ESAME DI FONDAMENTI DI INFORMATICA T-2 del 7/2/2018
Proff. E. Denti – G. Zannoni
Tempo a disposizione: 4 ore MAX

NB: il candidato troverà nell'archivio ZIP scaricato da Esamix anche il software "Start Kit"

NOME PROGETTO ECLIPSE e CARTELLA : CognomeNome-matricola (es. RossiMario-0000123456)
NOME ZIP DA CONSEGNARE : CognomeNome-matricola.zip (es. RossiMario-0000123456.zip)

L'Unione Banche di Zannonia (UBZ) ha richiesto un'applicazione di ausilio per gli impiegati dei propri sportelli, che indichi visivamente quante banconote, e di che taglio, dare al cliente che effettui un prelievo in contanti.

DESCRIZIONE DEL DOMINIO DEL PROBLEMA

Quando un cliente si presenta allo sportello bancario per effettuare un prelievo, non tutte le combinazioni di banconote e monete possibili teoricamente lo sono anche realmente: ad esempio, non è realistico prelevare € 2000 in monete da € 2. Ciò che è effettivamente possibile dipende da:

1. l'importo da prelevare
2. la quantità di banconote dei vari tagli disponibili allo sportello in quel momento
3. le eventuali politiche aziendali (ad es. "non si danno più di dieci pezzi da € 10 in una singola operazione").

In generale, il cassiere sceglie sempre la combinazione che usa il massimo numero di banconote di taglio elevato, passando a usare banconote di taglio inferiore solo quando è indispensabile farlo.

ESEMPIO

Si supponga di voler prelevare € 180. Se lo sportello dispone di tutti i tagli e non ha limiti nella scelta delle banconote, le combinazioni possibili teoricamente sono molte: il cassiere però predilige quella che usa le banconote di taglio più elevato, ossia la prima fra quelle sotto elencate.

- 3x € 50 + 1x € 20 + 1x € 10 ← *prescelta dal cassiere*
- 3x € 50 + 3x € 10
- 9x € 20
- 7x € 20 + 4x € 10
- …
- 1x € 20 + 16x € 10
- 18x € 10

Tuttavia, se lo sportello non dispone di certe banconote (ad esempio, non ha banconote da € 10) o ne ha in quantità limitata (ad esempio, soltanto 5) o, per politica aziendale, non dà allo stesso cliente più di *tot* banconote di piccolo taglio (ad esempio, non più di 3 pezzi da € 10), alcune combinazioni non risultano più possibili e vanno escluse.
Ad esempio, se il cassiere ha finito le banconote da € 50, dovrà scegliere solo combinazioni che usino € 20, 10, 5, 2, 1.

