ESAME DI FONDAMENTI DI INFORMATICA T-2 dell'8/2/2023
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"

La valuta dei maghi, coniata dalla Gringott Bank, è composta da tre tipi di monete: **Galeoni** d'oro (*galleons* nell'originale inglese), **Falci** d'argento (*sickles* nell'originale inglese) e **Zellini** in bronzo (*knuts* nell'originale inglese). Un **Galeone** è suddiviso in **17 Falci**, ognuno dei quali è a sua volta suddiviso in **29 Zellini**.

Obiettivo dell'applicazione è progettare uno sportello automatico che **eroghi solo monete Gringott**, rispettando per ogni prelievo un massimale prestabilito per ogni utente. Per agevolare l'interazione con gli umani non-maghi, lo sportello deve altresì offrire la possibilità di esprimere gli importi nelle principali valute umane (Dollari, Euro o Sterline), il cui tasso di cambio con i Galeoni è fisso e prestabilito (1 galeone = 5 sterline = 6 dollari = 5.66 euro).

DESCRIZIONE DEL DOMINIO DEL PROBLEMA

Per effettuare un prelievo si può specificare l'ammontare desiderato o nella valuta dei maghi o in una delle valute umane supportate (Dollari, Euro o Sterline). Lo sportello rispetta le seguenti regole:

1) Solo gli utenti preventivamente autorizzati, elencati nell'apposito file, possono compiere prelievi
2) L'importo è erogabile solo se non supera il massimale per ogni singola operazione specificato nell'apposito file
3) L'importo in monete Gringott deve dare il massimo numero di galeoni possibile, *garantendo tuttavia che siano erogati sempre almeno 5 falci d'argento*.
4) Nel caso di importi specificati in Dollari, Euro o Sterline, deve essere usato il tasso di cambio fisso prestabilito.

ESEMPI DI APPLICAZIONE DELLE REGOLE 3 & 4:

- Richiesti 20 galeoni → Erogati 19 galeoni + 17 falci
- Richiesti 19 galeoni + 20 zellini → Erogati 18 galeoni + 17 falci + 20 zellini
- Richieste 500 sterline → (1 galeone = £ 5.00) [=100 galeoni teorici] → Erogati 99 galeoni +17 falci
- Richiesti 500 euro → (1 galeone = € 5.66) Erogati 88 galeoni + 5 falci + 22 zellini (importo effettivo: 499,67 €)
- Richiesti 500 dollari → (1 galeone = $ 6.00) Erogati 83 galeoni + 5 falci + 19 zellini (importo effettivo: 499,67 €)

Trattandosi di una valuta di maghi, si presuppone che la disponibilità di monete sia illimitata.

Il file di testo ceilings.txt contiene i massimali permessi per ogni utente per ogni operazione di prelievo.

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

PARTE 1 – Modello dei dati: Punti 13 [TEMPO STIMATO: 45-60 minuti]

PARTE 2 – Persistenza: Punti 11 [TEMPO STIMATO: 60-80 minuti]

