ESAME DI FONDAMENTI DI INFORMATICA T-2 del 22/7/2022
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"

È stato richiesto di sviluppare un'applicazione che consenta all'ufficio personale della ditta *HappyWork* di tenere traccia delle ore svolte/da svolgere da ogni dipendente, come meglio descritto di seguito.

DESCRIZIONE DEL DOMINIO DEL PROBLEMA

Il contratto di lavoro fissa, per ogni dipendente, le ***ore di lavoro*** da svolgere giorno per giorno in una settimana lavorativa. Esse possono essere anche diverse da un giorno all'altro (ad esempio, una settimana lavorativa di 36h potrebbe essere articolata sia su sei giorni da 6h ciascuno, sia su cinque giorni, di cui due da 9h e tre da 6h, etc.).

Un sistema *marcatempo* registra le ore lavorate: i dipendenti che entrano/escono dal lavoro devono timbrare le entrate, le uscite e le pause pranzo. Più precisamente il marcatempo registra:
- **l'ora di entrata** e **l'ora di uscita** dal lavoro (anche più volte al giorno, es. mattina, pomeriggio, etc.)
- **l'ora di inizio** e **l'ora di fine** della **pausa pranzo**.

Tutti gli orari si intendono nella stessa giornata: non esistono turni a cavallo della mezzanotte (se esistessero sarebbero comunque spezzati come se si entrasse/uscisse in due giornate diverse).

Le ***ore lavorate*** si calcolano banalmente facendo la differenza fra l'ora di uscita e l'ora di entrata, *detratta la pausa pranzo*. Nei giorni non lavorativi (di norma sabato, domenica e festivi) chiaramente non risultano timbrature nel sistema marcatempo, dato che il dipendente non è stato presente sul posto di lavoro.

Il dipendente può anche prendere ***permessi***, che possono essere solo per alcune ore ("riposo compensativo a ore") o per l'intera giornata ("riposo compensativo"): anche queste voci vengono registrate dal sistema marcatempo.

Al termine di ogni mese, tutte le voci registrate dal marcatempo sono assemblate nel ***cedolino,*** che riassume i dati del dipendente (nome e cognome, ore da svolgere nei vari giorni della settimana) e le varie timbrature effettuate, nonché i permessi che il dipendente ha chiesto.

Per garantire adeguata flessibilità, il dipendente può svolgere *più o meno ore* di quelle previste in una data giornata, in modo molto libero, grazie all'istituzione della ***banca ore***. Ogni giorno si calcola la differenza fra le ore previste e quelle effettivamente svolte: se il dipendente ha svolto meno ore di quelle previste, quelle mancanti saranno detratte dal suo conto in banca ore; viceversa, se ne svolge di più, le ore extra verranno accumulate sul suo conto in banca ore. E' anche possibile andare "a debito", ossia avere sulla banca ore saldi negativi: le ore mancanti dovranno poi essere recuperate in periodi successivi. I permessi si considerano come ore non svolte: le corrispondenti ore vengono quindi detratte dal conto in banca ore.

Per ogni dipendente, la banca ogni giorno registra:
- **le ore di lavoro previste** per quella giornata (possono essere diverse da un giorno della settimana all'altro)
- **le ore lavorate** (esclusa, come detto, la pausa pranzo)
- **il saldo ore disponibili sul conto** del lavoratore (positivo o negativo).

Al termine di ogni mese occorre quindi produrre **l'*estratto conto*** della banca ore, con tutti i dettagli delle ore lavorate e dei permessi fruiti, nonché il ***saldo finale*** (positivo o negativo) delle ore presenti sul conto.

A differenza del cedolino, l'estratto conto deve elencare tutti i giorni del mese, anche quelli in cui non si è lavorato e non risultano quindi timbrature.

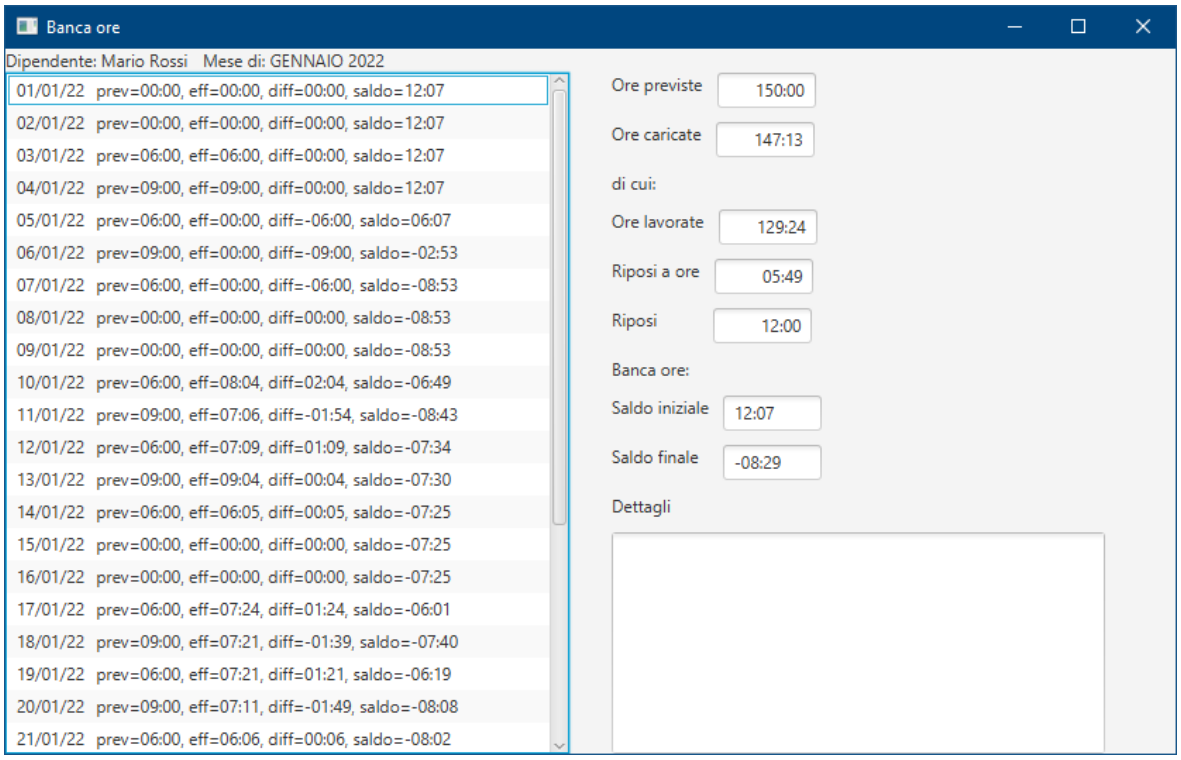
Il file **Cedolino.txt** contiene le timbrature mensili di un dipendente, nel formato descritto più oltre.
L'applicazione dovrà generare a video l'estratto conto della banca ore, nel formato illustrato nelle figure seguenti.

