

ESAME DI FONDAMENTI DI INFORMATICA T-2 del 17/1/2024

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"

DentiniaTV ha deciso di lanciare un nuovo programma di intrattenimento basato sul noto format del Gioco dei Pacchi. A tal fine ha richiesto lo sviluppo di un'applicazione che supporti tutta l'evoluzione del gioco. *Rispetto al format televisivo, questa versione NON prevede l'offerta del "cambio pacco", ma solo offerte in denaro.*

DESCRIZIONE DEL DOMINIO DEL PROBLEMA

Il Gioco dei Pacchi prevede preliminarmente l'associazione fra N "territori" (solitamente "Regioni", ma potrebbero essere anche province, comuni, o altro), altrettanti pacchi numerati da 1 a N, e altrettanti premi, di cui alcuni di basso valore e altri di valore elevato. L'associazione Territorio/Numero pacco/Premio contenuto è sorteggiata segretamente prima dall'inizio del gioco e non è nota ai partecipanti.

Uno dei partecipanti assume il ruolo di "concorrente", gli altri fungono da attori passivi per aprire i rispettivi pacchi quando il conduttore glielo richiede. ***In questa simulazione software, gli attori passivi saranno simulati dal computer, mentre l'utente/giocatore assumerà il ruolo di concorrente.***

Il gioco si basa sull'interazione fra il concorrente e il "Dottore", personaggio dietro le quinte che, volta per volta:

- dice al concorrente quanti pacchi aprire: ogni pacco aperto è "perso" dal concorrente, nel senso che il premio in esso contenuto non è più disponibile e viene tolto dal tabellone
- dopo che il concorrente ha aperto la quantità di pacchi richiesti, propone al concorrente un'offerta in denaro, "vagamente" proporzionata ai premi ancora disponibili: se il concorrente l'accetta, il gioco termina; altrimenti si passa alla manche successiva, ripetendo le stesse operazioni.

Il numero di pacchi da aprire a ogni manche viene stabilito dal Dottore volta per volta, con l'unico vincolo di garantire che nella manche finale (se raggiunta) rimanga in gioco almeno un pacco, oltre a quello già in possesso del concorrente.

In questa simulazione software:

- ***il numero di pacchi da aprire ogni volta è sorteggiato fra 1 e N/3***
- ***l'offerta del Dottore è sorteggiata fra 1/4 della media dei premi ancora disponibili e la media stessa.***

Territori e premi sono elencati nei file di testo [Territori.txt](#) e [Premi.txt](#), nel formato descritto più oltre.

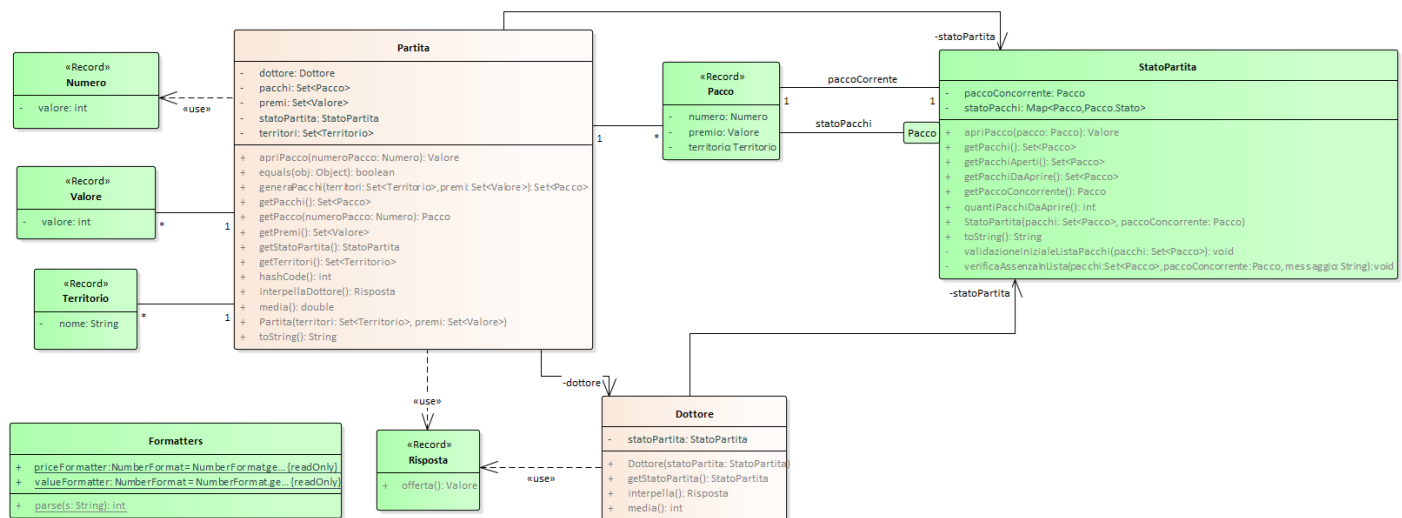
TEMPO STIMATO PER SVOLGERE L'INTERO COMPITO:

