ESAME DI FONDAMENTI DI INFORMATICA T-2 del 23/07/2018

Proff. E. Denti – R. Calegari – G. Zannoni Tempo: 4 ore

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)

NB: l'archivio ZIP da consegnare deve contenere l'intero progetto Eclipse

La società *Park-O-Mat* deve predisporre il software di un innovativo tipo di parcometro *smart*.

DESCRIZIONE DEL DOMINIO DEL PROBLEMA

Il parcometro regola la sosta nelle zone a pagamento ("strisce blu"), secondo una tariffazione a fasce orarie.

Una ***fascia oraria*** esprime un intervallo di tempo *all'interno di una singola giornata* ed è caratterizzata da:

- il giorno della settimana a cui si riferisce (es. Martedì);
- l' orario iniziale e finale (es. dalle 8 alle 13, intendendosi l'intervallo chiuso a sinistra e aperto a destra);
- l'essere *a pagamento* o *gratuita* (queste ultime corrispondono tipicamente a orari notturni o festivi).

Una ***tariffa*** è definita da un insieme di fasce orarie che definiscono quando la sosta si paga, ed è caratterizzata da:

- un nome univoco;
- il costo orario (in €/ora);
- un insieme di ***fasce orarie*** in cui il pagamento è attivo (es. Lunedì 8-20, Martedì 8-13 e 15-20, etc.);
- un periodo di *franchigia* iniziale (in minuti), in cui la sosta non viene comunque fatta pagare (può essere zero);
- un periodo *minimo* di sosta tariffata (in minuti), che determina il pagamento minimo effettuabile (può essere zero).

Una tariffa non è quindi tenuta a specificare esplicitamente anche le fasce gratuite, pur potendolo fare.

Una ***tariffa valida*** è invece una tariffa che copre tutti i giorni della settimana, 24h/24, senza "buchi", specificando quindi non solo tutte le fasce orarie a pagamento, ma *anche tutte le fasce orarie gratuite*.

Per ***ticket di sosta*** si intende un pagamento riferito a una sosta che inizia in un dato momento (giorno e ora) e termina in un altro momento (giorno e ora): ad essa corrisponde un ***costo totale*** calcolato come segue **(don't panic! ☺):**

1. si pospone l'orario effettivo di inizio sosta di una quantità pari al periodo di franchigia: ad esempio, una sosta che inizi alle 7.30 in una zona dove la franchigia è di 1h si considera iniziare alle 8.30, ai fini del calcolo del costo.
2. si calcola la durata della sosta e, se risulta inferiore al periodo minimo, si tariffa comunque il minimo: ad esempio, se la sosta effettiva è di 45 minuti ma il minimo tariffabile è 1h, viene tariffata comunque 1h.
3. si sommano i costi relativi alle fasce orarie e giorni coperti dalla sosta, considerando naturalmente solo i periodi a pagamento ed escludendo i periodi gratuiti: ad esempio, una sosta che inizi effettivamente alle 16 di un giorno e termini alle 12 di un altro, in una zona in cui si paghi dalle 8 alle 20, dovrà tariffare il periodo 16-20 del primo giorno, escludere il periodo 20-24 dello stesso giorno, far pagare il periodo 8-20 di tutti i giorni intermedi escludendo i corrispondenti periodi 0-8 e 20-24, e far pagare infine il periodo 8-12 dell'ultimo giorno, escludendo il periodo 0-8.

ESEMPI DI FASCE ORARIE

lunedì, 7-20, a pagamento — *si intende quindi che si paga dalle 7:00 alle 19:59 comprese*
giovedì, 0-7, sosta gratuita — *si intende quindi che la sosta è gratuita dalle 0:00 alle 6:59 comprese*

ESEMPI DI TARIFFE *(elencano solo le fasce a pagamento)* [NB: questo non è il formato del file, è solo un esempio]

Tariffa H1 — lun-dom 7-20 € 0.50/h, franchigia 0, minimo 60 minuti (7 fasce settimanali a pagamento)

Tariffa C — lun-mar, gio-sab 8-13 e 15-20, € 0.50/h, franchigia 60 minuti, minimo 60 minuti
mer 8-13 € 0.50/h, franchigia 60 minuti, minimo 60 minuti (13 fasce settimanali a pagamento)

