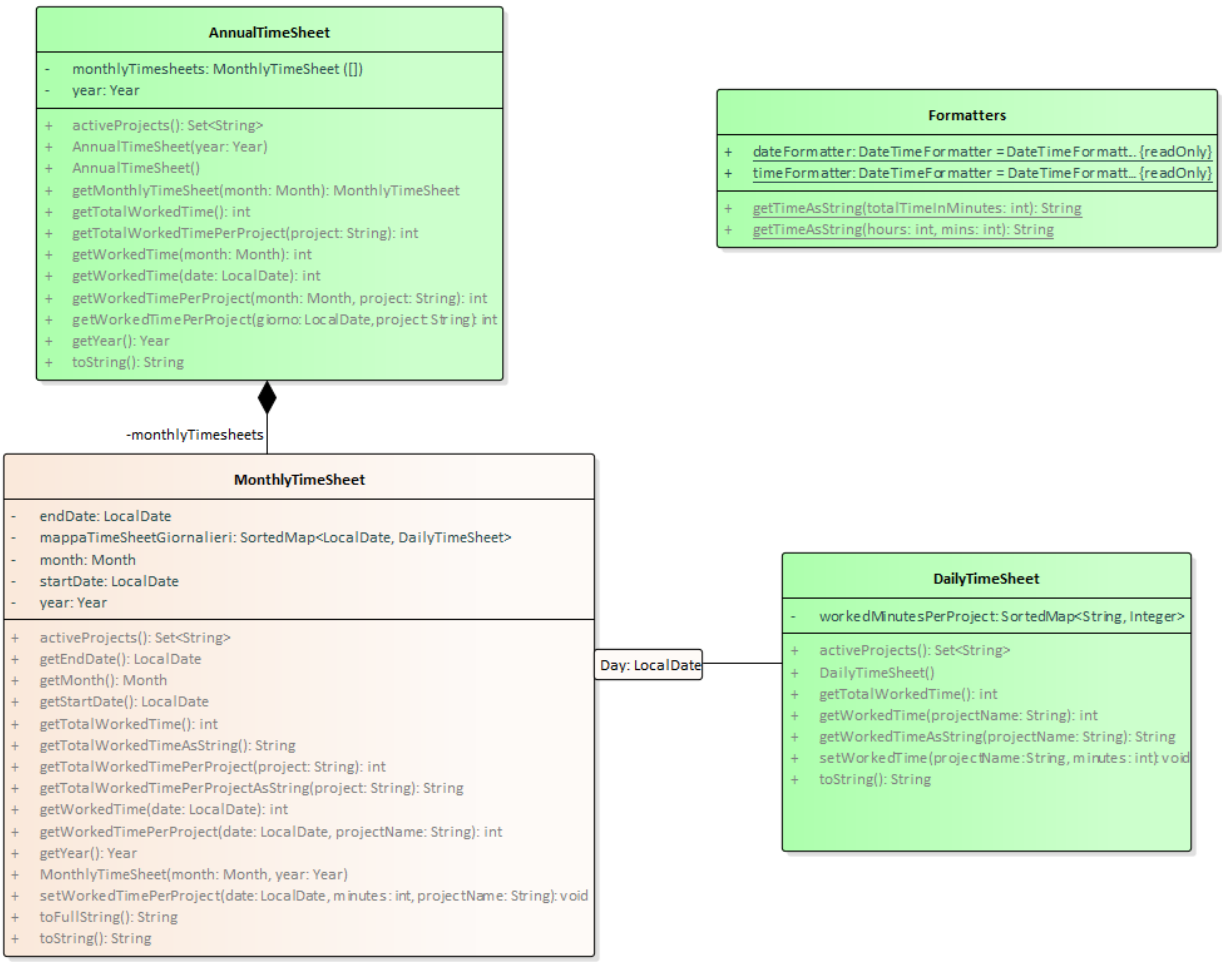
- salva costantemente il tuo lavoro: l'informatica a volte può essere "subdolamente ostile"..
- in particolare: se ora compila e stai per fare modifiche, salva la versione attuale (non si sa mai)