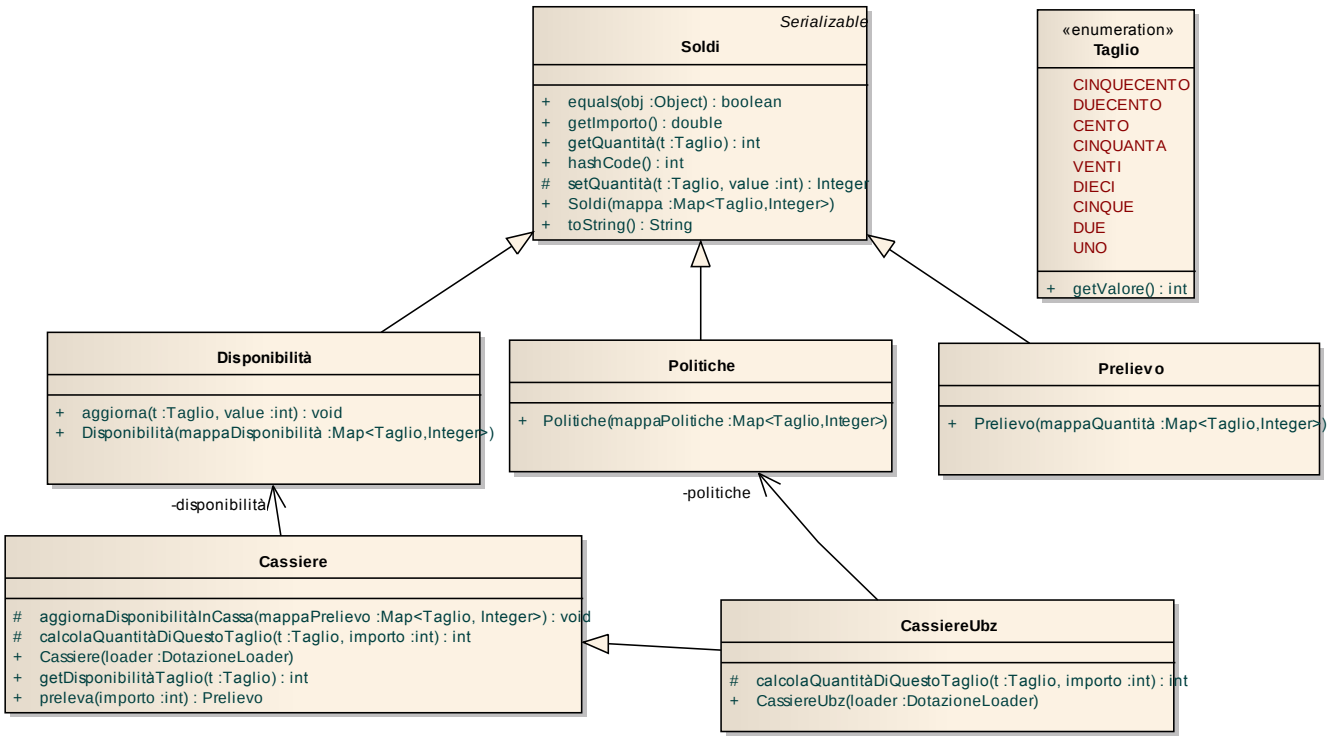
Nel caso di UBZ, la politica aziendale esprime appunto un tetto al numero di banconote o monete erogabili di un certo taglio: il cassiere UBZ, quindi, non dovrà tenere conto soltanto dei normali vincoli di cassa, come nell'esempio sopra, ma anche di questi ulteriori vincoli, che potranno portare a escludere determinate combinazioni – o, in casi limite, a rendere perfino impossibile il prelievo per impossibilità di comporre la cifra richiesta.

Il file **binario** DotazioneSportello.dat contiene **due oggetti** (una istanza di ***Disponibilità*** e una di ***Politiche***) che specificano, per ogni taglio di banconote e monete da 1€ e 2€ (si trascurano le monete di valore inferiore), rispettivamente le *quantità disponibili inizialmente* all'apertura dello sportello e le *politiche aziendali*, ossia il numero massimo di banconote o monete di quel taglio che si possono dare a un cliente nel corso di una singola operazione.

IMPORTANTE: eliminare dal progetto la parte grafica non usata.

Parte 1 (punti: 13)

Dati (package ubz.model) *(punti: 7)*



SEMANTICA:

a) L'enumerativo ***Taglio*** (fornito) elenca tutti i tagli di banconote e monete disponibili, in ordine decrescente da € 500 a € 1; ogni istanza dell'enumerativo incorpora il corrispondente valore numerico, recuperabile col metodo *getValore.*

b) La classe ***Soldi*** (fornita) rappresenta un dato insieme di banconote e monete, memorizzate internamente sotto forma di mappa **Map<Taglio,Integer>**: il costruttore riceve un argomento del medesimo tipo. Sono forniti i metodi accessor *getQuantità(Taglio)* e *setQuantità(Taglio, int)* e *getImporto*: il primo restituisce la quantità di banconote o monete di quel taglio presenti, il secondo [protetto] la re-imposta, mentre il terzo calcola e restituisce l'importo totale corrispondente ai soldi presenti. Il metodo *toString* stampa in forma sintetica l'importo e la composizione dello stesso.