ESEMPIO DI TARIFFA VALIDA *(elenca anche le fasce gratuite così da coprire tutte la settimana 24h/24, 7gg/7)*

Tariffa H1 valida — lun-dom 0-7, sosta gratuita (7 fasce settimanali gratuite)
lun-dom 7-20 € 0.50/h, franchigia 0, minimo 60 minuti (7 fasce settimanali a pagamento)
lun-dom 20-24, sosta gratuita (7 fasce settimanali gratuite)

ESEMPI DI TICKET SOSTA

sosta in zona H1 martedì dalle 7.20 alle 8.45 → durata 1h25 (ok, supera il minimo) → € 0.50 x (1+25/60) = € 0.71

sosta in zona H1 giovedì dalle 7.20 alle 8.05 → durata 0h45 → minimo 1h → € 0.50

sosta in zona C il giovedì dalle 7.30 alle 15 → (franchigia 1h = inizio effettivo 8.30) → durata 4h30 (8:30-13) → € 2.25

sosta in zona C da lunedì 19.30 a martedì 16.00 → (franchigia 1h = inizio effettivo 20.30, paga dalle 8 di martedì) → tariffate 6h (dalle 8 alle 13 e dalle 15 alle 16) → € 3.00

sosta in zona C da martedì 19.30 a mercoledì 16.00 → (franchigia 1h = inizio effettivo 20.30, paga dalle 8 di mercoledì) → tariffate 5h (dalle 8 alle 13: mercoledì pomeriggio è gratis) → € 2.50

Il file di testo Tariffe.txt specifica le tariffe, nel formato dettagliato più oltre.

IMPORTANTE: per evitare problemi con Java9 in laboratorio, si suggerisce di **far partire Eclipse attraverso lo script ausiliario StartEclipse.bat fornito nello start kit.**

INFORMAZIONE UTILE: a causa di modifiche nel provider delle specifiche **Locale** intervenute fra Java 8 e Java 9, ***il formattatore di valute per Locale.ITALY opera diversamente in Java 9 rispetto a Java 8.*** Ciò impatta anche il funzionamento dei metodi *parse* utilizzati per la conversione stringa(prezzo)/numero.

Per ottenere il comportamento classico è necessario aggiungere alle *Run Configurations* →*Arguments* la specifica:

```
-Djava.locale.providers=COMPAT
```