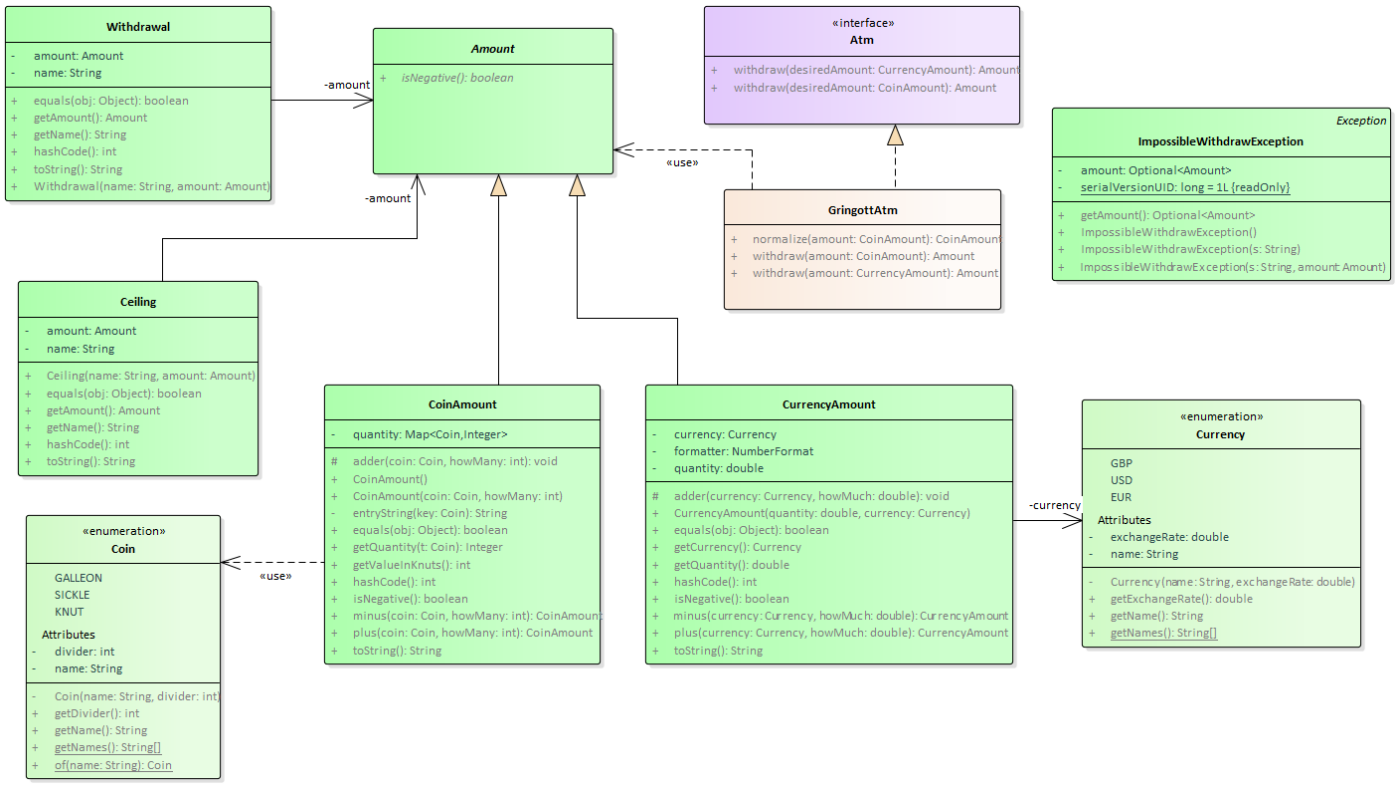
PARTE 3 – Grafica: Punti 6 [TEMPO STIMATO: 30-40 minuti]

JAVAFX – Parametri run configuration nei LAB

```
--module-path "C:\applicativi\moduli\javafx-sdk-17.0.2\lib"
--add-modules javafx.controls
```

Parte 1 – Modello dei dati (punti: 13)

Package: gringott.model **[TEMPO STIMATO: 45-60 minuti]**



SEMANTICA:

a) L'enumerativo ***Coin*** (fornito) elenca le monete Gringott: ognuna incorpora sia il proprio nome, sia il divisore (ossia di quanti sottomultipli è costituita). Entrambi queste proprietà sono recuperabili con gli appositi accessor. In aggiunta, per comodità, il metodo statico *getNames* restituisce l'elenco dei nomi delle monete sotto forma di array di stringhe, mentre il metodo-factory statico *of* restituisce l'opportuna costante enumerativa a partire dal suo nome (case-insensitive).

b) L'enumerativo ***Currency*** (fornito) elenca le valute umane riconosciute: ognuna incorpora sia il proprio nome, sia il tasso di cambio contro 1 galeone (quindi la sterlina, GBP, incorpora il fattore 5.0 perché 1 galeone vale 5 sterline, mentre l'euro, EUR, incorpora 5.66 perché 1 galeone vale appunto 5.66 euro; e così via). Entrambe queste proprietà sono recuperabili con gli appositi accessor. In aggiunta, per comodità, il metodo statico *getNames* restituisce l'elenco dei nomi delle valute sotto forma di array di stringhe.

c) La classe astratta ***Amount*** (fornita) rappresenta il concetto generale di un certo "ammontare" di denaro, indipendentemente dal modo con cui viene espresso. Dichiara solo il metodo astratto *isNegative*, vero se l'importo rappresentato è negativo.

d) Le due classi concrete ***CoinAmount*** e ***CurrencyAmount*** (fornite) specializzano ***Amount*** nei due casi in cui l'importo sia espresso rispettivamente in monete Gringott o in valute umane.