ESEMPIO DI CEDOLINO (da leggere): il formato è descritto in dettaglio più oltre

```
Dipendente:       Mario Rossi
Mese di:          GENNAIO 2022
Ore previste:      6H/9H/6H/9H/6H/0H/0H
Saldo precedente: 12H07M

03 Lunedì         08:30    14:30
04 Martedì        08:30    17:30
05 Mercoledì      08:30    14:30    Riposo Compensativo
07 Venerdì        08:30    14:30    Riposo Compensativo
10 Lunedì         07:30    13:42
10 Lunedì         13:42    13:53    Pausa Pranzo
10 Lunedì         13:53    15:45
11 Martedì        07:30    13:28
11 Martedì        13:28    13:38    Pausa Pranzo
11 Martedì        13:38    14:46
11 Martedì        14:46    16:40    Riposo Compensativo a Ore
12 Mercoledì      07:30    13:49
12 Mercoledì      13:49    13:59    Pausa Pranzo
12 Mercoledì      13:59    14:49
13 Giovedì        07:30    13:39
13 Giovedì        13:39    13:49    Pausa Pranzo
13 Giovedì        13:49    16:44
...
```

ESEMPIO DI ESTRATTO CONTO A VIDEO (il formato è descritto in dettaglio più oltre)



PARTE 1 – Modello dei dati: Punti 10 [TEMPO STIMATO: 50-65 minuti]
PARTE 2 – Persistenza: Punti 15 [TEMPO STIMATO: 75-95 minuti]
PARTE 3 – Grafica: Punti 5 [TEMPO STIMATO: 10-20 minuti]

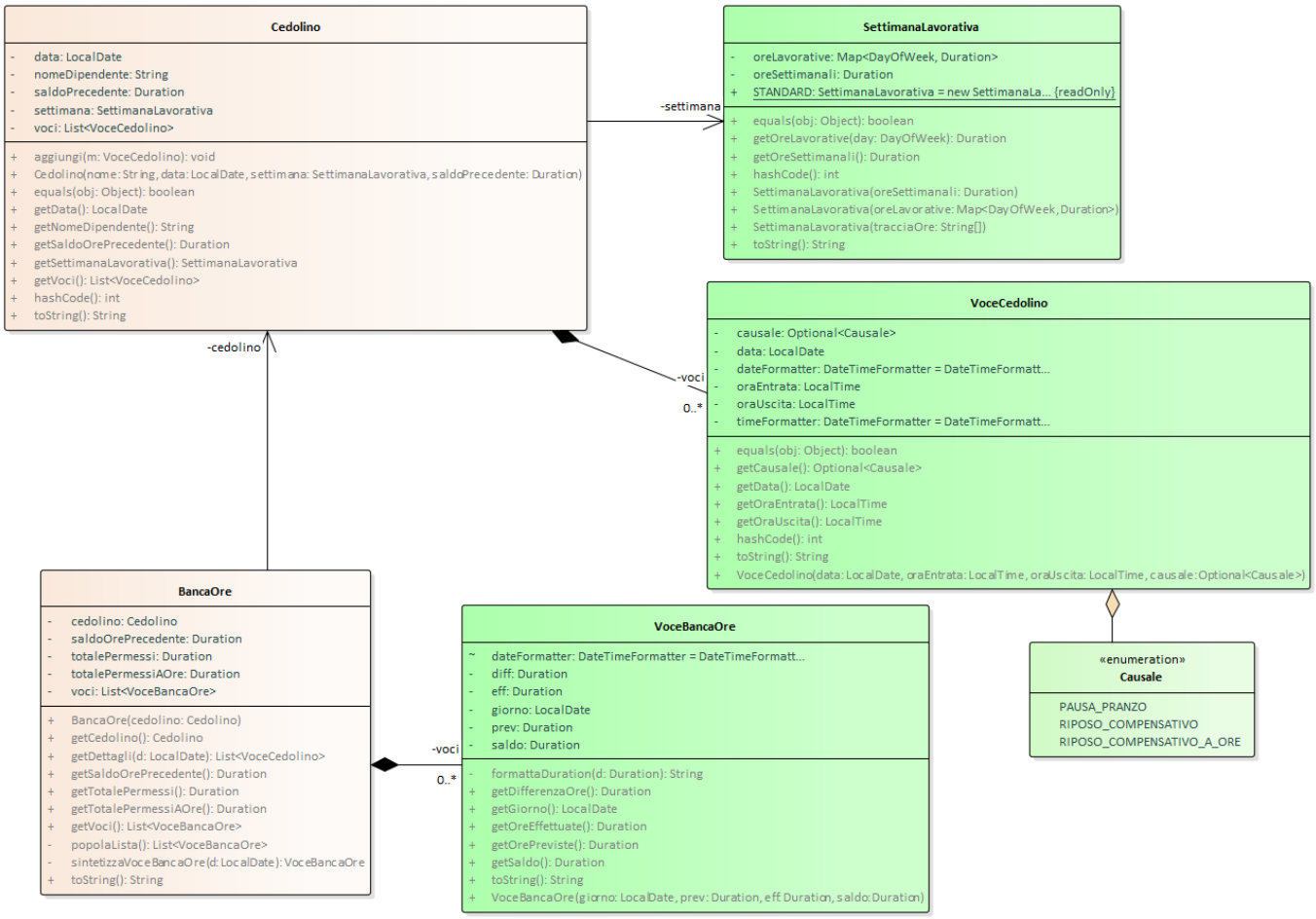
TEMPO STIMATO PER SVOLGERE L'INTERO COMPITO: 2h15 – 3h