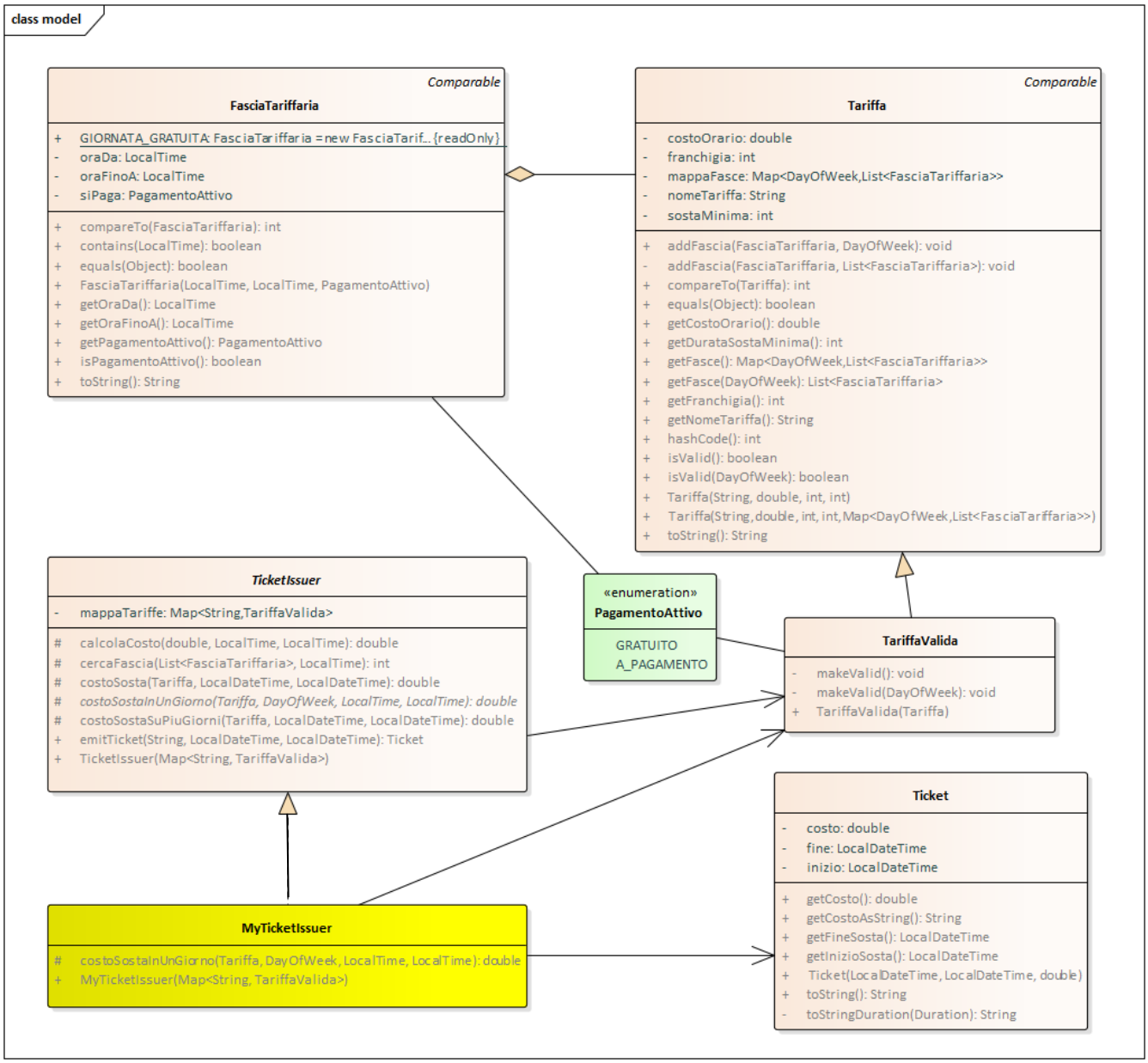
Essa va aggiunta sia alla *run configuration* di JUnit, sia a quella dell'Application, sia a quella dell'ApplicationMock.

NB: le run configuration vengono generate da Eclipse ***dopo il primo run*** di JUnit o dell'application stessa.

Parte 1 (punti: 20)

Dati (namespace parcomat.model) *(punti: 10)*

Il modello dei dati deve essere organizzato secondo il diagramma UML di seguito riportato:



SEMANTICA:

a) l'enumerativo di utilità ***ItDayOfWeek*** (fornito, ma non mostrato nell'UML sopra) definisce i giorni della settimana in italiano e **fornisce due comodi metodi di conversione** rispettivamente da/verso ***java.time.DayOfWeek***;

b) la classe ***FasciaTariffaria*** *(fornita)* rappresenta un intervallo di tempo (coppia di ***java.time.LocalTime***) in un dato giorno della settimana, a pagamento o gratuita (caratteristica espressa dall'enumerativo ***PagamentoAttivo***); è ***Comparable*** sull'orario di inizio fascia, **così da permettere un facile ordinamento sequenziale**; il metodo ***equals*** è ovviamente coerente. Oltre agli accessor, il metodo ***contains*** verifica se un orario appartenga alla fascia stessa.

c) la classe ***Tariffa*** (fornita) è caratterizzata dal suo nome univoco, dal costo orario (€/ora), dalla durata della franchigia, dal periodo minimo tariffato e una o più ***FasciaTariffaria***; per comodità di configurazione, al costruttore principale, che riceve tutti questi argomenti, si affianca un costruttore ausiliario che non riceve le fasce tariffarie, le quali possono essere aggiunte successivamente tramite il metodo ***addFascia*** che aggiunge una data fascia al giorno della settimana indicato, rispettandone l'ordine temporale. Il metodo ***isValid***, in due versioni, permette di verificare se la tariffa sia *valida*, ossia copra interamente la giornata, rispettivamente in un dato giorno o nell'intera settimana [NB: di norma NON lo sarà, perché le tariffe solitamente NON specificano le fasce gratuite]

d) la classe ***TariffaValida*** (fornita) incapsula una ***Tariffa*** rendendola automaticamente valida, ossia *aggiungendo in automatico le fasce orarie gratuite* necessarie a completare la copertura dell'intera settimana. Per una tariffa valida, quindi, il metodo ***isValid*** ha sempre esito positivo.