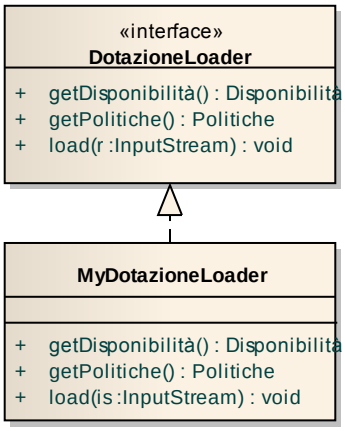
c) Le tre classi ***Disponibilità***, ***Politiche*** e ***Prelievo*** (fornite) specializzano ***Soldi*** per rappresentare rispettivamente i soldi presenti in cassa, le politiche aziendali, e un prelievo effettuato allo sportello: ***Disponibilità*** offre in più il metodo pubblico *aggiorna(Taglio t, int)* che consente di aggiornare la disponibilità in cassa di un certo taglio di banconote o monete.

d) La classe ***Cassiere*** (fornita) rappresenta lo sportello bancario: costruita a partire da un ***DotazioneLoader,*** incorpora al suo interno la ***Disponibilità,*** recuperabile tramite apposito accessor. Il metodo fondamentale è *preleva,* che, dato un importo, sintetizza la combinazione di banconote e monete da dare al cliente, restituendola sotto forma di ***Prelievo.*** A tal fine sfrutta due metodi *protetti*, *calcolaQuantitàDiQuestoTaglio* e *aggiornaDisponibilitàInCassa*: il primo calcola quante banconote o monete di un dato taglio sono necessarie per comporre il prelievo, mentre il secondo toglie dalla disponibilità di cassa le banconote e le monete utilizzate per un il prelievo. Nella versione base di questa classe, il metodo *calcolaQuantitàDiQuestoTaglio* non tiene conto delle politiche aziendali: pertanto, non c'è limite al numero di banconote o monete di un dato taglio che possono essere erogate in una singola operazione (se non quello della loro disponibilità in cassa).

e) la classe ***CassiereUbz*** (**da realizzare**) specializza ***Cassiere***: costruita a partire da un ***DotazioneLoader***, incorpora al suo interno le ***Politiche*** (in un oppotuno *field*) re-implementando *calcolaQuantitàDiQuesto-Taglio* in modo da tener conto delle politiche aziendali: pertanto, la quantità erogabile di un certo taglio sarà limitata, oltre che dalla disponibilità, anche dalle politica aziendale corrispondente.
NB: chi non riuscisse a svolgere questa parte potrà fornire una implementazione vuota che si limiti a ereditare dal cassiere base, in modo da poter proseguire con la successiva parte del compito.

Persistenza (ubz.persistence) *(punti 6)*

Come già anticipato, il file **binario** DotazioneSportello.dat contiene **due oggetti** (una istanza di ***Disponibilità*** e una di ***Politiche***) che specificano, per ogni taglio di banconote e monete, rispettivamente le *quantità disponibili inizialmente* all'apertura dello sportello e le *politiche aziendali*, ossia il numero massimo di banconote o monete di quel taglio che si possono dare a un dato cliente nel corso di una singola operazione.



SEMANTICA:

a) L'interfaccia ***DotazioneLoader*** (fornita) dichiara tre metodi:
   - *loadMaps* che carica dal file binario la disponibilità e le politiche aziendali;
   - *getDisponibilità* e *getPolitiche* restituiscono rispettivamente l'istanza di ***Disponibilità*** o ***Politiche*** caricate.