Parte 1 – Modello dei dati

Punti: 10

Package: bancaore.model

[TEMPO STIMATO: 50-65 minuti]



SEMANTICA:

a) l'enumerativo ***Causale*** (fornito) definisce le tre possibili causali relative alle pause e ai permessi: PAUSA PRANZO, RIPOSO COMPENSATIVO (giornaliero), RIPOSO COMPENSATIVO A ORE.

b) La classe ***SettimanaLavorativa*** (fornita) rappresenta le ore lavorative previste per un lavoratore nei sette giorni della settimana, sotto forma di altrettante ***Duration***: i tre costruttori accettano o una durata globale (***Duration***) da ripartire convenzionalmente in parti uguali sui cinque giorni lunedì-venerdì, o una mappa che associa a ogni giorno della settimana (***DayOfWeek***) le rispettive durate, o una stringa di sette durate separate da barre nel formato "*n*H*m*M/ *n*H*m*M…/*n*H*m*M", in cui ogni "*n*H*m*M" rappresenta una durata di *n* ore e *m* minuti, da intendersi da lunedì a domenica. ESEMPIO, "6H/9H/6H/9H/6H/0H/0H" rappresenta una settimana lavorativa di 36 ore, di cui 6 da svolgere il lunedì, mercoledì e venerdì, 9 martedì e giovedì, mentre sabato e domenica sono liberi.
I due metodi *getOreSettimanali* e *getOreLavorative* restituiscono una ***Duration*** che rappresenta, rispettivamente, il totale delle ore lavorative settimanali previste e le ore previste per il giorno della settimana (***DayOfWeek***) specificato.

c) la classe ***VoceCedolino*** (fornita) rappresenta una voce del cedolino, che contiene data, orario di entrata e uscita, ed eventuale ***Causale*** (optional): in particolare, le voci con causale istanziata rappresentano pause o permessi, mentre quelle con causale vuota rappresentano ore lavorate. Appositi accessor consentono di estrarre le proprietà rilevanti: opportune *equals*, *hashCode* e *toString* completano l'implementazione.

d) la classe ***Cedolino*** **(da completare nel metodo pubblico *aggiungi*)** rappresenta il cedolino, composto dai dati di intestazione (nome lavoratore, mese e anno a cui il cedolino si riferisce, settimana lavorativa, saldo banca ore al mese precedente) e una lista di ***VoceCedolino.*** Il costruttore crea un cedolino con lista voci inizialmente vuota, che dovrà poi essere via via popolata tramite il **metodo *aggiungi* (da implementare).** Poiché possono essere presenti più voci per la stessa data, la lista delle varie voci deve essere mantenuta ordinata per data crescente e in subordine per ora di entrata crescente: pertanto, *il metodo aggiungi deve garantire che l'inserimento di una nuova voce avvenga in modo da mantenere la lista costantemente ordinata secondo tale criterio*. (NB: non è richiesto verificare che le varie voci siano coerenti, ossia che non si determinino "buchi" o ci siano sovrapposizioni: ci si affida per questo all'hardware del sistema marcatempo).
Il cedolino mantiene al suo interno anche il totale delle ore di permesso (sia giornalieri che a ore) richiesti nel mese. Appositi metodi accessor consentono di estrarre le proprietà rilevanti, mentre idonee *equals*, *hashCode* e *toString* completano l'implementazione.

e) la classe ***VoceBancaOre*** (fornita) rappresenta una voce della banca ore, che contiene data, ore previste, ore effettivamente lavorate e saldo attuale della banca alla data specificata. Anche in questo caso, appositi accessor consentono di estrarre le proprietà rilevanti e un'apposita *toString* emette una stringa ben formattata che riassume la voce stessa.

f) la classe ***BancaOre*** **(da completare nel metodo privato *sintetizzaVoceBancaOre*)** costituisce la banca ore: a tal fine incapsula un ***Cedolino,*** ricevuto come argomento dal costruttore, e configura la banca a partire dai dati in esso contenuti. Appositi metodi accessor consentono, al solito, di estrarre le proprietà rilevanti. La banca mantiene al proprio interno una lista di ***VoceBancaOre,*** popolata dal metodo ausiliario *popolaLista (fornito),* che garantisce la generazione di una e una sola ***VoceBancaOre*** per ogni giorno del mese. A sua volta questi si appoggia, per produrre l'oggetto ***VoceBancaOre*** relativo a una specifica data, al metodo privato ausiliario ***sintetizzaVoceBancaOre*** **(da implementare)**, che cura la generazione della ***VoceBancaOre*** per il giorno richiesto. A tal fine, esso deve:

- recuperare tutte le voci presenti nel cedolino per la data richiesta *(NB: potrebbero non essercene, dato che non tutte le date del mese sono in generale presenti nel cedolino)*
- calcolare le ore di lavoro globalmente effettuate nella giornata richiesta, escludendo pause pranzo e riposi di qualsiasi tipo
- recuperare le ore previste per quel giorno della settimana
- calcolare la differenza (positiva o negativa) fra le ore previste e quelle effettivamente svolte, aggiornando poi, di conseguenza, il saldo della banca ore
- nel caso di riposi compensativi (giornalieri o ad ore), limitarsi ad aggiornare i relativi totali.