e) la classe ***Ticket*** (fornita) rappresenta un ticket di sosta con le sue caratteristiche;

f) la classe astratta ***TicketIssuer*** (fornita) emette il ticket per un dato periodo di sosta, tramite il metodo pubblico ***emitTicket***; a tal fine si appoggia a un nutrito numero di metodi ausiliari (*protected*), descritti sotto.
Fra questi, il metodo astratto ***costoSostaInUnGiorno*** calcola l'importo della sosta in un dato giorno della settimana, dall'orario dato come primo argomento (incluso) all'orario dato come secondo argomento (escluso).

- il costruttore riceve una mappa con tutte le ***TariffaValida*** disponibili, indicizzata per nome;
- il metodo pubblico ***emitTicket*** calcola il costo della sosta da un dato momento iniziale (un ***LocalDateTime***) a un dato momento finale (un altro ***LocalDateTime***) e ne emette il relativo ticket;
- il metodo ***costoSosta*** calcola il costo della sosta dal momento iniziale (***LocalDateTime***) al momento finale (***LocalDateTime***), distinguendo il caso in cui la sosta inizi e finisca in un'unica giornata (metodo astratto ***costoSostaInUnGiorno***) da quello in cui si estenda su più giorni (metodo ***costoSostaSuPiuGiorni***);
- il metodo di utilità ***cercaFascia*** cerca in una lista ordinata di fasce orarie (primo argomento) la fascia oraria in cui ricade il ***LocalTime*** fornito come secondo argomento;
- il metodo di utilità ***calcolaCosto***, dato il costo orario, calcola il costo della sosta fra due istanti (***LocalTime***) di una stessa giornata, gestendo anche il caso particolare in cui l'istante finale sia mezzanotte.

g) la classe **MyTicketIssuer (da realizzare)** deve concretizzare ***TicketIssuer*** implementando ***costoSostaInUnGiorno*** in modo da riflettere lo specifico algoritmo di calcolo del costo descritto nel Dominio del problema (ricordando che le istanze di ***FasciaTariffaria*** di un giorno sono opportunamente ordinate all'interno della tariffa).

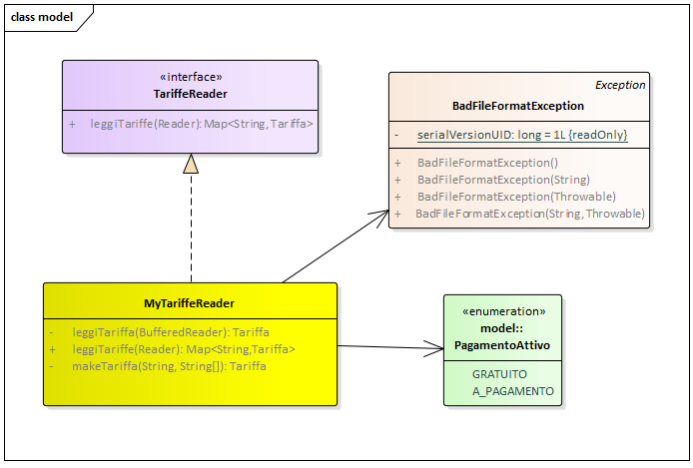
Persistenza (namespace parcomat.persistence) *(punti: 10)*

Come già anticipato, il file Tariffe.txt specifica le tariffe: ognuna occupa più righe, ciascuna delle quali descrive una possibile fascia tariffaria in un dato giorno della settimana. Il numero di righe che descrivono una tariffa è quindi variabile, dipendendo dalle fasce orarie e dai giorni della settimana coinvolti (v. esempio sotto). Più precisamente:

- la prima riga contiene la parola chiave "**`Tariffa`**" seguita dal nome della tariffa (che non contiene spazi);
- la seconda riga contiene nell'ordine costo, franchigia e minimo tariffario, così formattati:
  - il costo orario è composto dalla parola chiave "**`costo`**" seguita da un valore double;
  - la franchigia è composta dalla parola chiave "**`franchigia`**" seguita da un valore intero;
  - il minimo tariffabile è composto dalla parola chiave "**`minimo`**" seguita da un intero.
- le righe successive contengono l'elenco delle fasce orarie a pagamento, nel seguente formato:
  - il giorno della settimana, in italiano;
  - la fascia oraria, nel formato "*orainiziale*-*orafinale*", dove *orainiziale* e *orafinale* sono interi.
- La riga finale contiene la frase "**`Fine tariffa`**" .

ESEMPIO DEL FILE Tariffe.txt

```
Tariffa C
costo 0.50, franchigia 60, minimo 60
Lunedì,     8-13
Lunedì,    15-20
Martedì,  8-13
Martedì, 15-20
Mercoledì,8-13
Giovedì,  8-13
Giovedì, 15-20
Venerdì,  8-13
Venerdì, 15-20
Sabato,     8-13
Sabato,    15-20
Fine tariffa
```



```
Tariffa H1
costo 0.50, franchigia 0, minimo 60
Lunedì,     8-20
Martedì,  8-20
Mercoledì,8-20
Giovedì,  8-20
Venerdì,  8-20
Sabato,     8-20
Fine tariffa
```

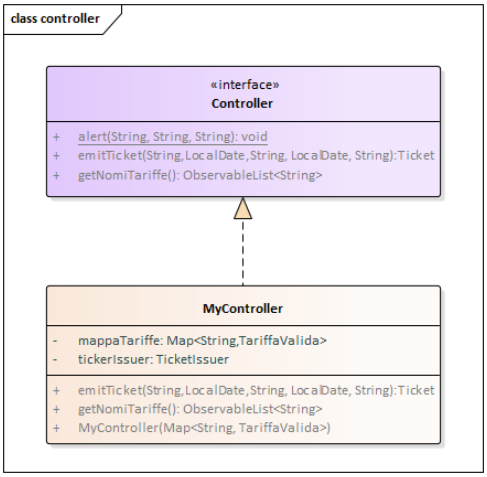
L'interfaccia ***TariffeReader*** (fornita) dichiara il metodo ***leggiTariffe*** che, dato un ***Reader***, legge le tariffe dal file e le restituisce sotto forma di mappa ***Map<String, Tariffa>*** indicizzata per nome tariffa.

La classe *MyTariffeReader* (da realizzare) implementa tale interfaccia effettuando i necessari controlli sul formato del file, lanciando ***BadFileFormatException*** (fornita) in caso di errori di formato, o propagando ***IOException*** in caso di errori di lettura con specifico messaggio d'errore.

Parte 2 (punti: 10)

Controller (namespace parcomat.ui.controller) *(punti: 5)*

L'interfaccia ***Controller*** (fornita) dichiara il metodo ***getNomiTariffe***, che restituisce la *lista osservabile* dei nomi delle tariffe disponibili, e il metodo ***emitTicket*** che emette il ticket di sosta. Quest'ultimo riceve come argomenti:



- la data di inizio sosta (un ***LocalTime***)
- l'orario di inizio sosta (una stringa nel formato ***hh:mm***)
- la data di fine sosta (un ***LocalTime***)
- la data di inizio sosta (una stringa nel formato ***hh:mm***)

È fornito inoltre il metodo statico ***alert*** per far comparire una finestra di dialogo che segnali errori: i tre argomenti rappresentano il titolo della finestra, l'header e il testo del messaggio (Figg. 5 e 6).