2h15 – 3h

PARTE 1 – Modello dei dati:	Punti 14	[TEMPO STIMATO: 80-95 minuti]
PARTE 2 – Persistenza:	Punti 8	[TEMPO STIMATO: 35-50 minuti]
PARTE 3 – Grafica:	Punti 8	[TEMPO STIMATO: 20-35 minuti]

JAVAFX – Parametri run configuration nei LAB

```
--module-path "C:\applicativi\moduli\javafx-sdk-19\lib"  
--add-modules javafx.controls
```



SEMANTICA:

- I record **Numero**, **Territorio**, **Valore** (forniti) costituiscono altrettanti *wrapper* per un valore (intero o stringa) al fine di garantire maggiore espressività rispetto al mero uso di un tipo primitivo e di una generica stringa. [NB: si ricorda che l'uso di record implica la generazione automatica di *equals*, *hashCode* e di tanti accessor quanti gli argomenti del record, ognuno di nome uguale all'argomento a cui accede]
- Analogamente, il record **Pacco** (fornito) esprime un pacco come associazione (Territorio, Numero, Valore). [NB: al suo interno è definito l'enumerativo di servizio **Stato**, non descritto in questa sede perché utilizzato unicamente da **StatoPartita** per registrare quali pacchi siano aperti e chiusi]
- La classe **Formatters** (fornita) definisce due formattatori, da usare in tutta l'applicazione per formattare e fare parsing di numeri reali (*valueFormatter*) e valori in Euro (*priceFormatter*) secondo lo standard italiano
- La classe **StatoPartita** (fornita) rappresenta lo stato della **Partita** (vedere oltre): il suo ruolo è registrare lo stato dei pacchi aperti/chiusi, nonché verificare la coerenza fra l'insieme dei pacchi dei partecipanti e il pacco del concorrente. Tali aspetti non vengono qui descritti in dettaglio in quanto non è previsto che questa classe debba essere manipolata direttamente dal candidato. Si evidenzia soltanto la presenza di alcuni metodi utili, richiamati comunque più oltre: *getPacchiDaAprire*, *getPaccoConcorrente* che restituiscono rispettivamente i soli pacchi ancora da aprire in mano ai partecipanti (escluso quindi quello del concorrente) e il solo pacco del concorrente. Tali metodi sono utili per implementare la classe **Dottore**, più oltre specificata.
- La classe **Partita** (da completare) ha due compiti: gestire la configurazione del gioco a partire dagli insiemi di territori e premi, e fungere da front-end verso la funzionalità di apertura pacchi gestita dalla classe di back-end **StatoPartita**. Per quanto riguarda il primo compito, controlla in primis la consistenza dei due set ricevuti come argomenti, poi genera tutti gli N pacchi, sorteggia quello del concorrente e lo rimuove dal set dei pacchi, che così facendo contiene d'ora in avanti tutti e soli i pacchi dei partecipanti. Espone molti metodi per accedere a tutti i set e le proprietà rilevanti, ma gli unici da implementare sono i due metodi:
 - apriPacco*, che apre il pacco con il numero specificato, restituendone il premio. Più in dettaglio, preliminarmente verifica che il numero del pacco sia nel range 1..N, altrimenti lancia **IllegalArgument-Exception** con opportuno messaggio d'errore; se tutto è ok, recupera dall'insieme dei pacchi quello con il numero richiesto, lo apre tramite il metodo *apriPacco* di **StatoPartita** e restituisce il **Valore** del premio in esso racchiuso.
 - generaPacchi*, che costruisce gli N pacchi sorteggiando i numeri e accoppiandoli a territori e premi. Più in dettaglio, itera sui due insiemi ricevuti accoppiando un territorio, un premio e un numero di pacco –

quest'ultimo sorteggiato fra 1 e N, senza ripetizioni e senza escluderne alcuno – usandoli per costruire un nuovo **Pacco**, che viene poi aggiunto all'insieme dei pacchi da restituire.