ESEMPIO: per la giornata di lunedì 10 gennaio, in cui il cedolino riporta:

10 Lunedì 07:30	13:42	
10 Lunedì 13:42	13:53	Pausa Pranzo
10 Lunedì 13:53	15:45	

la voce banca ore da sintetizzare deve riportare: ore previste 6 (perché il lunedì la settimana lavorativa del dipendente prevede 6 ore), ore effettivamente svolte 8:04 (6:12 al mattino + 1:52 al pomeriggio), differenza +2:04 e quindi, poiché il saldo precedente era -8:53, nuovo saldo -6:49.

Analogamente, il giorno successivo, martedì 11 gennaio, in cui il cedolino riporta:

11 Martedì	07:30	13:28	
11 Martedì	13:28	13:38	Pausa Pranzo
11 Martedì	13:38	14:46	
11 Martedì	14:46	16:40	Riposo Compensativo a Ore

la voce sintetizzata deve riportare: ore previste 9 (perché è martedì), ore effettivamente svolte 7:06 (5:58 al mattino + 1:08 al pomeriggio), differenza -1:54 e quindi nuovo saldo -8:43.

Parte 2 – Persistenza — Punti: 15

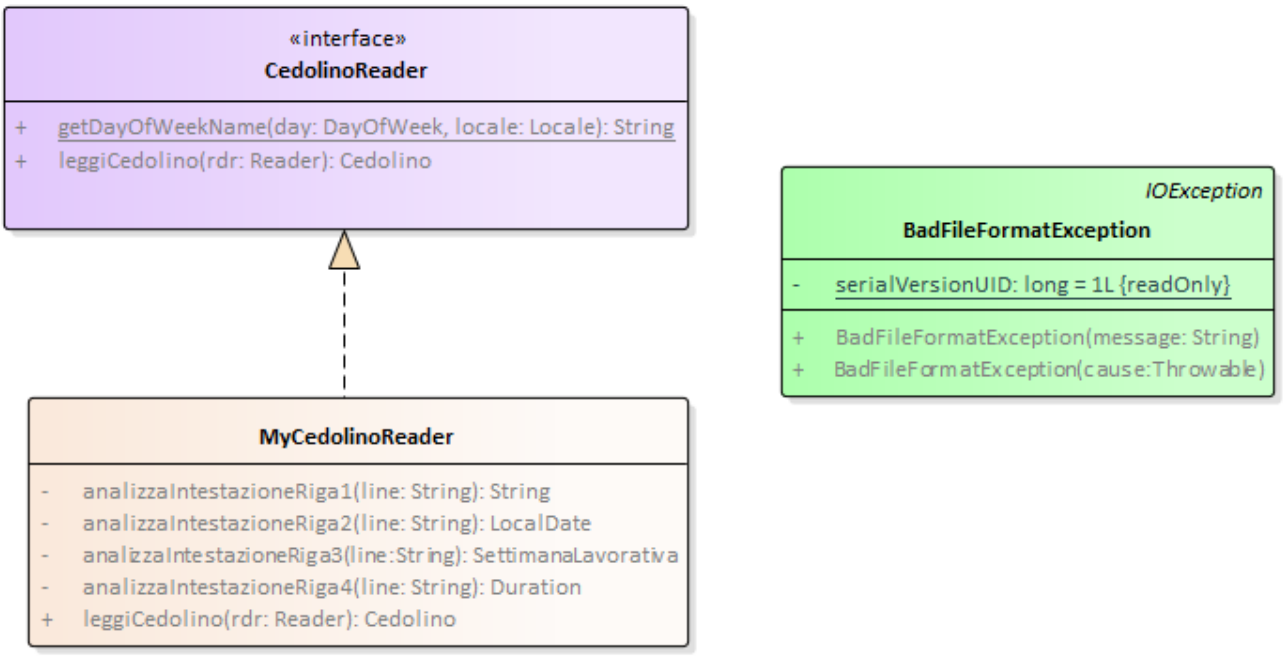
Package: bancaore.persistence — **[TEMPO STIMATO: 75-95 minuti]**

Come anticipato sopra, il file di testo **Cedolino.txt** contiene le timbrature mensili di un dipendente. Le prime quattro righe costituiscono un'intestazione, che specifica:

- il nome del dipendente
- il mese e anno a cui il cedolino si riferisce
  NB: il nome del mese può essere scritto con qualunque mix di maiuscole/minuscole
- le ore lavorative previste nella settimana
- il saldo precedente della banca ore di questo lavoratore, nella forma nHmM (n=ore, m=minuti)

Le righe successive (fra cui possono esservi anche righe vuote, utili soltanto a livello estetico) descrivono ciascuna una voce del cedolino, tramite tre o quattro elementi separati da tabulazioni:

- il giorno del mese (01-31) e della settimana (lunedì, martedì, ...), separati fra loro da uno o più spazi
  NB: il giorno della settimana può essere scritto con qualunque mix di maiuscole/minuscole
- gli orari di entrata e di uscita, nel formato italiano SHORT
- eventualmente, per le sole timbrature relative a pause pranzo o riposi, la causale, intesa come stringa che può valere "Riposo compensativo", "Riposo compensativo a ore" o "Pausa pranzo", con un qualunque mix di maiuscole/minuscole; le timbrature relative a ore effettivamente lavorate sono prive di causale.