- Nel caso di ***CoinAmount***, il costruttore crea un ammontare zero, a cui possono poi essere aggiunte o tolte unità di singole monete tramite i due metodi *plus* e *minus*: essi rispettano il pattern cascading e possono quindi essere invocati in sequenza per costruire importi complessi (v. esempi nel codice).
  Il metodo *getQuantity* restituisce la quantità di una certa moneta (galeoni, falci o zellini) presente nell'***Amount***, mentre il metodo *getValueInKnuts* restituisce l'equivalente in zellini (il sottomultiplo più piccolo, dunque un valore sempre intero) dell'ammontare stesso. Un'apposita *toString* restituisce una stringa opportuna che descrive compiutamente l'importo.

- Nel caso di ***CurrencyAmount***, invece, il costruttore crea direttamente l'ammontare richiesto nella valuta (***Currency***) desiderata: i due metodi *plus* e *minus*, che operano come sopra, in questo caso *non* sono indispensabili e sono inclusi a soli fini di test ed espansioni future.
  Il metodo *getQuantity* restituisce la quantità di denaro presente nell'***Amount***, mentre il metodo *getCurrency* restituisce la valuta in cui tale ammontare è espresso. Anche qui un'apposita *toString* restituisce una stringa opportuna che descrive compiutamente l'importo.

e) La classe ***Ceiling*** (fornita) rappresenta il massimale per singola operazione concesso a un dato utente: come tale, incorpora il nome utente (una stringa) e il corrispondente ***Amount***.

f) L'interfaccia ***Atm*** (fornita) dichiara i due metodi *withDraw* che effettuano un prelievo espresso rispettivamente in ***CoinAmount*** e ***CurrencyAmount.***

g) La classe ***GringottAtm* (da completare)** implementa ***Atm*** applicando le regole 3 e 4 del Dominio del Problema, erogando per quanto possibile il massimo numero di galeoni d'oro ma garantendo altresì che siano sempre presenti almeno 5 falci d'argento e applicando, nel caso di valute umane, gli opportuni tassi di cambio.
   Se l'ammontare richiesto è negativo dev'essere lanciata apposita ***ImpossibleWithdrawException*** (fornita).

h) La classe ***Withdrawal*** (fornita) è un puro contenitore di dati che esprime il prelievo di un certo ***Amount*** da parte di un dato utente. Entrambe le proprietà sono recuperabili tramite appositi accessor.

Parte 2 – Persistenza (punti: 11)

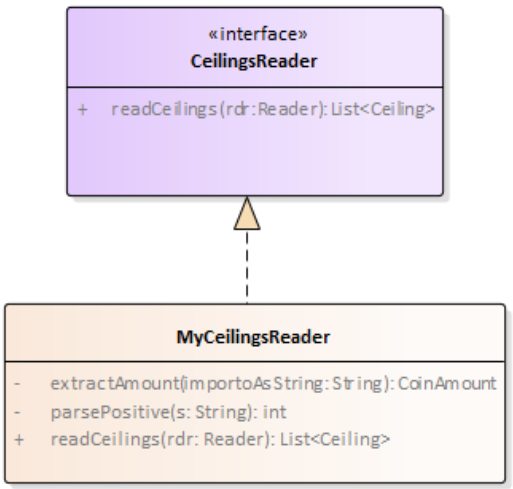
Package: gringott.persistence **[TEMPO STIMATO: 60-80 minuti]**

Il file di testo Ceilings.txt contiene i massimali permessi a ogni utente per ogni operazione. Ogni riga descrive un singolo utente: non sono previste omonimie, quindi si può assumere senz'altro che il nome dell'utente sia univoco.

Ogni riga contiene, separati da tabulazioni, il nome dell'utente e il massimale ad esso associato, espresso sempre e comunque in monete Gringott.
A sua volta, tale massimale è composto da un elenco di 1-3 parti (galeoni, falci, zellini), separate da virgole, che possono anche non essere tutte presenti: ogni parte a sua volta è composta da un numero intero seguito, dopo uno o più spazi, dal nome della moneta.