- f) Il record **Risposta** (fornito) incorpora un **Valore** inteso come risposta del Dottore
- g) La classe **Dottore** (da completare nell'implementazione) fornisce anch'essa fondamentalmente due metodi:
- **interpella**, che sintetizza la **Risposta** con l'offerta del dottore, calcolata sorteggiando un valore casuale compreso fra $M/4$ ed M , dove M è la media dei pacchi ancora da aprire, ottenuta dal metodo **media**
 - **media**, che calcola e restituisce, sotto forma di intero, la media dei pacchi ancora da aprire incluso quello del concorrente, per recuperare i quali si possono sfruttare i metodi **getPacchiDaAprire** e **getPaccoConcorrente** di **StatoPartita**

A tale scopo la classe **Dottore** riceve lo **StatoPartita** come argomento (e, per soli fini di test, lo rende anche accessibile tramite un apposito accessor).

Parte 2 – Persistenza

Package: **pacchi.persistence**

(punti: 8)

[TEMPO STIMATO: 35-50 minuti]

Territori e premi sono elencati nei due file di testo **Territori.txt** e **Premi.txt**, rispettivamente.

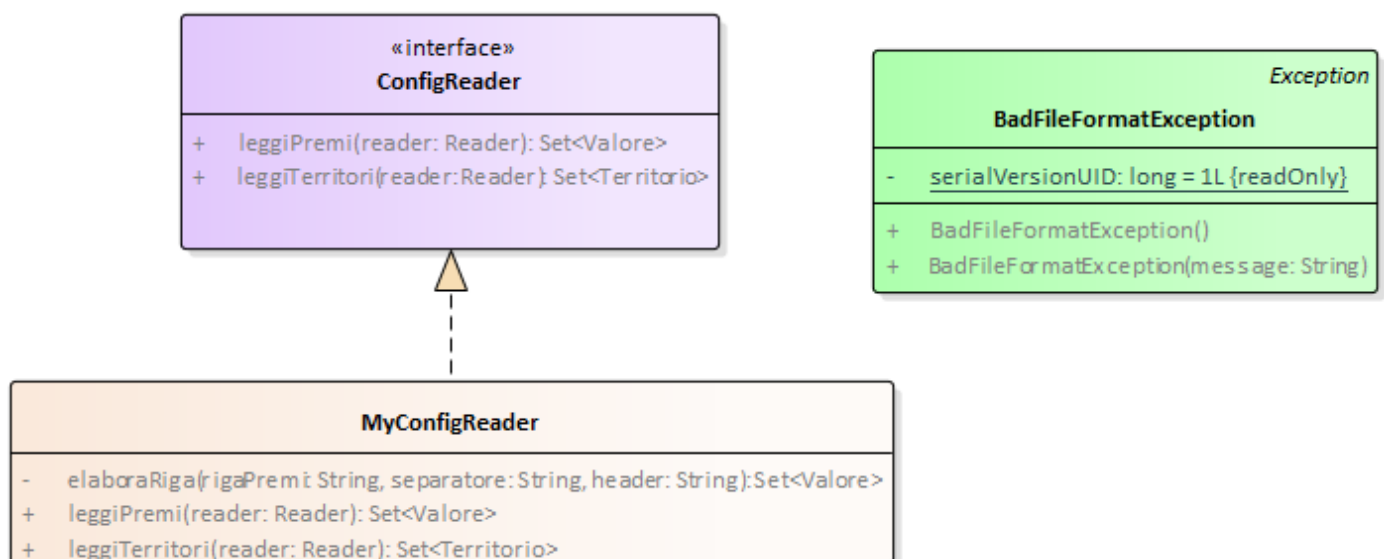
- i **territori** sono semplicemente elencati uno per riga: il numero delle righe non è ovviamente noto a priori;
- i **premi** sono invece elencati in due sole righe, una per i premi bassi ($<1000€$) e una per i premi alti ($\geq 1000€$), che devono contenere lo stesso numero di valori, nel seguente formato:
 - intestazione "PREMI ALTI:" o "PREMI BASSI:", rispettivamente
 - elenco dei rispettivi valori, separati da virgole, formattati come numeri in formato italiano, utilizzando ove appropriato il simbolo delle migliaia (SENZA simbolo di valuta)

Premi bassi e premi alti devono essere presenti in egual quantità.