SEMANTICA:

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

b) L'interfaccia ***CedolinoReader*** (fornita) dichiara il metodo *leggiCedolino,* che legge da un *Reader* (ricevuto come argomento) i dati di un intero cedolino, configurando e restituendo l'opportuno oggetto ***Cedolino.*** Tale interfaccia ospita anche il metodo statico di utilità *getDayOfWeekName*, che consente di ottenere la stringa (minuscola) corrispondente al giorno della settimana fornito come argomento, espresso nella cultura locale passata come secondo argomento (ESEMPIO: passando DayOfWeek.THURSDAY e Locale.ITALY si otterrà in uscita la stringa "giovedì")

c) La classe ***MyCedolinoReader*** (**da completare**) implementa ***CedolinoReader***: non prevede costruttori e implementa il metodo ***leggiCedolino*** in accordo alle specifiche sopra definite. In caso di problemi di I/O deve propagare l'opportuna ***IOException,*** in caso di ***Reader*** nullo ***IllegalArgumentException*** e in caso di altri problemi di formato dei file ***BadFileFormatException***, il cui messaggio dettagli l'accaduto.

Al fine di modularizzare opportunamente la lettura, *leggiCedolino* si appoggia a quattro metodi privati *analizzaIntestazioneRiga1, …, analizzaIntestazioneRiga4* che elaborano ciascuno una delle quattro righe di intestazione, restituendo l'opportuno oggetto – o lanciando ***BadFileFormatException*** se necessario. **Di questi quattro, l'unico da realizzare è *analizzaIntestazioneRiga2:*** gli altri tre sono già implementati.

Successivamente, il ciclo principale di lettura di *leggiCedolino* **(da implementare)** deve leggere ed elaborare le righe corrispondenti alle singole voci del cedolino, costruendo altrettante ***VoceCedolino*** da aggiungere al ***Cedolino*** stesso. Nel farlo deve verificare che:

i. la riga contenga o tre o quattro elementi

ii. il primo elemento sia costituito da un numero intero fra 1 e 31, seguito da uno o più spazi e dal nome (case-insensitive) del giorno della settimana corrispondente
SUGGERIMENTO*: a partire dalla data del cedolino, che contiene già mese e anno, modificare il giorno della settimana con quello dato, verificando poi, a parte, che il nome del giorno della nuova data così ottenuta sia quello atteso – sfruttando il metodo CedolinoReader.getDayOfWeekName*
OPPURE*: combinare gli elementi relativi al giorno con quelli relativi a mese e anno (forniti nell'intestazione del cedolino) per ottenere una data in formato full, di cui sia facile fare il parse*

iii. il secondo e il terzo elemento siano orari correttamente formattati secondo la cultura italiana (NB: provvede ***VoceCedolino*** a verificare che l'orario di entrata sia antecedente all'orario di uscita)

iv. l'eventuale quarto elemento, se presente, contenga una delle stringhe (case-insensitive) "PAUSA PRANZO", "RIPOSO COMPENSATIVO" o "RIPOSO COMPENSATIVO A ORE", da trasformare nella corrispondente ***Causale.***

Parte 3 - Grafica — Punti: 5

Package: bancaore.controller — *(punti 0)*

Il **Controller** (fornito) è organizzato secondo il diagramma UML nella figura seguente: esso lavora su una ***BancaOre*** ricevuta all'atto della costruzione.

SEMANTICA:

- il costruttore riceve la ***BancaOre*** su cui opererà e ne ricava, memorizzandolo internamente per comodità, il corrispondente ***Cedolino***; calcola poi, tramite il metodo ausiliario *calcolaOreLavorate*, il totale delle ore lavorate (ossia, non di riposi o pause pranzo) risultanti dalle voci della banca per tutti i giorni del mese.
- una nutrita serie di metodi accessor, dall'ovvia denominazione, permette di ottenere sotto forma di stringa già opportunamente formattata nelma forma hh:mm i totali delle ore previste nel mese, di quelle lavorate, di quelle relative ai diversi tipi di riposi utilizzati, nonché la loro somma (ore caricate totali); altri metodi consentono altresì di ottenere i saldi iniziale e

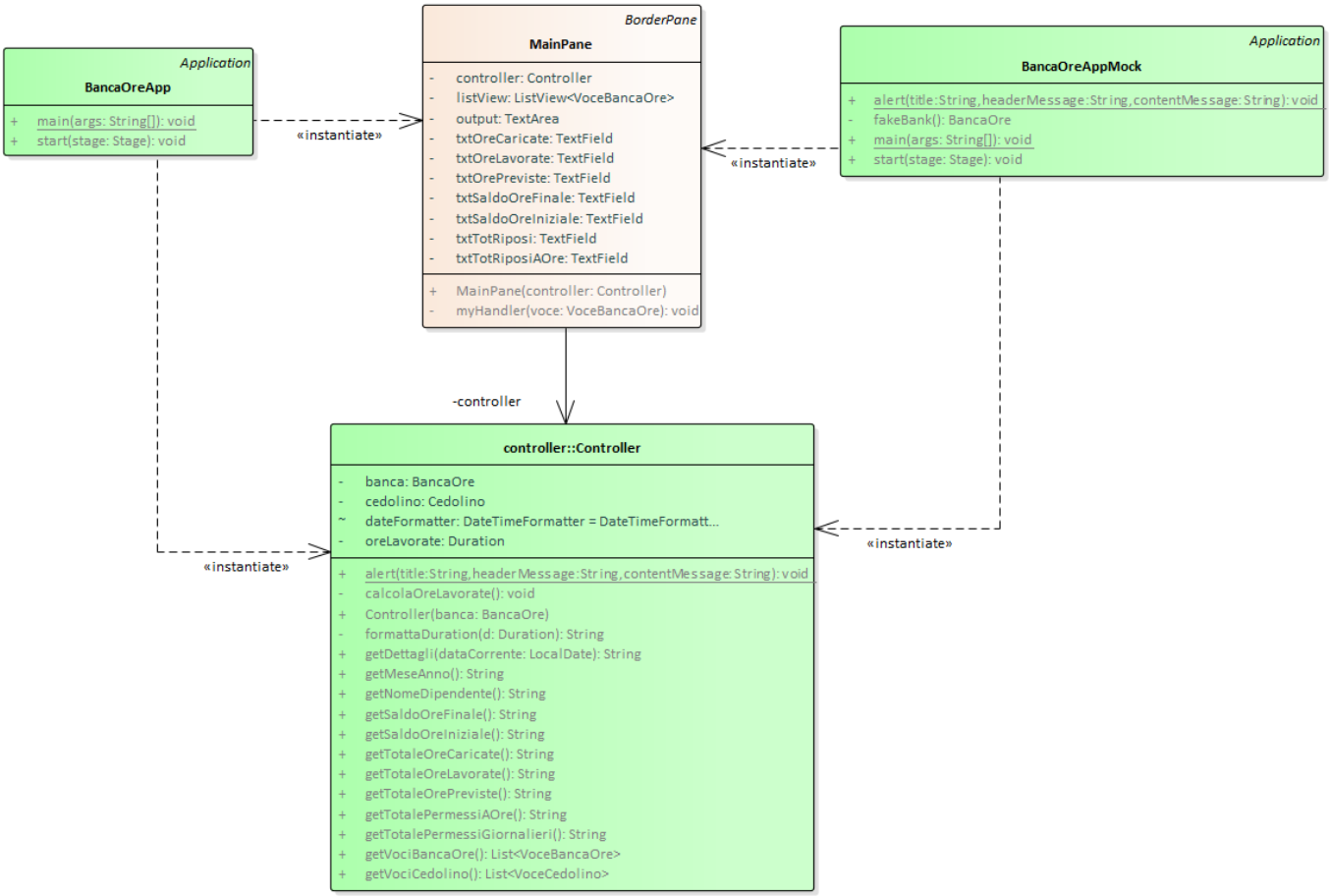
finale delle ore disponibili sul conto. Il metodo *getDettagli* restituisce una stringa, opportunamente formattata, contenente i dettagli delle voci del cedolino corrispondenti a una specifica data