Checklist di consegna

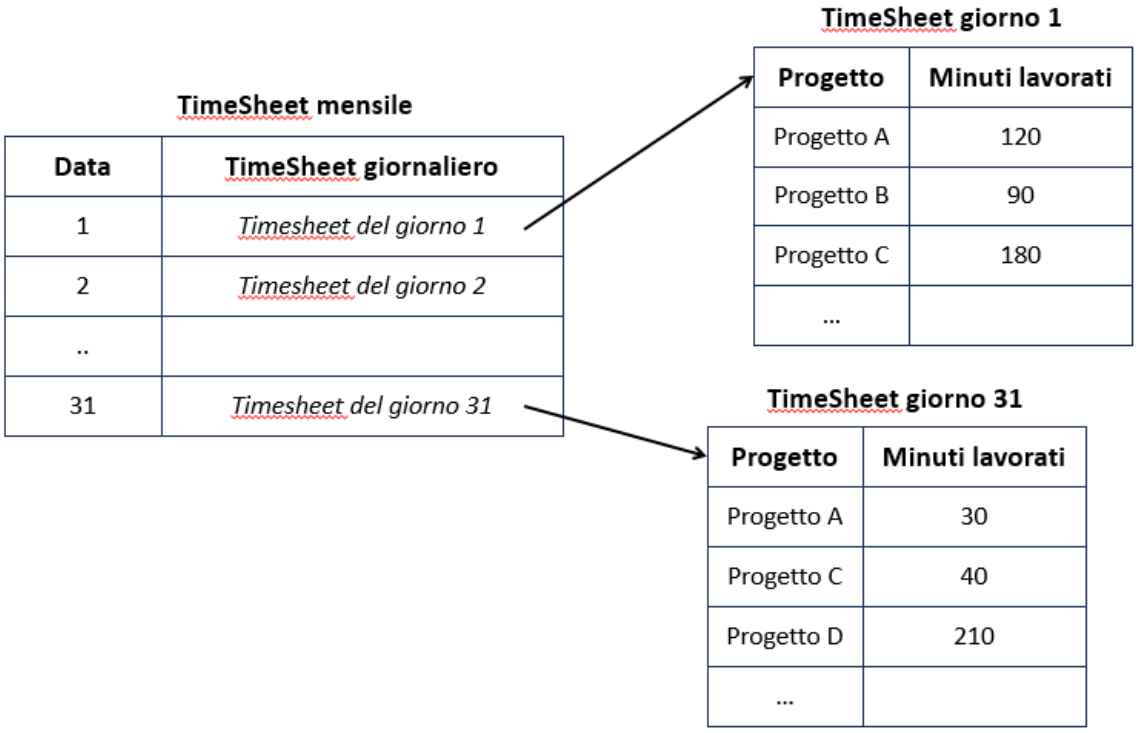
- Hai fatto un **JAR eseguibile**, che contenga cioè l'indicazione del main?
- Hai controllato che **si compili** e **ci sia tutto**? [NB: non includere il PDF del testo]
- Hai **rinominato** IL PROGETTO, lo ZIP e il JAR esattamente come richiesto?
- Hai **chiamato** la cartella del progetto esattamente 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?

Parte 1 – Modello dei dati (punti: 11)

Package: timesheet.model **[TEMPO STIMATO: 50-70 minuti]**



PREMESSA: poiché la tabella cartacea opera su tre dimensioni (giorni, progetti, ore lavorate) mentre la mappa Java esprime solo due dimensioni (colonne), occorre spezzare la rappresentazione su più strutture collegate. A tal fine, il ***timesheet mensile*** verrà espresso come mappa (giorni, timesheet giornalieri), in cui il ***timesheet giornaliero*** è a sua volta una mappa (progetti, minuti lavorati), come mostrato nella figura seguente:



SEMANTICA:

- La classe ***Formatters*** fornisce alcuni formattatori già configurati per ora e data in standard italiano, nonché due metodi getTimeAsString che forniscono una stringa già formattata nel formato HH:MM partendo, rispettivamente, o da un totale in minuti, o da ore e minuti espressi come interi.
- La classe ***DailyTimeSheet*** rappresenta il resoconto giornaliero, che riporta i minuti lavorati su ogni progetto. I progetti per i quali quel giorno non si è lavorato NON vengono riportati. Il costruttore alloca un resoconto vuoto. Oltre a un'apposita *toString*, sono forniti i seguenti metodi:
  - *setWorkedTime* imposta i minuti lavorati per un dato progetto: il metodo verifica preventivamente che il nome del progetto non sia nullo né blank, o che i minuti non siano negativi, lanciando nel caso, secondo le note linee guida, ***NullPointerException*** o ***IllegalArgumentException***.
  - *getWorkedTime* restituisce i minuti lavorati sul progetto specificato sotto forma di intero, verificando preventivamente la validità dell'argomento come sopra; il metodo gemello *getWorkedTimeAsString* restituisce il tempo lavorato sotto forma di stringa formattata come HH:MM
  - *getTotalWorkedTime* restituisce i minuti lavorati complessivamente quel giorno, in tutti i progetti
  - *activeProjects* restituisce l'insieme dei nomi dei progetti per i quali quel giorno si è lavorato.

- La classe ***MonthlyTimeSheet*** (da completare) rappresenta il resoconto mensile: a ogni giorno del mese è associata un'opportuna istanza di ***DailyTimeSheet***. Il costruttore riceve mese e anno, sotto forma di istanze rispettivamente di ***Month*** e ***Year****:* ne verifica la validità e alloca le istanze di ***DailyTimeSheet*** necessarie.
  - gli accessor *getMonth*, *getYear*, *getStartDate (*)*, *getEndDate (*)* restituiscono rispettivamente mese e anno di riferimento (i due argomenti ricevuti dal costruttore), nonché la data iniziale e finale del mese (ad esempio, per marzo 2024, rispettivamente 01/03/2024 e 31/03/2024), ovviamente tenendo conto della durata in giorni dello specifico mese.
  - *getTotalWorkedTime* restituisce i minuti totali lavorati nel mese, sotto forma di intero; il metodo gemello *getTotalWorkedTimeAsString* restituisce tale tempo lavorato sotto forma di stringa HH:MM
  - *getTotalWorkedTimePerProject* restituisce i minuti totali lavorati nel mese per uno specifico progetto: anche in questo caso, il metodo gemello *getTotalWorkedTimePerProjectAsString* restituisce il tempo lavorato sotto forma di stringa HH:MM.
  - *activeProjects* restituisce l'insieme dei nomi dei progetti per i quali quel mese si è lavorato.
  - Il metodo *getWorkedTime* **(da fare)** restituisce i minuti lavorati in un dato giorno, espresso sotto forma di ***LocalDate*****,** per tutti i progetti. Più precisamente, il metodo riceve la data richiesta, verifica preventivamente che essa sia non nulla e compresa nel mese (*), lanciando nel caso le opportune eccezioni con opportuni messaggi d'errore: solo in caso positivo accede al corrispondente ***DailyTimeSheet*** e recupera i minuti totali lavorati in quel giorno. Nel caso in cui non vi sia ancora alcun ***DailyTimeSheet*** istanziato per quel giorno, deve restituire (ovviamente) 0 minuti lavorati.
  - La coppia di metodi *getWorkedTimePerProject* / *setWorkedTimePerProject* **(entrambi da fare)** permette di recuperare / impostare i minuti lavorati in uno dato giorno, espresso sotto forma di ***LocalDate***, per un dato progetto. Più precisamente:
    - *getWorkedTimePerProject* riceve la data richiesta e il nome del progetto desiderato, verifica preventivamente che essa sia non nulla e compresa nel mese (*) e che il nome del progetto non sia nullo né blank, lanciando nel caso le opportune eccezioni con opportuni messaggi d'errore: solo in caso positivo accede al corrispondente ***DailyTimeSheet*** e recupera i minuti lavorati in quel giorno per lo specifico progetto;
    - *setWorkedTimePerProject* riceve la data relativa al giorno richiesto, i minuti lavorati e il nome del progetto a cui si riferiscono: verifica preventivamente che la data sia non nulla e compresa