ESEMPIO

```
Harry       20 galeoni
Hermione    19 galeoni, 20 zellini
Enrico      200 galeoni
Ambra       100 galeoni, 20 zellini
Roberta     100 galeoni, 28 zellini
Ron         12 galeoni, 10 falci, 6 zellini
```



SEMANTICA:

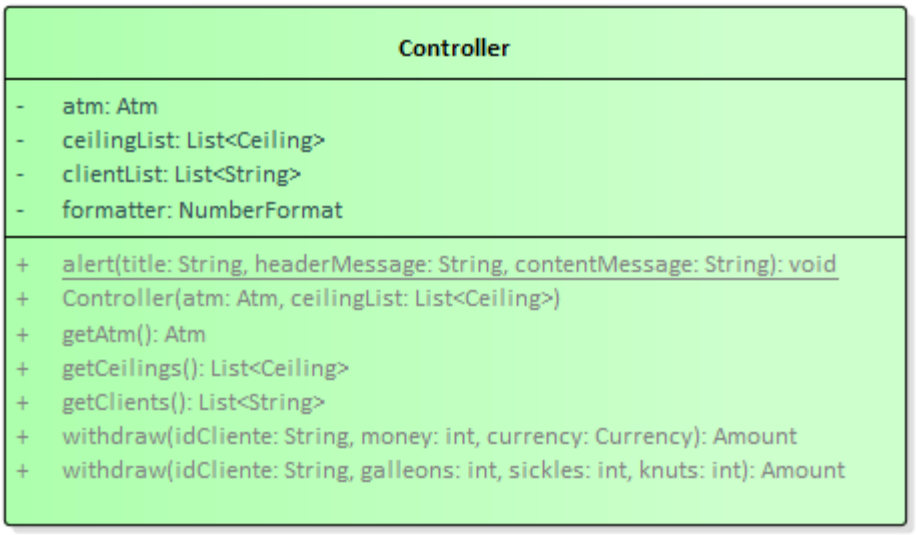
a) L'interfaccia ***CeilingsReader*** (fornita) dichiara il metodo *readCeilings* che carica da un appsito Reader (già aperto) i dati necessari, restituendo una lista di ***Ceiling***. Il metodo lancia:
   - l'opportuna ***BadFileFormatException*** con messaggio d'errore appropriato in caso di problemi nel formato del file (mancanza di elementi, mancanza di parti numeriche quando previste, numeri negativi, etc.)
   - una ***IOException*** in caso di altri problemi di I/O.

b) La classe ***MyCeilingsReader*** (**da realizzare**) implementa ***CeilingsReader*** secondo le specifiche sopra descritte.

Parte 3 (punti: 6)

Package: gringott.controller *(punti 0)*

Il Controller è organizzato secondo il diagramma UML seguente:



SEMANTICA:

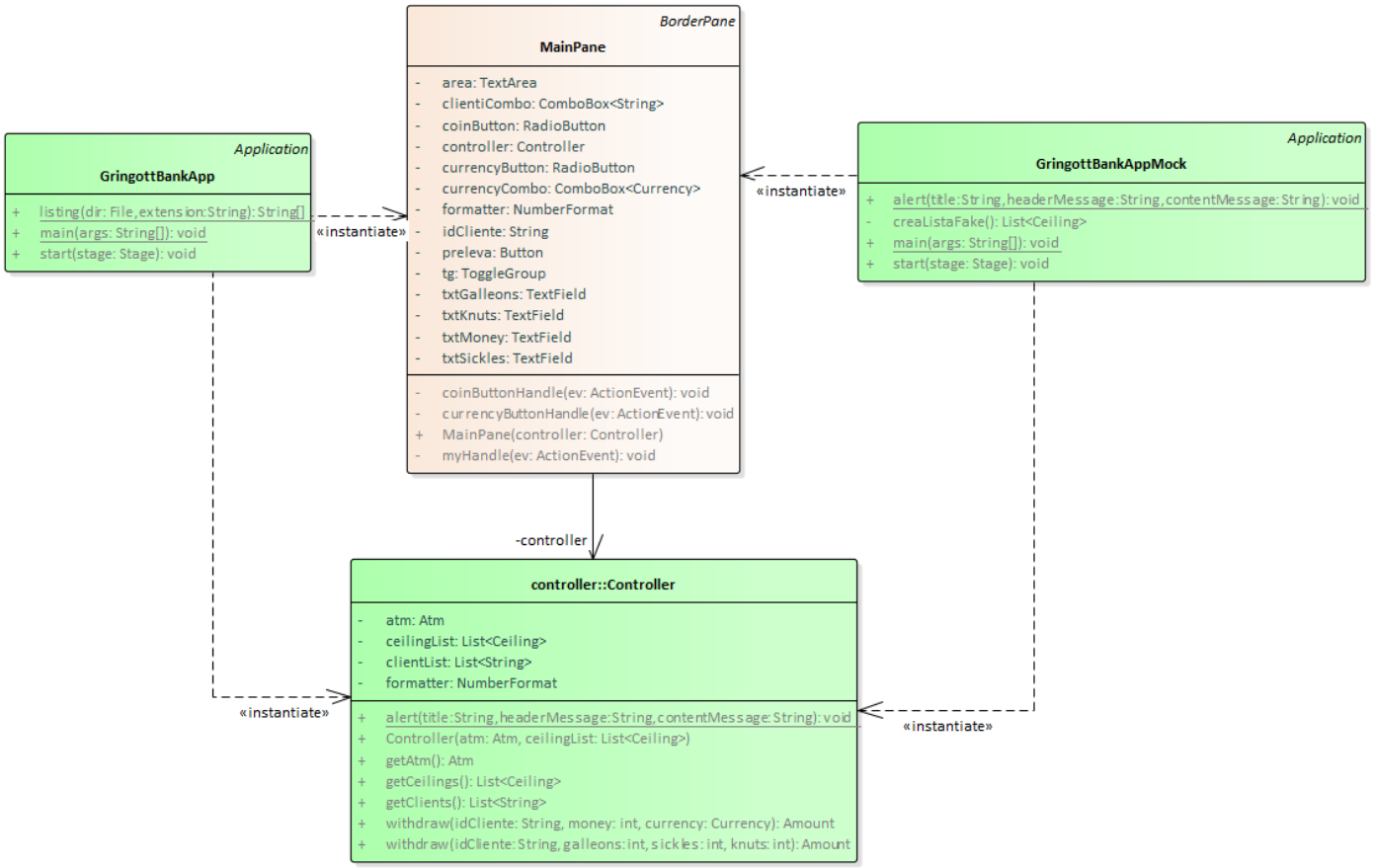
La classe ***Controller*** (fornita) incorpora in fase di costruzione di un ***Atm*** e una lista di massimali (***Ceiling***), effettuando quindi le verifiche previste dalle regole 1 & 2 del Dominio del Problema. Oltre ai due ovvi accessor che recuperano gli argomenti passati al costruttore, rende disponibili i seguenti metodi:

- *getClients*, che restituisce la lista dei clienti (stringhe) costruita a partire da quella dei massimali;
- *withdraw* (in due varianti, con diverse liste di argomenti) che consentono alla user interface di richiedere un prelievo per un dato utente e un certo ammontare, espresso o come monete Gringott (quindi tramite terna di interi per galeoni, falci e zellini) o come valuta umana (quindi tramite importo e ***Currency***). In entrambi i casi il risultato è un ***Optional<Amount>***, vuoto nel caso il prelievo risulti impossibile, o istanziato all'opportuno ***CoinAmount*** o ***CurrencyAmount*** se il prelievo è possibile. Lancia **ImpossibleWithdrawException** in caso di prelievo non autorizzato: nel caso il motivo sia il superamento del massimale, l'eccezione incorpora anche un campo dati di tipo ***Amount*** che esprime l'importo che sarebbe stato da erogare.
- il metodo statico *alert*, utilizzabile anche dal MainPane, consente di far comparire all'utente una finestra di dialogo con opportuno messaggio d'errore.

Package: gringott.ui **[TEMPO STIMATO: 30-40 minuti]** ***(punti 6)***

La classe ***GringottBankApp*** (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 ***GringottBankAppMock***.

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



L'interfaccia grafica si presenta come segue:

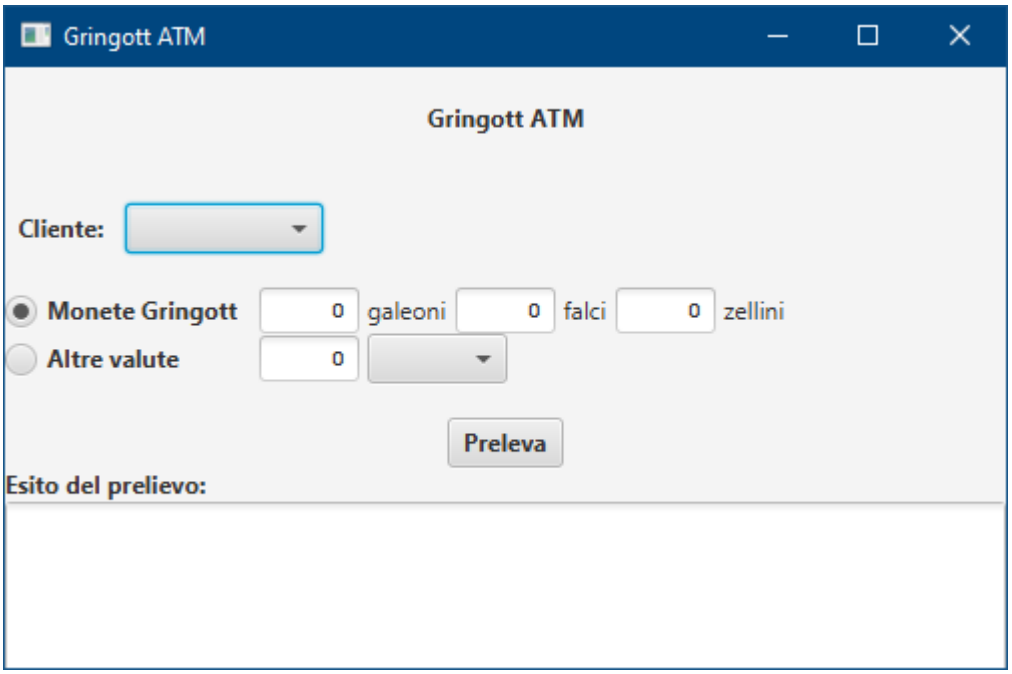
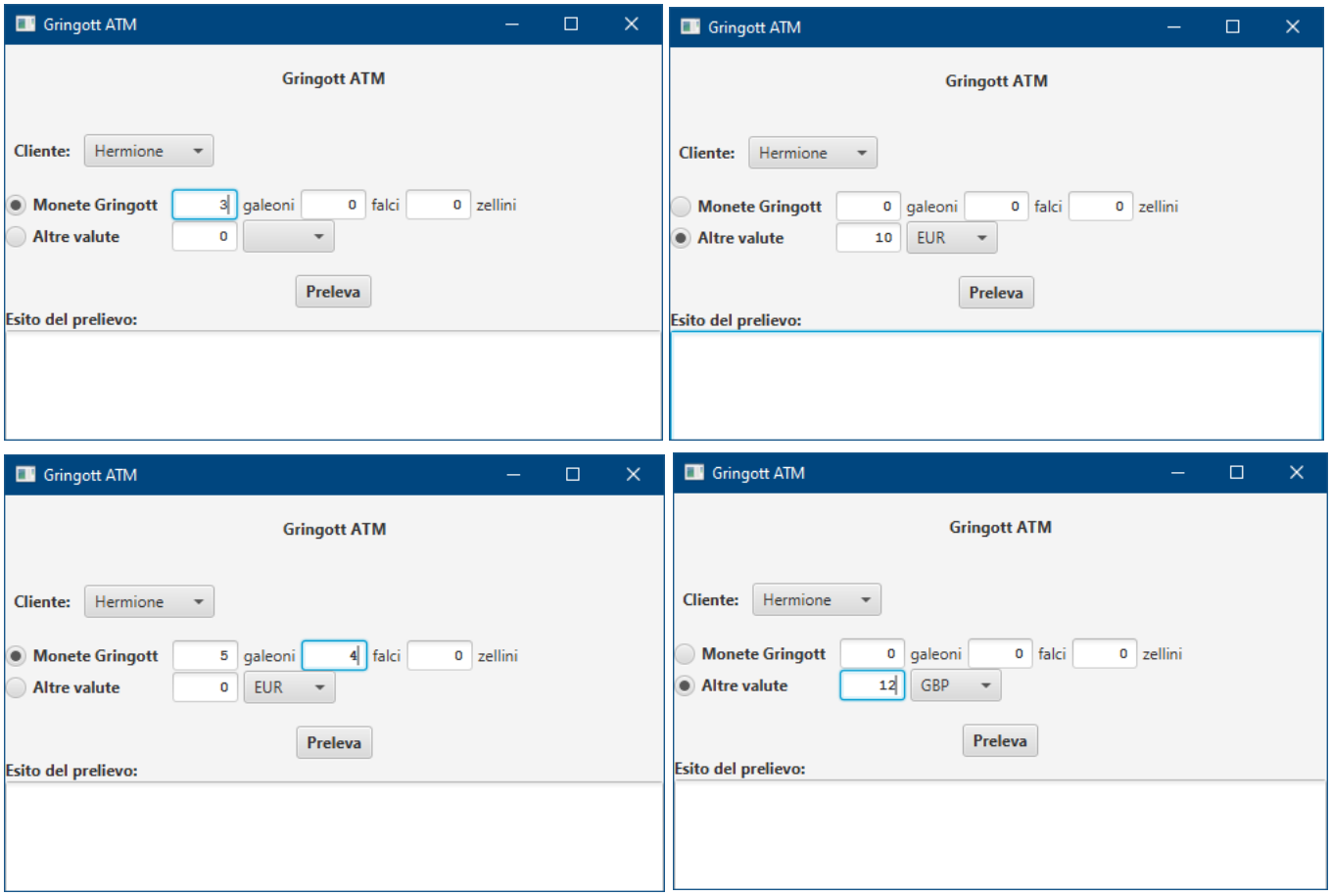