- la classe contiene anche il **metodo statico ausiliario** *alert*, utile per mostrare avvisi all'utente.

Package: bancaore.ui **[TEMPO STIMATO: 10-20 minuti]** ***(punti 5)***

La classe ***BancaOreApp*** (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 ***BancaOreAppMock***.

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



L'interfaccia grafica si presenta come segue:

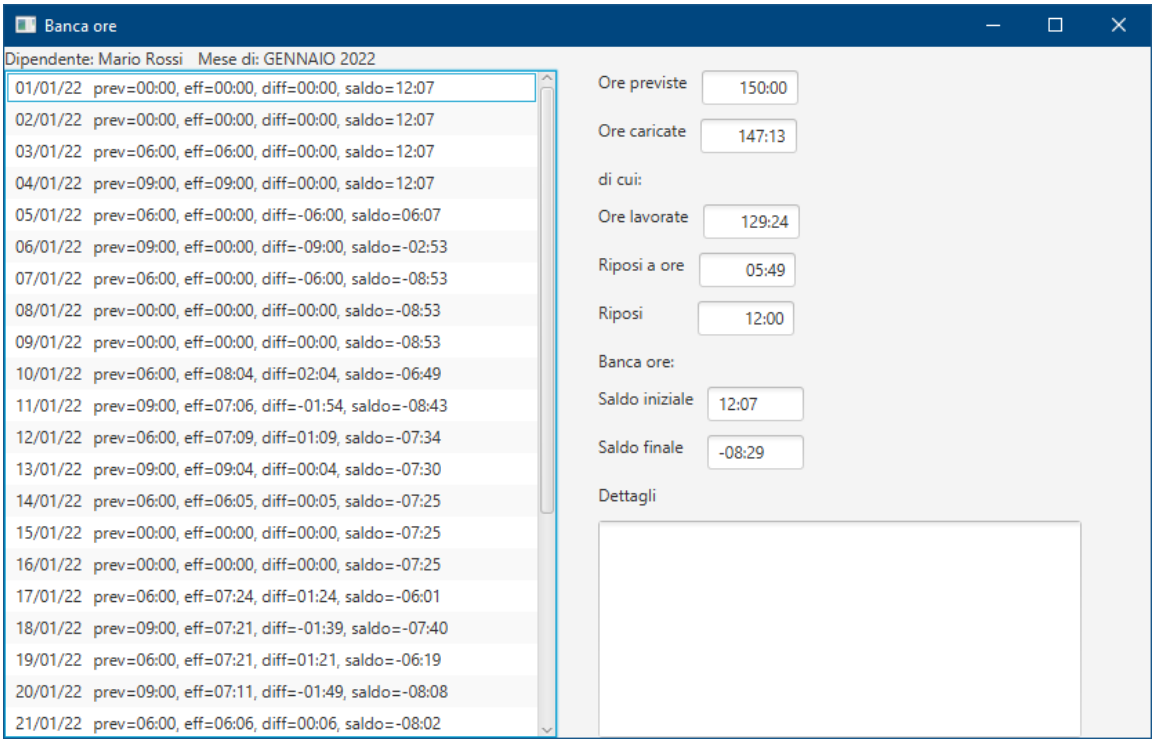


Fig. 1: la situazione iniziale della GUI

- in alto, due etichette riportano rispettivamente il nome del dipendente e, in maiuscolo, mese e anno a cui l'estratto conto si riferisce
- a sinistra, una **ListView** mostra le voci della banca ore, ordinate per data crescente

- a destra, una serie di etichette e campi di testo (non editabili) mostrano i dati rilevanti (ore previste nel mese, ore caricate con relativo dettaglio, saldi iniziale e finale delle ore disponibili sul conto. Subito sotto, un'area di testo funge da dispositivo di uscita per mostrare, a richiesta dell'utente, i dettagli della voce selezionata.

L'utente può interagire unicamente selezionando le varie righe sulla sinistra: in risposta, nell'area di testo vengono mostrate le singole voci del cedolino che "scorporano" la voce della banca ore selezionata.