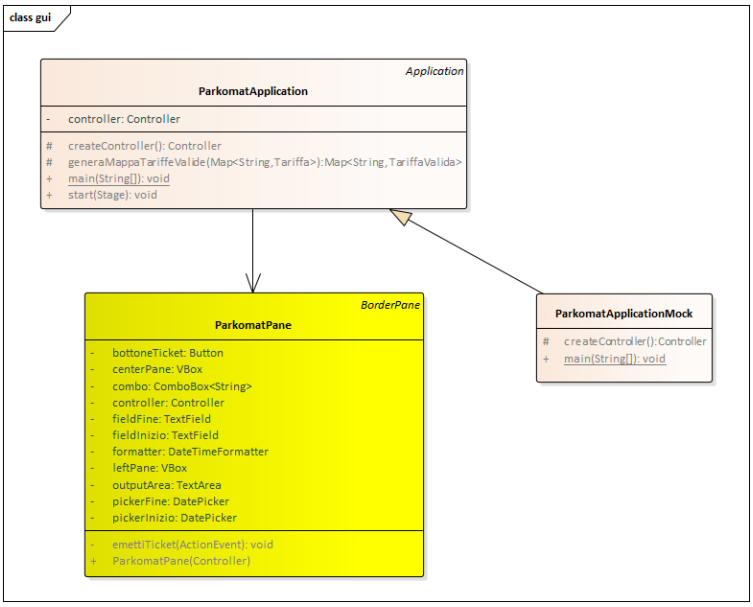
La **classe *MyController* (da realizzare)** deve implementare ***Controller***, segnalando errore nel caso in cui l'istante finale preceda quello iniziale (Fig. 6).

Interfaccia utente (namespace parcomat.ui.javafx) *(punti: 5)*

L'interfaccia utente deve essere simile (non necessariamente identica) all'esempio mostrato nelle figure seguenti.

La classe *ParkomatPane* (da realizzare), che estende ***BorderPane***, deve prevedere:

- in alto, una combo box per la scelta delle tariffe disponibili;
- a sinistra, due ***DatePicker*** e due campi di testo, per inserire data e orario di inizio e fine sosta;
- a destra, un'area di testo in cui mostrare il ticket emesso;
- sotto, un pulsante *Emetti ticket* per emettere il ticket di sosta.



In caso di problemi (mancanza di file, errori di lettura o di formato, date fine sosta precedenti l'inizio sosta) l'applicazione deve mostrare opportuni dialoghi, tramite il metodo statico ***Controller.alert*** (Figg. 5 e 6).