nel mese (*), che i minuti non siano negativi e che il nome del progetto non sia nullo né blank, lanciando nel caso le opportune eccezioni con opportuno messaggio d'errore; solo se tutto fila liscio, accede al ***DailyTimeSheet*** del giorno specificato e imposta in esso i minuti lavorati per il progetto specificato.

  - *toString* restituisce la sintesi delle ore lavorate nel mese sui diversi progetti, mentre *toFullString* fornisce un livello di dettaglio maggiore, specificando le ore lavorate per ogni singolo giorno.

- La classe ***AnnualTimeSheet*** rappresenta il resoconto annuale, ottenuto per composizione dei dodici resoconti mensili: a tal fine, mantiene al suo interno un array di ***MonthlyTimeSheet***, indicizzato in base al mese. Il costruttore primario riceve l'anno di riferimento e alloca le opportune strutture interne, mentre il costruttore ausiliario senza argomenti assume di default l'anno corrente. Sono forniti i seguenti metodi:
  - *getYear* restituisce semplicemente l'anno a cui il resoconto si riferisce, come istanza di ***Year***
  - *getMonthlyTimeSheet* fornisce il riferimento al resoconto mensile interno relativo al mese specificato come argomento (che naturalmente non può essere nullo: altrimenti, ***NullPointerException***)
  - *getTotalWorkedTime* restituisce i minuti lavorati complessivamente nell'anno, per tutti i progetti
  - *getTotalWorkedTimePerProject* restituisce i minuti lavorati complessivamente nell'anno per il solo progetto ricevuto come argomento (se nullo, viene lanciata ***NullPointerException***)
  - *getWorkedTime(Month)* restituisce i minuti lavorati nel mese specificato, per tutti i progetti, sotto forma di intero; la validità dell'argomento è verificata come sopra
  - *getWorkedTime(LocalDate)* restituisce i minuti lavorati nel giorno specificato, per tutti i progetti, sotto forma di intero; la validità dell'argomento è verificata come sopra
  - *getWorkedTimePerProject(Month, String)* restituisce i minuti lavorati nel mese specificato, per il solo progetto specificato, sotto forma di intero; la validità dell'argomento è verificata come sopra
  - *getWorkedTimePerProject(LocalDate, String)* restituisce i minuti lavorati nel giorno specificato, per il solo progetto specificato, sotto forma di intero; la validità dell'argomento è verificata come sopra
  - *activeProjects* restituisce l'insieme dei nomi dei progetti per i quali si è lavorato in quell'anno
  - *toString* produce il resoconto annuale per composizione dei resoconti mensili.

Parte 2 – Persistenza (punti: 8)

Package: timesheet.persistence **[TEMPO STIMATO: 35-40 minuti]**

Il file di testo Projects.txt elenca i progetti attivi con il numero totale di ore previste per ciascuno. Ogni riga contiene:

- il nome del progetto (che può contenere spazi), seguito da una o più tabulazioni
- la frase "`ore previste:`", seguita da uno o più spazi
- un numero intero, che rappresenta il totale di ore previste complessivamente sul progetto

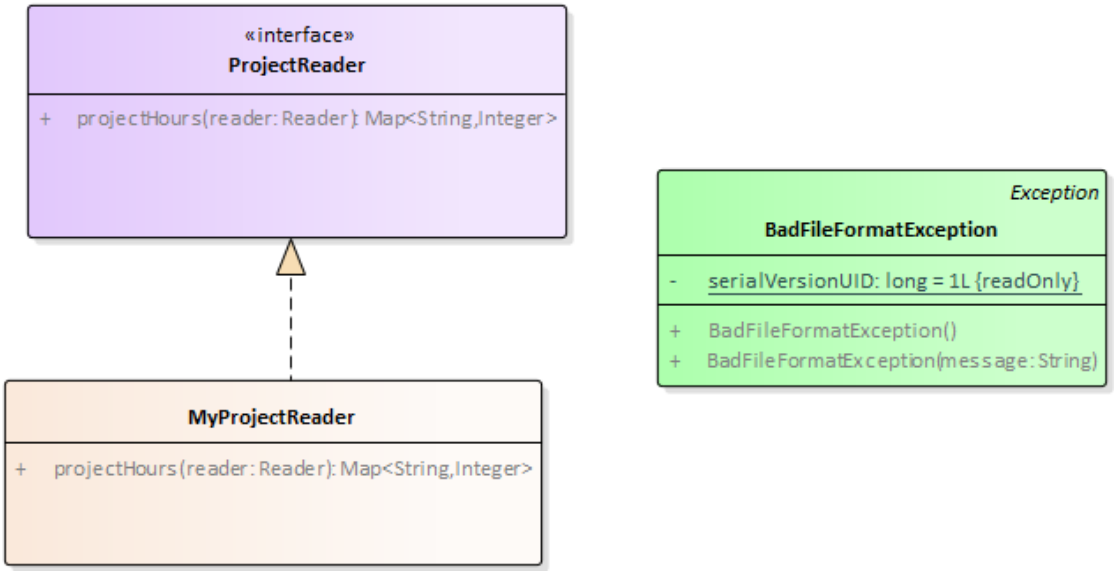
```
Lezioni di Fondamenti T2      ore previste: 80
Lezioni di Linguaggi          ore previste: 64
Progetto Peonie               ore previste: 350
Progetto Innova               ore previste: 700
...
```