Fig. 1: la situazione iniziale della GUI, con combo vuota e nessuna selezione ancora effettuata

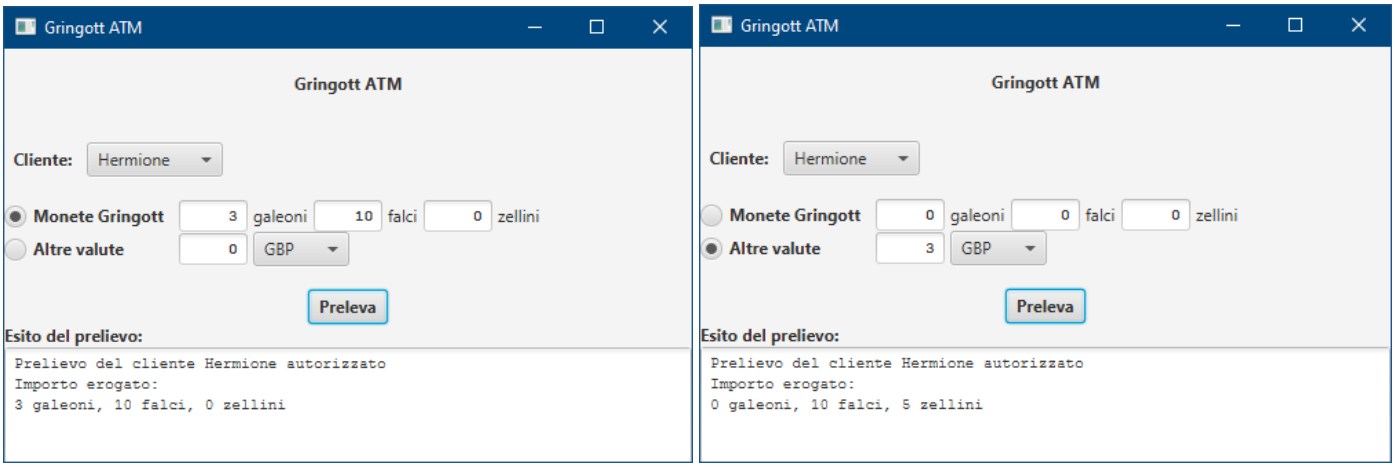
- in alto, una combo elenca i possibili clienti; subito sotto, due bottoni radio consentono di scegliere se specificare l'importo in monete Gringott o valute umane. A tal fine, di fianco a ciascuno di essi sono presenti alcuni campi di testo (con, occorrendo, apposite etichette descrittive) o, nel caso delle valute umane, la combo per scegliere quale valuta usare fra quelle supportate. Sotto a tutto ciò, il pulsante PRELEVA attiva la richiesta di prelievo.
- In basso, un'area di testo (inizialmente vuota e non scrivibile dall'utente) mostra i dettagli del prelievo.