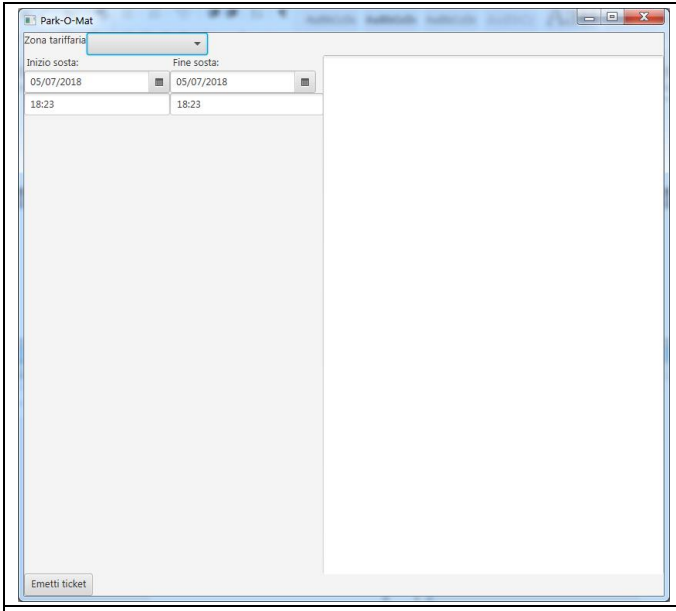


Figura 1

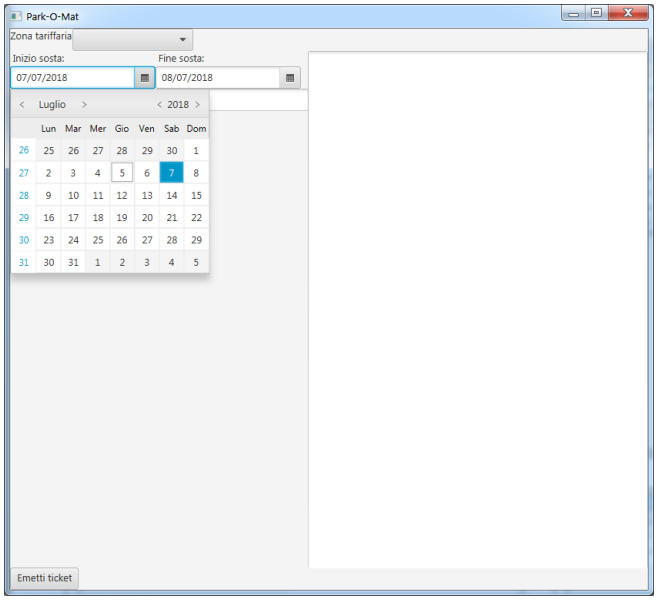


Figura 2

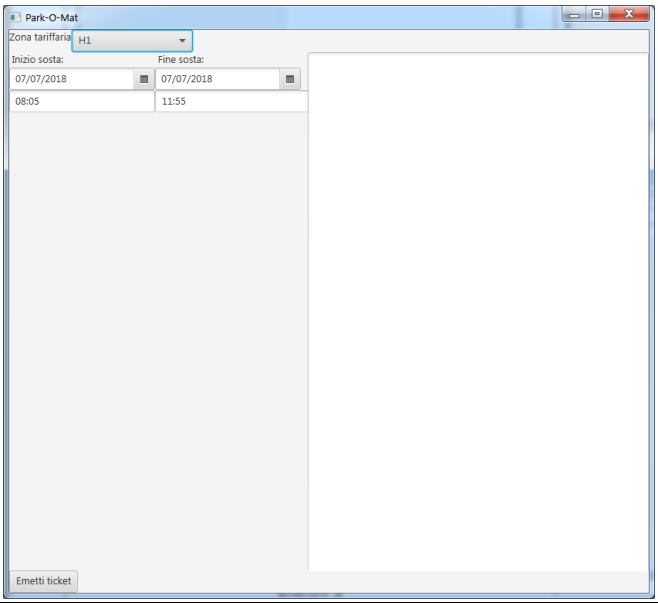


Figura 3

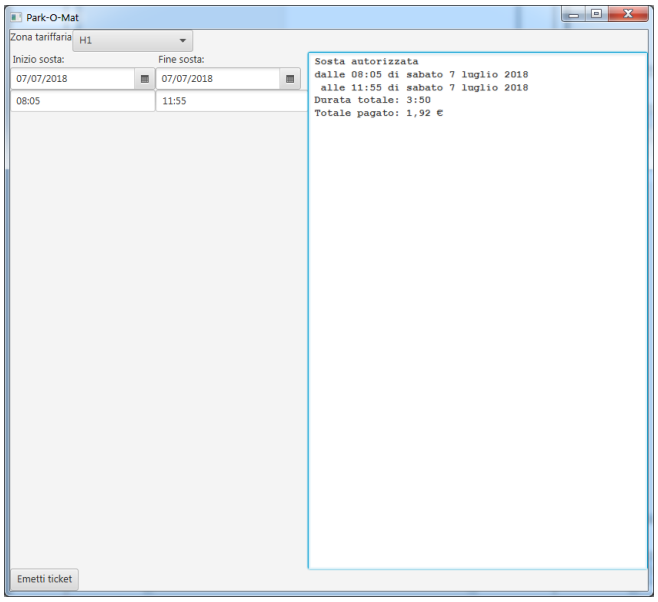


Figura 4

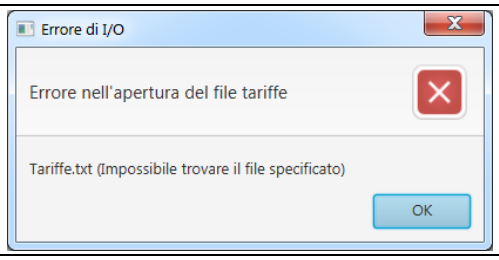


Figura 5

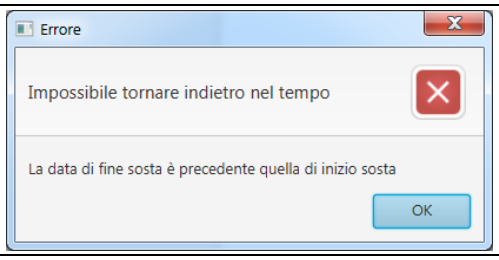


Figura 6