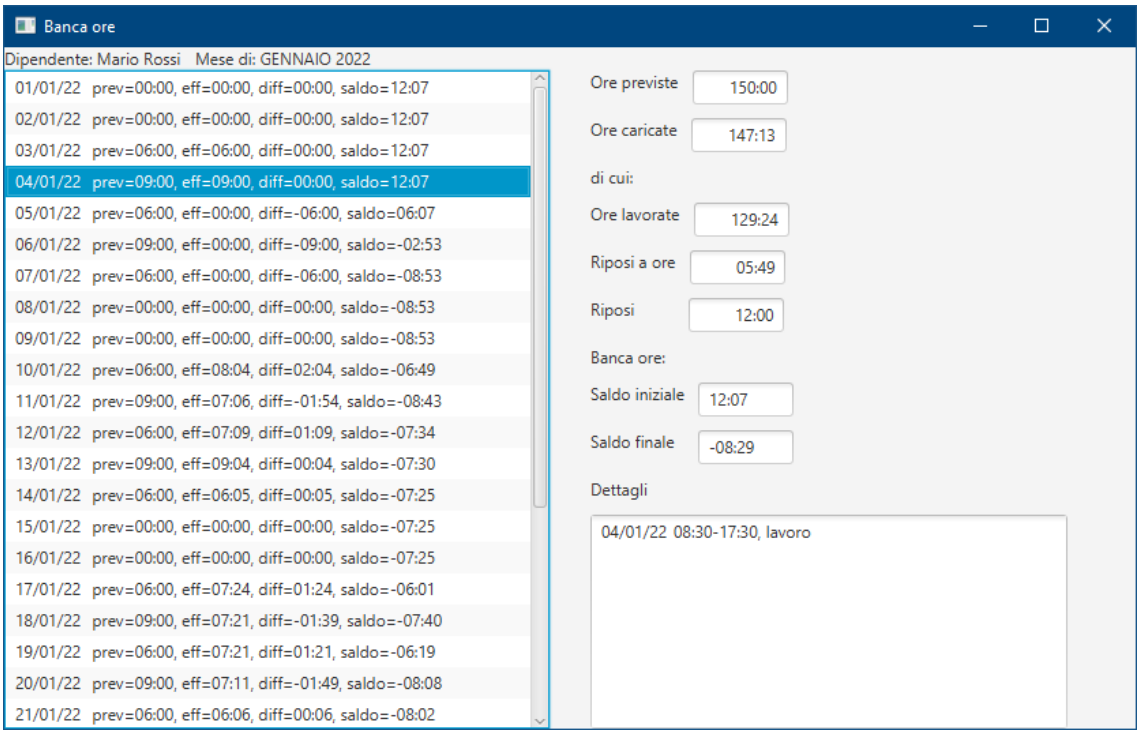


Fig. 2: i dettagli relativi a una giornata con una sola voce di cedolino

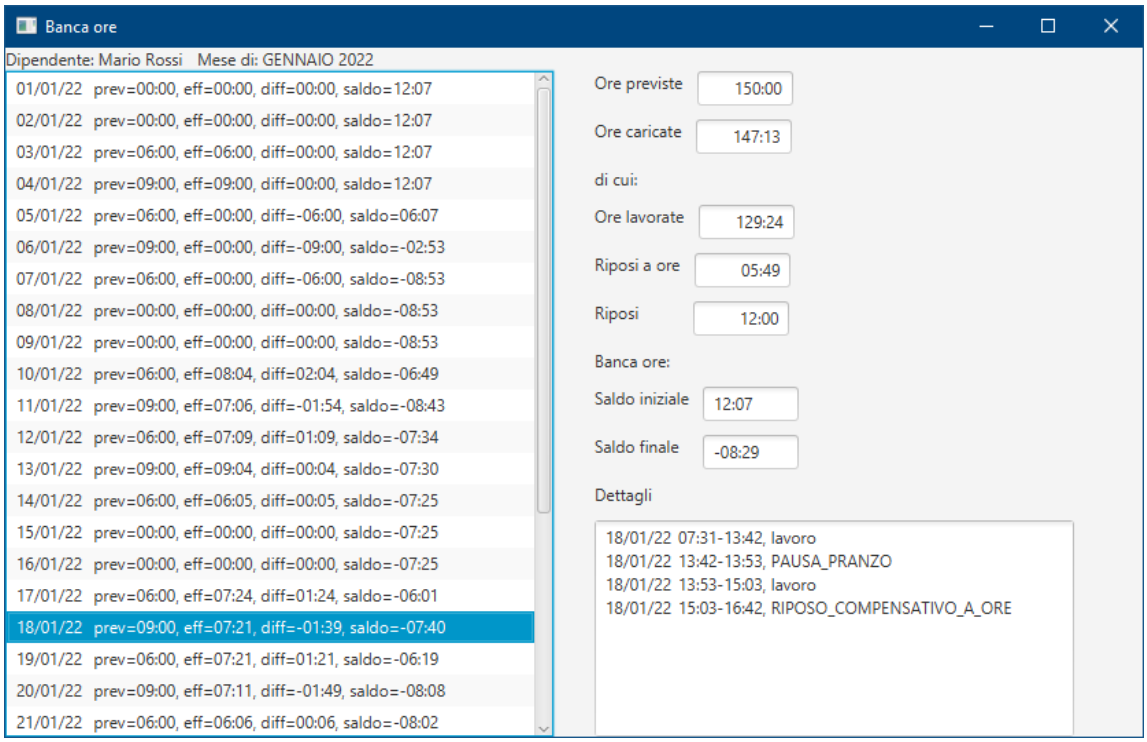


Fig. 3: i dettagli relativi a una giornata con più voci di cedolino

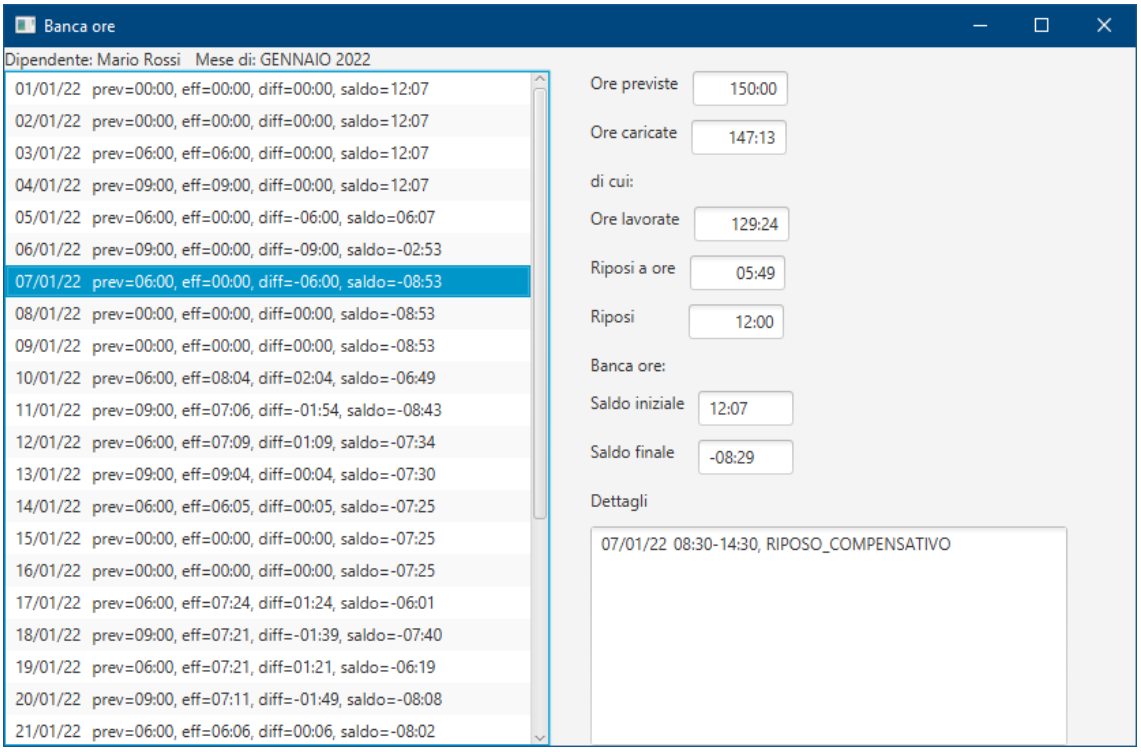


Fig. 4: i dettagli relativi a una giornata con una sola voce di riposo compensativo

Se il cedolino non contiene voci per la data selezionata, viene mostrata opportuna indicazione:

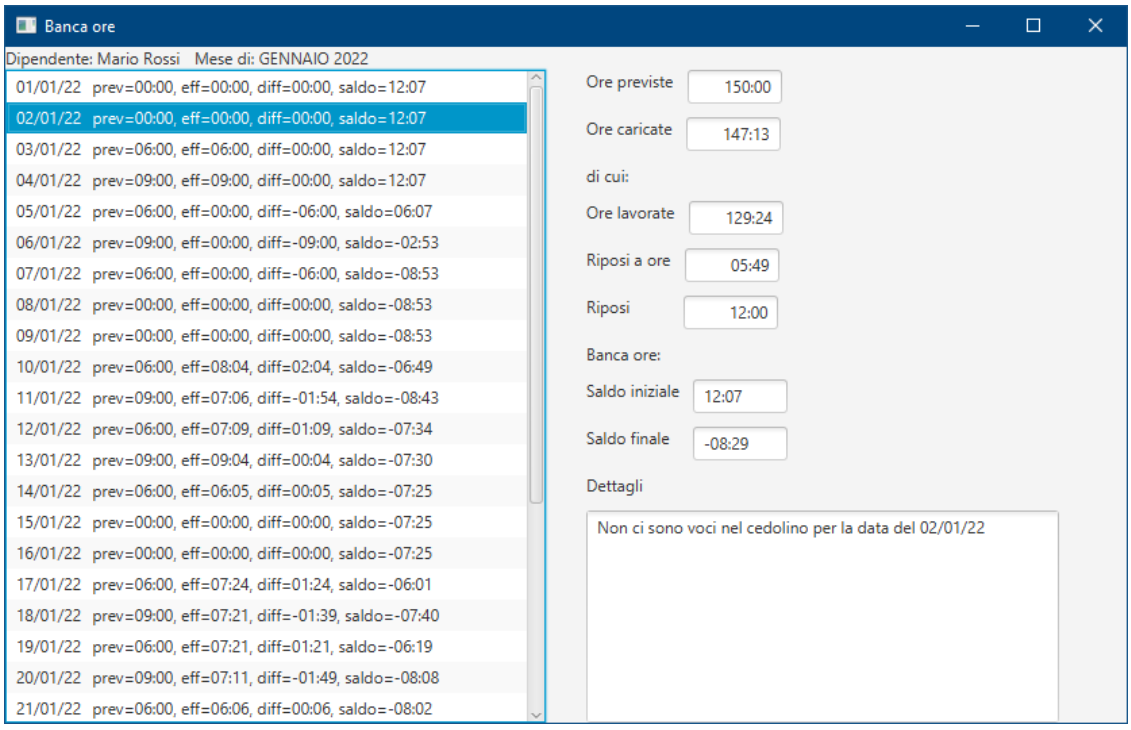


Fig. 5: i dettagli relativi a una giornata per la quale il cedolino non conteneva voci

Il MainPane è fornito *quasi totalmente realizzato*: è presente tutta la parte strutturale, mentre sono da completare la configurazione della listView e la gestione dell'evento.

In particolare, la parte da completare riguarda:

1) il popolamento della ***ListView***
2) l'aggancio alla ***ListView*** dell'opportuno listener
3) la logica di gestione dell'evento

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?