SEMANTICA:

a) L'interfaccia ***ProjectReader*** (fornita) dichiara il metodo *projectHours,* che legge dal ***Reader*** fornito (già aperto) i dati e restituisce una ***Map<String,Integer>*** che associa a ogni progetto il rispettivo numero di ore; lancia ***BadFileFormatException*** con opportuno messaggio d'errore in caso di problemi nel formato del file, o ***IOException*** in caso di altri problemi di I/O.

b) La classe ***MyProjectReader*** (da realizzare) implementa ***ProjectReader***:

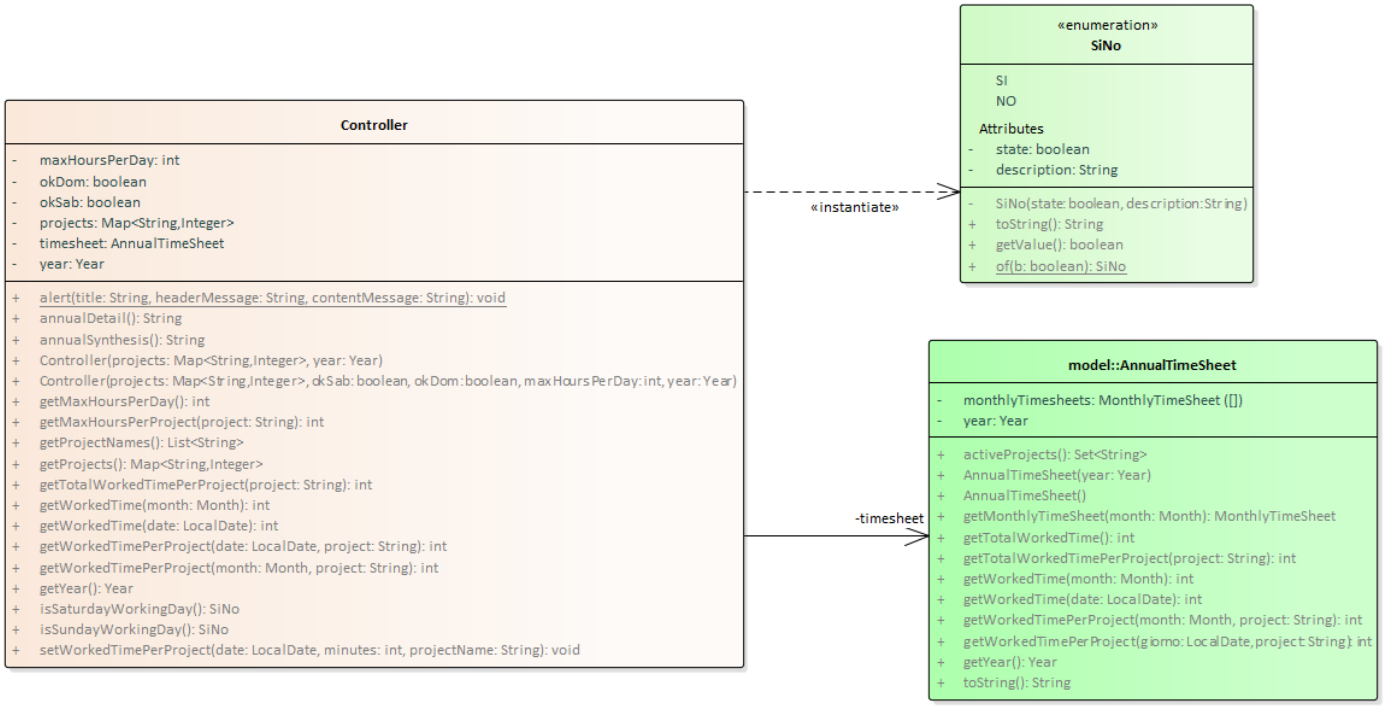
- non c'è alcun costruttore
- Il metodo *projectHours* (da implementare) effettua le letture: lancia ***NullPointerException*** in caso di reader nullo, o ***BadFileFormatException*** con dettagliato messaggio d'errore in caso di mancato rispetto del formato previsto, catturando eventuali altre eccezioni interne. In particolare deve verificare:
  - che gli elementi in ogni riga siano esattamente tre
  - che il secondo elemento sia la frase prevista
  - che l'ultimo elemento sia un intero strettamente positivo



Parte 3 (punti: 11)

Package: timesheet.controller **[TEMPO STIMATO: 10-15 minuti]** ***(punti 3)***

Il controller costituisce il nucleo centrale di questa applicazione: crea e mantiene le strutture dati, gestisce la logica di gioco e le modalità dello stesso. A tal fine è organizzato come segue:



SEMANTICA:

a) L'enumerativo ***SiNo*** esprime le due costanti **SI** e **NO** che incapsulano rispettivamente i booleani *true* e *false*, accoppiando però a ciascuno la descrizione testuale italiana, restituita poi da *toString*; l'accessor *getValue* consente di recuperare il valore booleano associato alle due costanti enumerative, mentre il metodo factory *of* consente, dualmente, di ottenere l'istanza di ***SiNo*** corrispondente al boolean ricevuto come argomento.

b) La classe ***Controller*** **(da completare nei costruttori)** governa il funzionamento dell'applicazione: mantiene al suo interno sia la mappa dei progetti ricevuta dal costruttore, sia l'***AnnualTimeSheet*** che costituisce lo stato dell'applicazione stessa. Più precisamente:

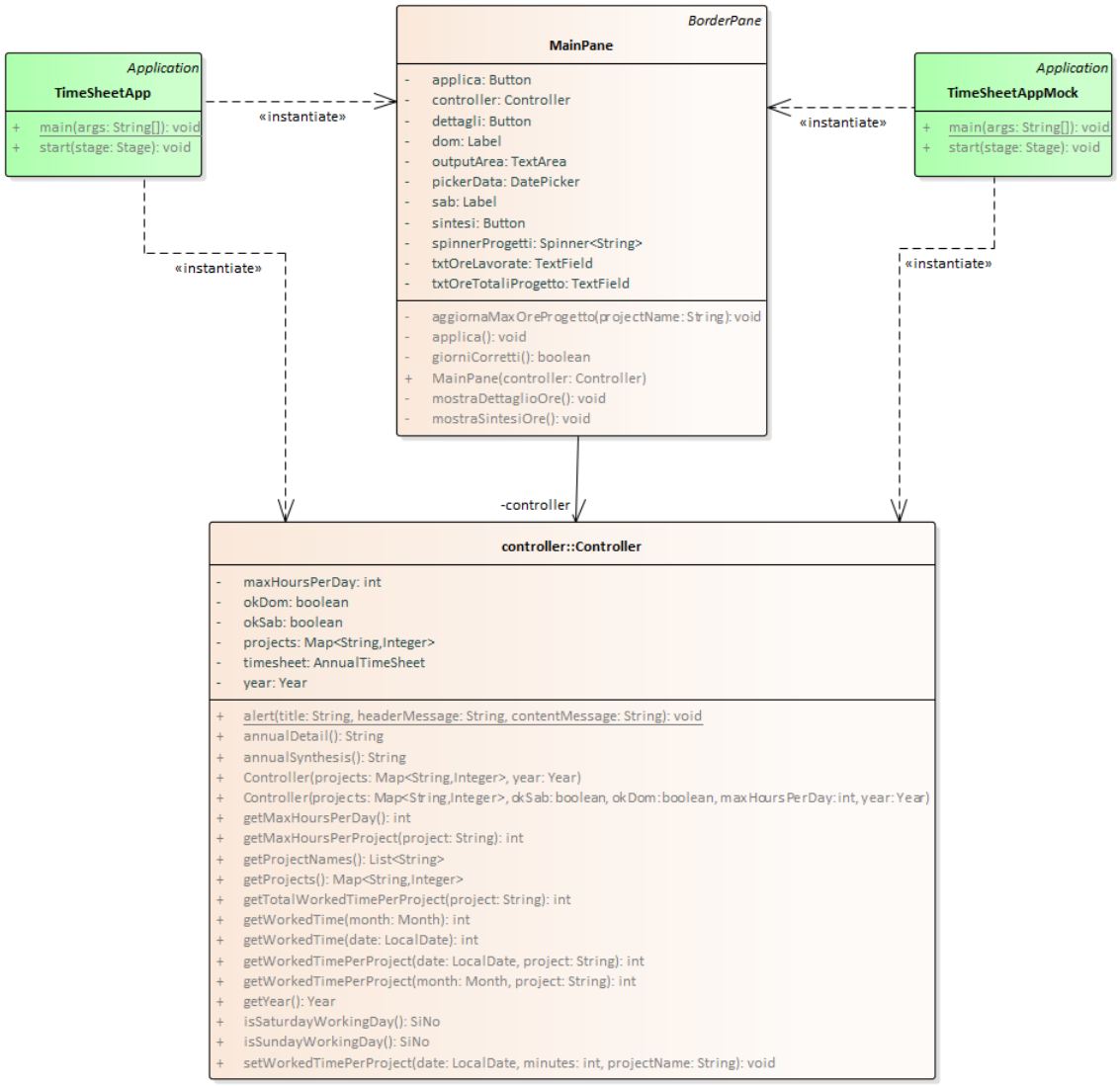
- Il costruttore primario (da fare) riceve la **Map<String,Integer>** restituita dalla persistenza, due booleani che esprimono l'eventuale possibilità di lavorare anche il sabato o la domenica, il numero massimo di ore lavorabili al giorno e l'anno di lavoro: dopo aver validato gli argomenti, memorizza il riferimento alla struttura dati ricevuta e crea l'***AnnualTimeSheet*** per l'anno specificato, su cui opererà da ora in poi.
- Il costruttore ausiliario (da fare) a due soli argomenti configura il controller nel caso più standard (lavoro permesso solo da lunedì a venerdì, max 8 ore lavorative al giorno) per l'anno specificato.
- Opportuni metodi delegano le operazioni alle corrispondenti classi del model. Più precisamente:
  - *getProjects* restituisce semplicemente la **Map<String,Integer>** ricevuta dal costruttore
  - *getProjectNames* restituisce i nomi dei progetti elencati in tale mappa (una lista di stringhe)
  - *isSaturdayWorkingDay* / *isSundayWorkingDay* verificano se rispettivamente sabato / domenica siano giorni lavorativi, restituendo il risultato sotto forma di istanza dell'enumerativo ***SiNo***
  - *getYear* restituisce l'anno specificato al costruttore, a cui si riferiscono le ore di lavoro
  - *getMaxHoursPerDay* restituisce il numero massimo di ore lavorative giornaliere
  - *getMaxHoursPerProject* restituisce il numero di ore previste per un progetto, sotto forma di intero
  - *getTotalWorkedTimePerProject* restituisce il numero totale di minuti lavorati per un dato progetto
  - *getWorkedTime(Month)* restituisce i minuti lavorati nel mese specificato, per tutti i progetti
  - *getWorkedTime(LocalDate)* restituisce i minuti lavorati nel giorno specificato, per tutti i progetti
  - *getWorkedTimePerProject(Month, String)* restituisce i minuti lavorati nel mese specificato, per il solo progetto specificato
  - *getWorkedTimePerProject(LocalDate, String)* restituisce i minuti lavorati nel giorno specificato, per il solo progetto specificato
  - *setWorkedTimePerProject(LocalDate, int, String)* imposta i minuti lavorati nel giorno specificato, per il progetto specificato: verifica che i minuti non siano negativi, che gli argomenti non siano nulli e che il nome del progetto non sia vuoto o blank
  - *annualDetail* restituisce una stringa che fornisce il resoconto dettagliato dell'anno
  - *annualSynthesis* restituisce invece una stringa che fornisce il resoconto sintetico dell'anno