ESEMPIO

PREMI BASSI: 0, 1, 5, 10, 20, 50, 75, 100, 200, 500

PREMI ALTI: 5.000, 10.000, 15.000, 20.000, 30.000, ..., 200.000, 300.000



SEMANTICA:

- a) L'interfaccia **ConfigReader** (fornita) dichiara i due metodi **leggiTerritori** e **leggiPremi** che caricano da un apposito Reader (già aperto) i dati necessari, restituendo un **Set** dell'opportuno tipo (**Territorio** o **Valore**) popolata. Entrambi lanciano:
- **IllegalArgumentException** con opportuno messaggio d'errore in caso di argomento (reader) nullo;
 - **BadFormatException** con messaggio d'errore appropriato in caso di problemi nel formato del file (mancanza/eccesso di elementi, errori nel formato dei numeri, etc.), *ivi incluso il caso in cui premi bassi e premi alti non siano presenti in egual quantità*
 - una **IOException** in caso di altri problemi di I/O.
- b) La classe **MyConfigReader** (da completare) implementa **ConfigReader** secondo le specifiche sopra descritte. Il metodo **leggiTerritori** è fornito già implementato: si deve quindi implementare soltanto **leggiPremi**. A tal fine, si conviene di delegare l'elaborazione della singola riga al metodo ausiliario privato **elaboraRiga**.

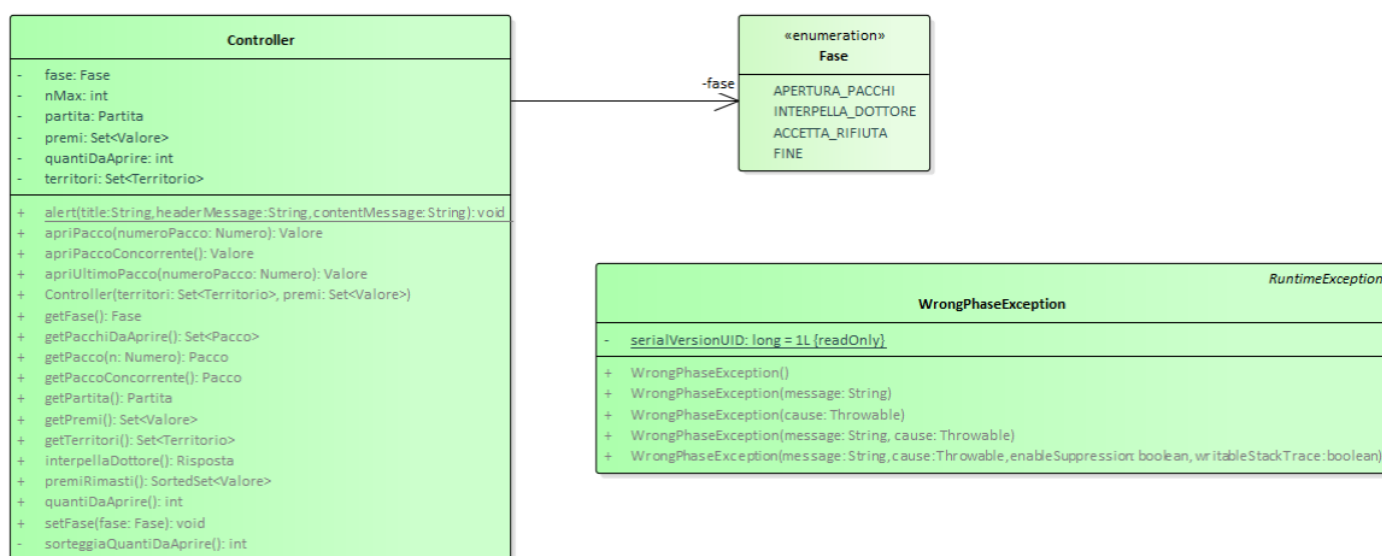
Parte 3

(punti: 8)

Package: **pacchi.controller**

(punti 0)

Il Controller è organizzato secondo il diagramma UML seguente:



SEMANTICA:

L'enumerativo **Fase** (fornito) descrive le quattro fasi in cui il gioco, e quindi il controller, può trovarsi: `APERTURA_PACCHI`, `INTERPELLA_DOTTORE`, `ACCETTA_RIFIUTA`, `FINE`.

La classe **Controller** (fornita) riceve in fase di costruzione gli stessi due insiemi di territori e premi già descritti nella classe **Partita**: se uno o entrambi sono nulli viene lanciata **IllegalArgumentException** con idoneo messaggio d'errore. Se invece tutto è regolare, il costruttore provvede a istanziare internamente la **Partita**, precalcolare il numero massimo di pacchi da aprire a ogni manche (da specifica, $N/3$) e impostare la fase ad `APERTURA_PACCHI`.