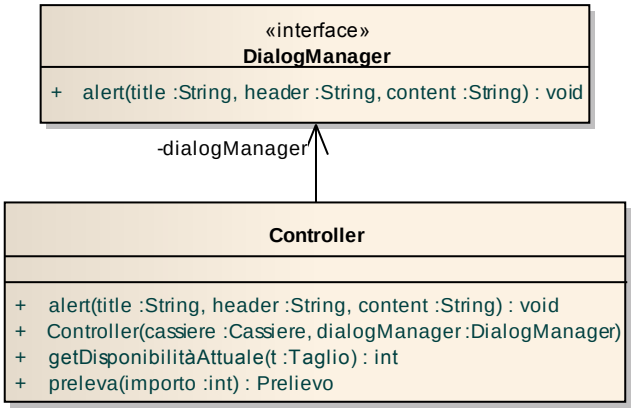
b) La classe ***MyDotazioneLoader*** (**da realizzare**) implementa ***DotazioneLoader***: il metodo *loadMaps* riceve in ingresso l'***InputStream*** da cui leggere i dati, ed emette:
   - una ***IllegalArgumentException*** nel caso l'inputstream ricevuto sia nullo;
   - l'opportuna ***BadFileFormatException*** con messaggio d'errore appropriato in caso di problemi nel formato del file (mancanza di entrambe le mappe, di una sola mappa, o presenza nel file binario di oggetti estranei di tipo diverso dalla mappa)
   - una ***IOException*** in caso di altri problemi di I/O.

PROSEGUE ALLA PAGINA SUCCESSIVA

Parte 2 (punti: 17)

Controller (ubz.ui.controller)

Il Controller è organizzato secondo il diagramma UML seguente:



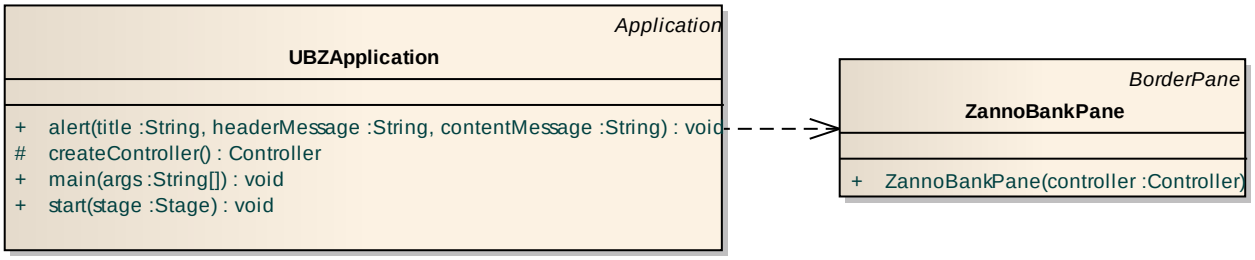
SEMANTICA:

La classe ***Controller*** (fornita) dichiara l'interfaccia del controller (metodo *getCassiere*) e implementa i metodi:

- *alert*, utile per mostrare avvisi all'utente tramite un ***DialogManager***: quest'ultimo è passato in fase di costruzione unitamente al ***Cassiere***;
- *getDisponibilitàAttuale*, che restituisce il numero di banconote disponibili per il taglio specificato: usa cassiere per ottenere tale informazione;
- *preleva*, che ridirige il controllo al cassiere per effettuare l'operazione di prelievo.

Interfaccia Utente JavaFX (ubz.ui.javafx) **per studenti A.A. 2016/17** *(punti 17)*

L'architettura segue il modello sotto illustrato:



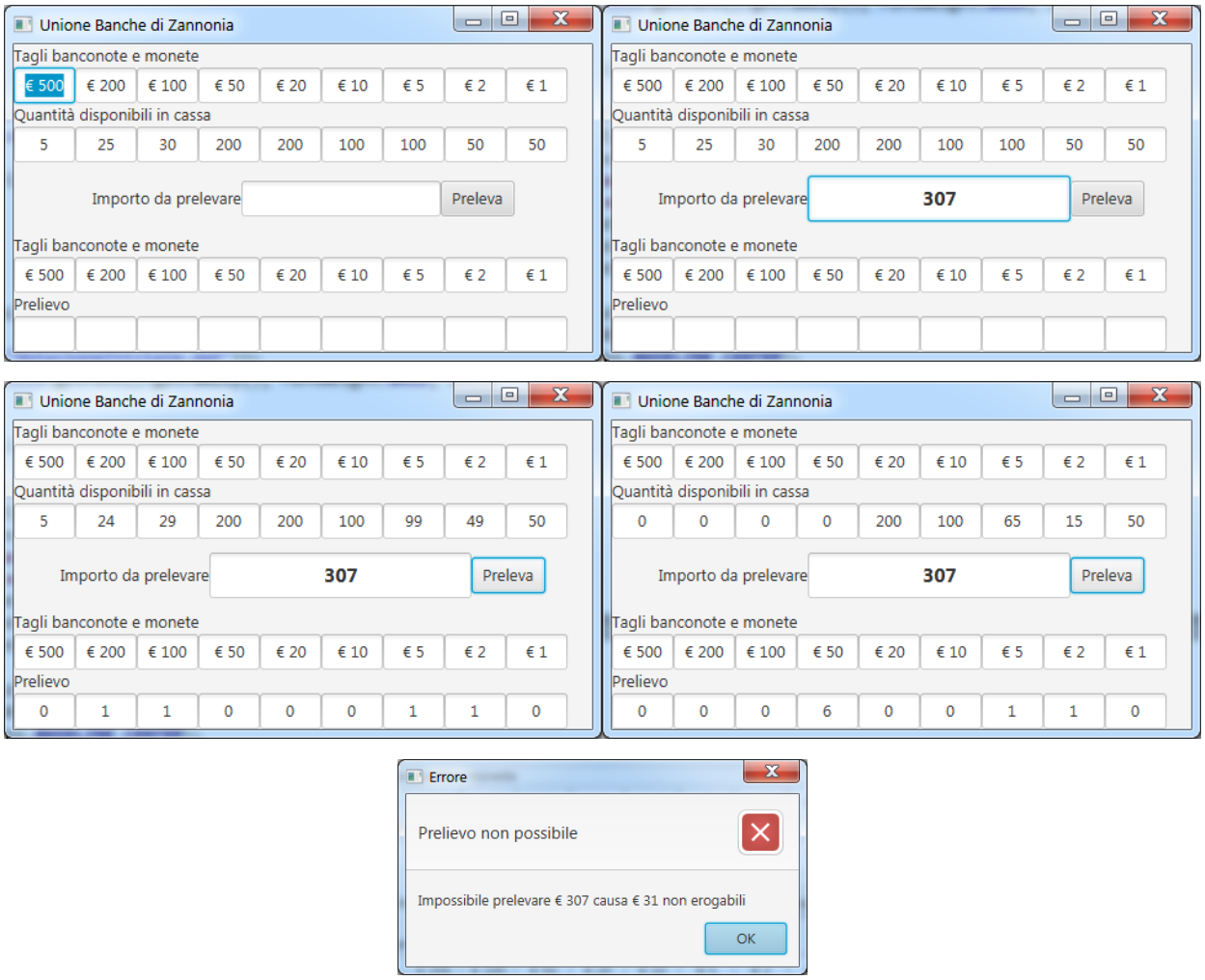
La classe ***UBZApplication*** (fornita) costituisce l'applicazione JavaFX che si occupa di aprire il file, il controller e incorporare lo ***ZannoBankPane*** (**da realizzare**). Per consentire di collaudare la GUI anche in assenza della parte di persistenza, è possibile avviare l'applicazione mediante la classe ***UBZApplicationMock***.

L'interfaccia utente deve essere simile (non necessariamente identica) all'esempio mostrato nella figura seguente.