  Il metodo statico *alert*, utilizzabile dal ***MainPane*** quando è attiva la grafica, consente di far comparire all'utente una finestra di dialogo che mostri il messaggio d'errore specificato.

Package: timesheet.ui **[TEMPO STIMATO: 40-55 minuti]** ***(punti 8)***

La classe ***TimeSheetApp*** (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 ***TimeSheetAppMock***.

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



- <u>In alto</u> sono riportate indicazioni statiche relative alla possibilità di lavorare il sabato e la domenica, e al numero massimo di ore giornaliere permesse.
- <u>Al centro</u> vi sono tutti i controlli per la scelta del progetto (uno ***Spinner*** di stringhe) e della data (un ***DatePicker***), l'inserimento delle ore lavorate (un ***TextField***) e i vari pulsanti (***Applica***, ***Dettaglio ore***, ***Sintesi ore***) che attivano le varie funzionalità dell'applicazione.
- <u>In basso</u> vi è l'area di output, che mostra i messaggi destinati all'utente.

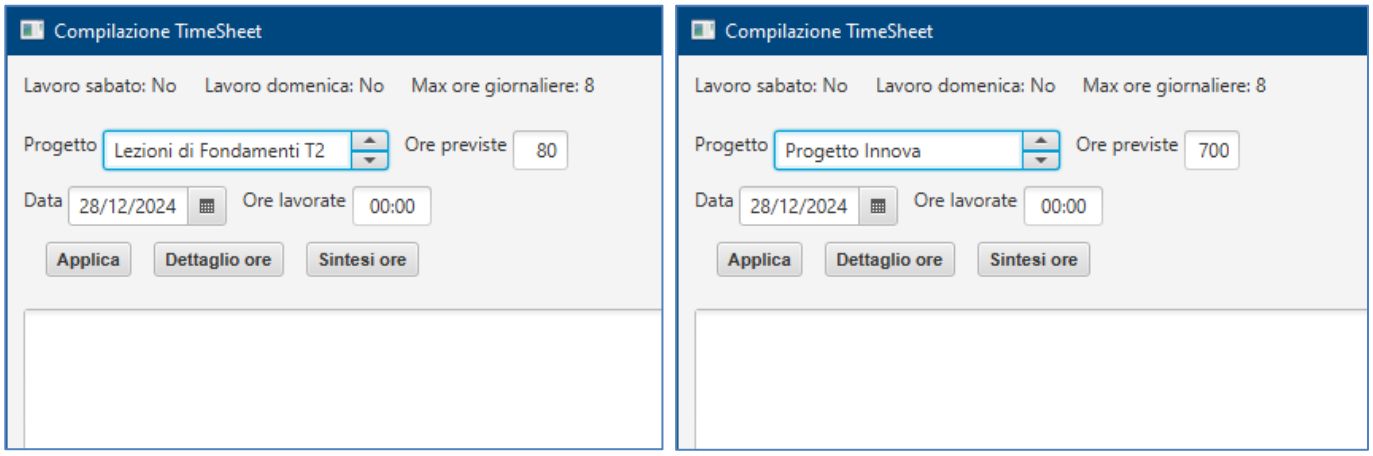


Fig. 1: situazione iniziale: notare le ore previste, adeguate al progetto mostrato nello spinner a sinistra.

Operando sulle frecce dello spinner si può cambiare progetto: le ore indicate a lato cambiano di conseguenza.

Agendo sul calendario si può scegliere la data di proprio interesse (Fig 2, sinistra) e inserire le ore lavorate premendo il pulsante ***Applica***: l'applicazione risponderà mostrando un messaggio di conferma, Nel caso il giorno scelto sia sabato o domenica, se l'impostazione non lo permette verrà mostrato un messaggio d'errore (Fig 2, destra) e non verrà effettuata alcuna operazione.