Comportamento:

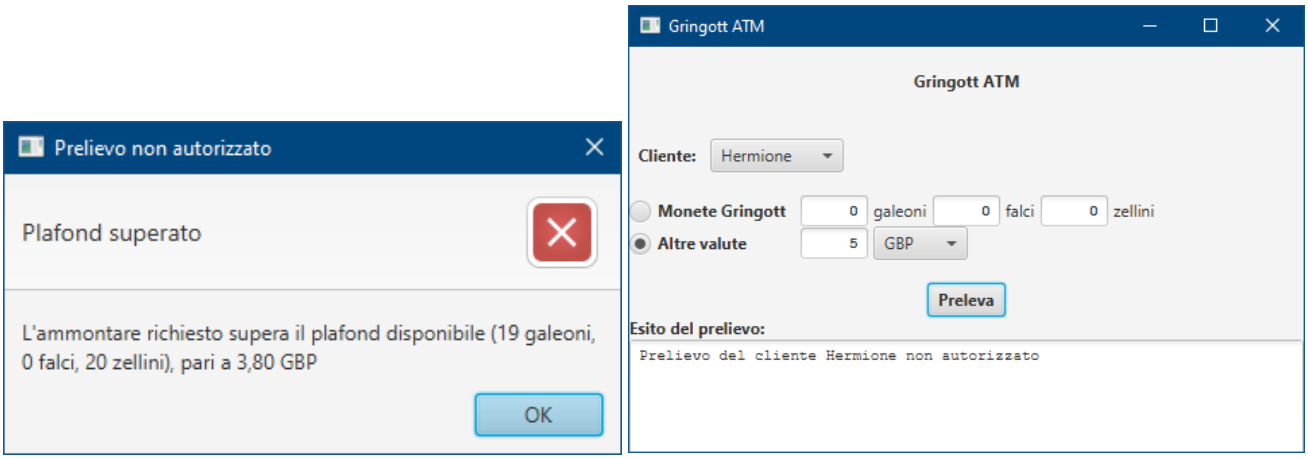
- la selezione di un utente dalla combo non produce effetti: tale valore è infatti consultato solo alla pressione del pulsante PRELEVA.
- La selezione di un bottone radio causa, invece, l'abilitazione in scrittura dei campi di testo a lato e la corrispondente disabilitazione e azzeramento di quelli non utilizzati (Figg. 2,3,4,5). Inizialmente dev'essere selezionato il caso delle monete Gringott.
- Premendo il pulsante PRELEVA, viene recuperato il nome del cliente (se non selezionato, dev'essere emesso apposito messaggio di errore tramite finestra di dialogo) e attivata, tramite Controller, la richiesta di prelievo: l'esito viene mostrato nell'area di testo sottostante (Figg. 6,7)



Figg. 2 / 3 / 4 / 5: la GUI selezionando monete Gringott (prima) o altre valute (dopo): si noti come in questo caso vengano azzerati i campi di testo delle monete Gringott. Dualmente, se dalle valute umane si tornano a selezionare monete Gringott, si azzerano a campi di testo delle prime; e così via.



Figg. 6 / 7 : prelievi autorizzati, sia in monete Gringott (prima) sia in valute umane (dopo).



Figg. 8 / 9: se il prelievo non è autorizzato, viene emesso su finestra di dialogo un apposito messaggio d'errore che ne dettaglia la causa (a sinistra), mentre l'area di testo riporta semplicemente l'esito negativo dell'operazione.

Il MainPane è fornito *parzialmente realizzato*: è presente quasi tutta l'impostazione strutturale, mentre sono da completare la configurazione di alcuni componenti e la gestione degli eventi.

In particolare, ***MainPane*** estende ***BorderPane*** e prevede:

1) sopra, una ***VBox*** altre box interne per combo, campi di testo, bottoni vari ed etichette ausiliarie
2) sotto, una ***VBox*** con la sola area di output.

La **parte da completare** riguarda:

1) la configurazione del gruppo di bottoni radio e l'aggancio dei relativi ascoltatori, già implementati nei due metodi ausiliari *coinButtonHandle* e *currencyButtonHandle*
2) la logica di gestione dell'evento, incapsulata nel metodo privato *myHandle*

In particolare, la gestione dell'evento principale, tramite *myHandle*, deve:

- verificare che sia selezionato un cliente nell'apposita combo e, se sì, recuperarlo, o in alternativa emettere apposito messaggio d'errore
- estrarre dai campi di testo i dati per il prelievo e, dopo averne verificato la correttezza, attivare il prelievo tramite i metodi *withdraw* del ***Controller*** intercettando, se necessario, **ImpossibleWithdrawException** per emettere (via Controller.*alert*) il corrispondente messaggio d'errore
- emettere nell'area di testo l'esito del prelievo, specificando innanzitutto se esso sia stato autorizzato e, in questo caso, l'importo effettivamente emesso.

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?