In alto, una griglia di campi di testo (non editabili) elenca i possibili tagli di banconote e monete, seguiti da un'analoga griglia con le disponibilità di cassa (Fig. 1). Al centro, un campo di testo (con testo centrato e font bold 16 punti) consente di inserire l'importo da prelevare (Fig. 2): il successivo tasto *Preleva* attiva il prelievo. In risposta a tale evento (Fig. 3), le due griglie di campi di testo sottostanti dettagliano la composizione del prelievo (ossia quante banconote e monete di quali tagli vengono erogate): la disponibilità di cassa (in alto) cala di conseguenza.

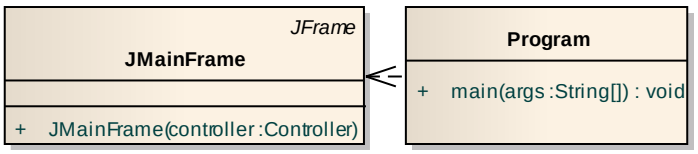
Di prelievo in prelievo, naturalmente, la cassa cala sempre più: le banconote di valore più elevato iniziano via via a mancare e il prelievo viene proposto con banconote di minor valore (Fig. 4), fino al punto in cui il prelievo stesso non può più avvenire (Fig. 5) – tipicamente, perché si violerebbero le politiche aziendali.

La classe ***ZannoBankPane*** deve estendere ***BorderPane***; per le griglie di campi di testo si suggerisce l'uso di opportuni ***TilePane.*** [NB: poiché non è possibile includere lo stesso componente grafico due volte in un Frame, la griglia dei tagli di banconote e monete deve essere duplicata per poter essere mostrata sia in alto sia in basso].



SWING: VEDERE ALLA PAGINA SUCCESSIVA

Interfaccia Utente Swing (ubz.ui.swing) **per studenti A.A. 2015/16 e precedenti** *(punti 17)*



L'interfaccia utente deve essere simile (non necessariamente identica) all'esempio mostrato nella figura seguente.

In alto, una griglia di campi di testo (non editabili) elenca i possibili tagli di banconote e monete, seguiti da un'analoga griglia con le disponibilità di cassa (Fig. 1). Al centro, un campo di testo (con testo centrato e font bold 16 punti) consente di inserire l'importo da prelevare (Fig. 2): il successivo tasto *Preleva* attiva il prelievo. In risposta a tale evento (Fig. 3), le due griglie di campi di testo sottostanti dettagliano la composizione del prelievo (ossia quante banconote e monete di quali tagli vengono erogate): la disponibilità di cassa (in alto) cala di conseguenza.

Di prelievo in prelievo, naturalmente, la cassa cala: le banconote di valore più elevato iniziano via via a mancare e il prelievo viene proposto con banconote di minor valore (Fig. 4), fino al punto in cui il prelievo non può più avvenire (Fig. 5) – tipicamente, perché si violerebbero le politiche aziendali.

[NB: poiché non è possibile includere lo stesso componente grafico due volte in un Frame, la griglia dei tagli di banconote e monete deve essere duplicata per poter essere mostrata sia in alto sia in basso].

La classe ***Program*** (fornita) contiene il *main* di partenza dell'intera applicazione, che si occupa di aprire il file e creare tutto il necessario. Per consentire di collaudare la GUI anche in assenza della parte di persistenza, è possibile avviare l'applicazione mediante la classe ***GUITest***.

La classe ***JMainFrame*** (***da realizzare***) deve organizzare l'interfaccia come sopra illustrato: il pannello principale deve adottare ***BorderLayout***. Per le griglie di campi di testo si suggerisce l'uso di opportuni sotto-pannelli gestiti da ***GridLayout.*** [NB: poiché non è possibile includere lo stesso componente grafico due volte in un JFrame, la griglia dei tagli di banconote e monete deve essere duplicata per poter essere mostrata sia in alto sia in basso].