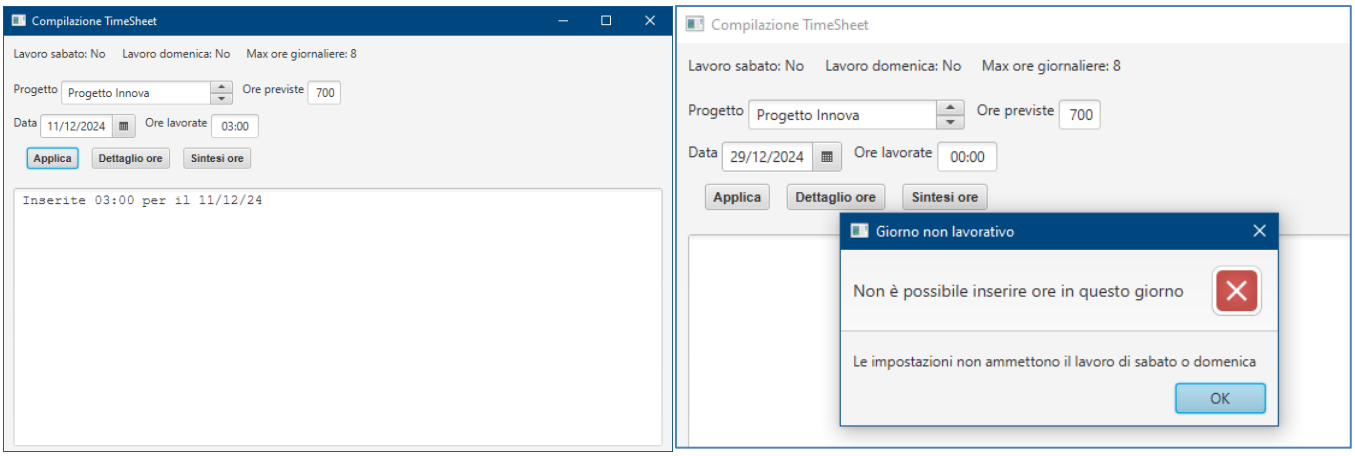


Fig. 2: selezione data e inserimento ore lavorate.

Procedendo a scegliere date e inserire ore per i vari progetti, si può via via compilare tutto il rendiconto. In ogni momento, il pulsante ***Dettaglio ore*** mostra il dettaglio delle ore lavorate in ogni singolo giorno per ogni progetto (Fig.3), mentre il pulsante ***Sintesi ore*** fornisce la visione d'insieme (Fig. 4).

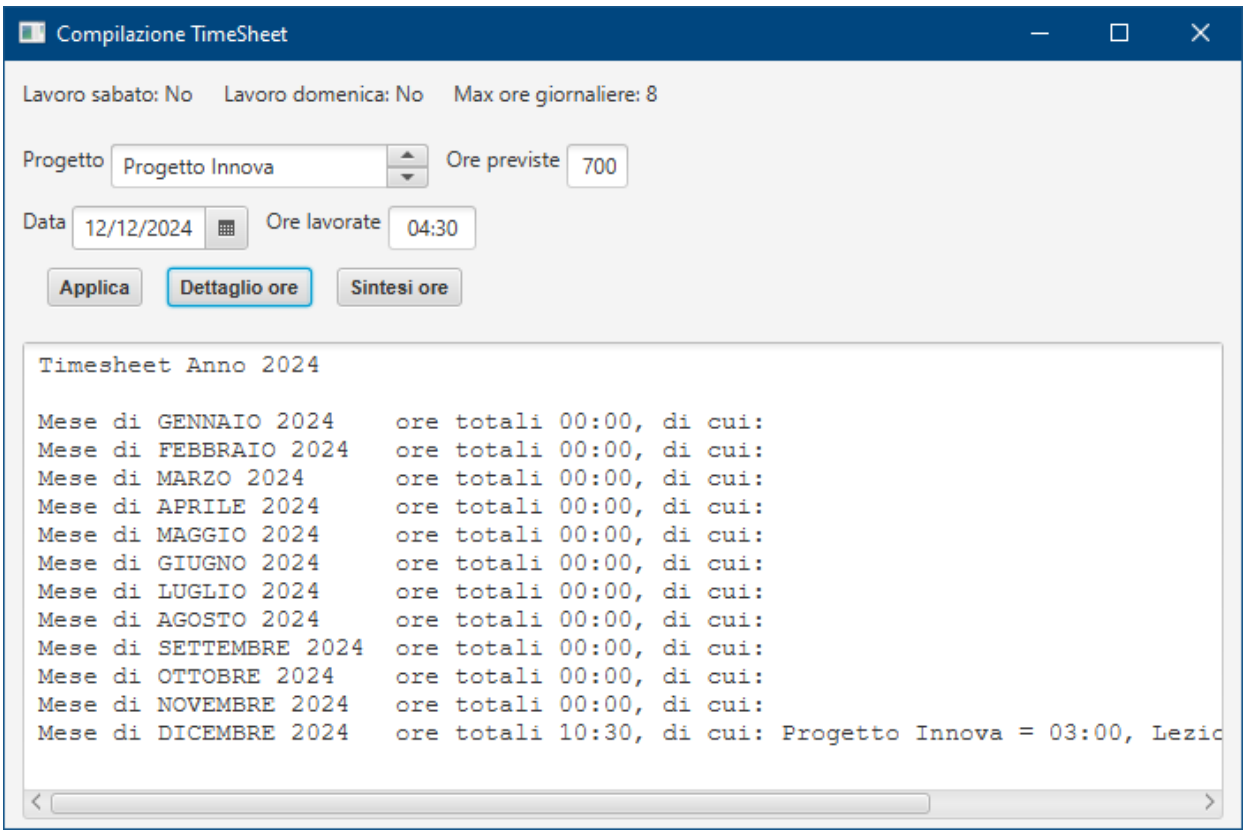


Fig. 3: dettaglio ore lavorate: per ogni mese vengono indicate le ore totali e la ripartizione fra i singoli progetti

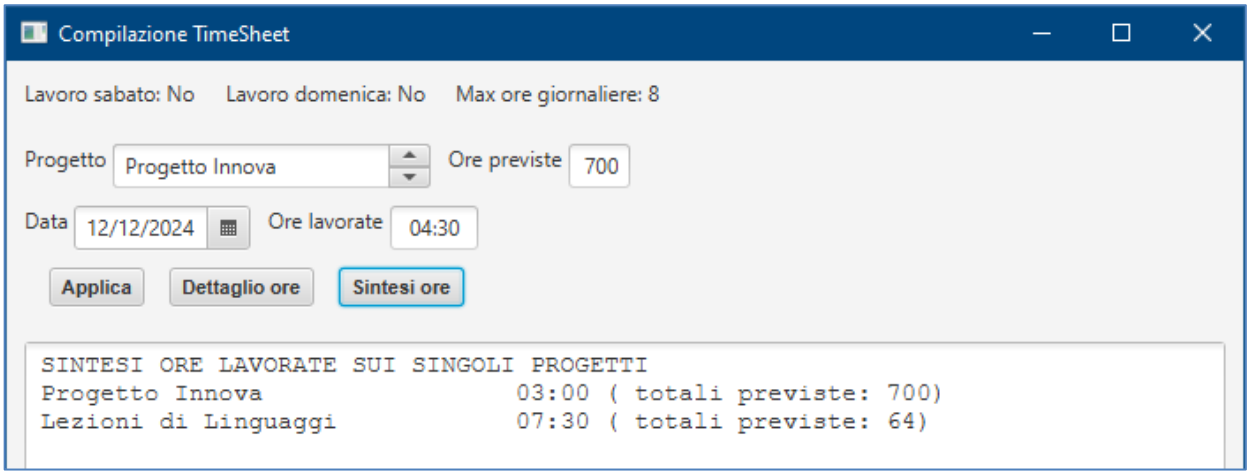
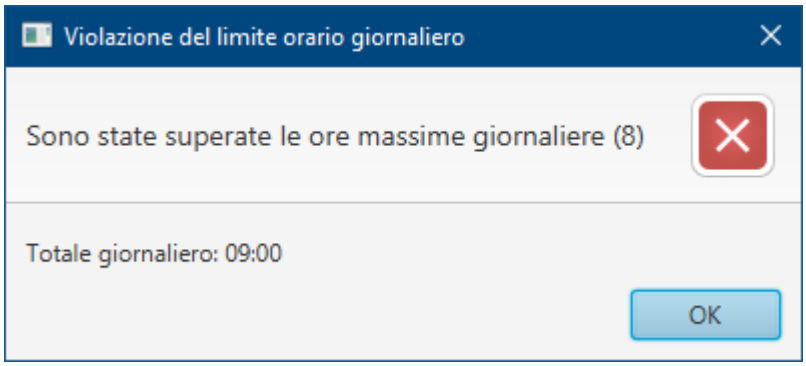


Fig. 4: sintesi ore lavorate: vengono elencate le ore complessivamente svolte in ogni progetto, con a fianco quelle totali previste per il progetto stesso.

Nel caso in cui si tenti di inserire, in un dato giorno, più ore di quelle massime permesse, viene emesso un messaggio d'errore (Fig. 5)



Il MainPane è fornito *quasi completamente realizzato*: è presente tutta la parte strutturale, mentre rimangono da realizzare due gestori degli eventi (metodi privati *applica* e *giorniCorretti*).

In particolare:

- *giorniCorretti* viene invocato all'atto della selezione di una data sul datepicker e deve verificare se il giorno selezionato sia lavorativo, in base alle impostazioni correnti dell'applicazione; più precisamente:
  - i giorni da lunedì a venerdì sono sempre lavorativi
  - il sabato e la domenica lo sono solo se ciò è permesso dall'impostazione iniziale del costruttore

  Nel caso il giorno selezionato risulti non lavorativo, il metodo deve emettere un messaggio d'errore tramite il metodo statico *alert* del Controller e restituire *false*; in caso invece di esito positivo, deve semplicemente restituire *true*.
- *applica* è associato alla gestione del pulsante omonimo e deve gestire tutto l'inserimento delle ore con i necessari controlli. In particolare, deve:
  - recuperare le ore lavorate, controllarne il formato e, se tutto risulta corretto, verificare tramite *giorniCorretti* che la data selezionata sia un giorno lavorativo valido.
  - solo in caso positivo, deve verificare che, aggiungendo le ore attuali, non si superi il massimo numero di ore lavorative giornaliere [NB: nel ricalcolo si presti attenzione a scorporare le ore lavorate già inserite in precedenza: ad esempio, se in precedenza si fossero inserite per quel giorno 5 ore, deve essere possibile cambiarle con 7, evitando che la semplice somma 5+7 = 12 dia la falsa impressione di aver superato il massimo giornaliero, es. di 8 ore]

- solo se anche questa verifica è positiva, recuperare dallo ***Spinner*** il nome del progetto e impostare le ore lavorate tramite il metodo *setWorkedTimePerProject* del ***Controller***, emettendo nell'area di testo un apposito messaggio di conferma (vedere Fig. 2 sopra).
- In tutti i casi di errore sopra evidenziati, il metodo dovrà emettere tramite *alert* un apposito, dettagliato messaggio d'errore all'utente.

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?