Molti metodi si limitano a richiamare quelli omonimi di **Partita**, altri forniscono invece servizi specifici.

Fra questi:

- **getPacchiDaAprire** restituisce l'insieme di pacchi ancora da aprire in un dato momento

- **apriPacco** apre il pacco di numero specificato, restituendone il valore contenuto. Il metodo agisce solo se la fase corrente è APERTURA_PACCHI, altrimenti lancia **WrongPhaseException** con apposito messaggio. Se il pacco da aprire era l'ultimo di questa manche, commuta la fase alla successiva INTERPELLA_DOTTORE.
- **apriUltimoPacco** è la versione specializzata per aprire l'ultimo pacco rimasto nell'ultima manche, quando il gioco ormai sta per terminare e resta solo da svelare cos'ha vinto il concorrente: per questo agisce solo se la fase attuale è FINE, altrimenti lancia **WrongPhaseException** con apposito messaggio.
- **apriPaccoConcorrente** apre il pacco del concorrente, mentre il metodo simile **getPaccoConcorrente** lo recupera e restituisce ma senza aprirlo
- **getFase/setFase** accedono alla fase corrente e permettono di modificarla
- **quantiDaAprire** restituisce il numero di pacchi da aprire in questa manche, preventivamente sorteggiato all'inizio della fase di apertura pacchi (tramite il metodo privato **sorteggiaQuantiDaAprire**)
- **premiRimasti** restituisce l'insieme ordinato di tutti i premi rimasti in gioco, ossia tutti i premi contenuti nei pacchi ancora da aprire più quello contenuto nel pacco del concorrente (il metodo è pensato per fornire i contenuti da mostrare sul tabellone di gioco).

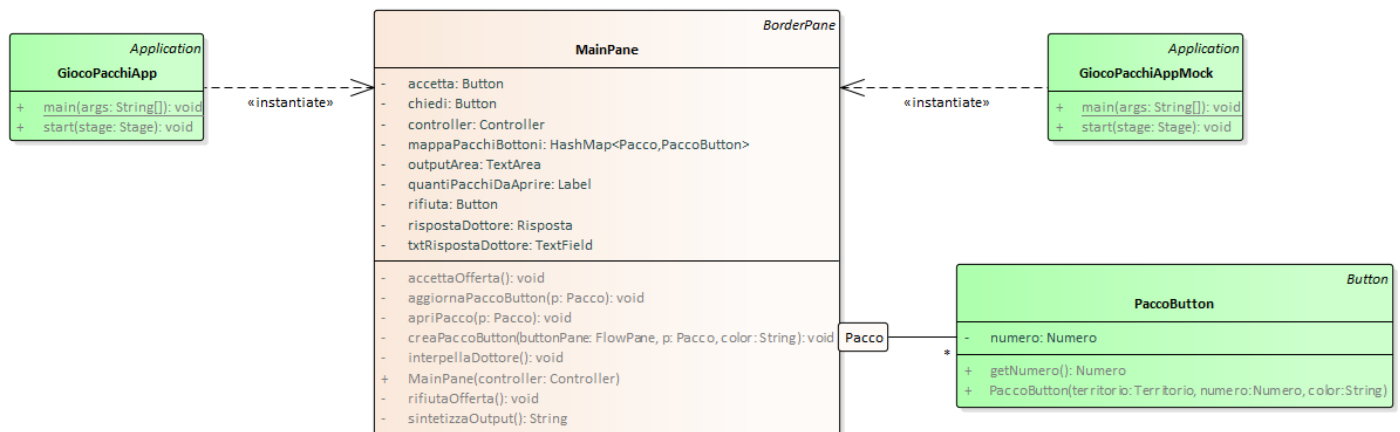
Infine, il metodo statico **alert**, utilizzabile anche dal **MainPane**, consente di far comparire all'utente, ove occorra, una finestra di dialogo con opportuno messaggio d'errore.

Package: pacchi.ui

[TEMPO STIMATO: 20-35 minuti] (punti 8)

La classe **GiocoPacchiApp** (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 **GiocoPacchiAppMock**: essa utilizza un minor numero di territori e può quindi essere utile anche per sveltire i collaudi.

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



L'interfaccia grafica si presenta come segue:

- a sinistra, tutta l'area di gioco, suddivisa in tre zone disposte verticalmente una sotto l'altra;
- a destra, un'area di testo che funge da tabellone, con l'elenco dei premi ancora in gioco e l'indicazione (al top) del pacco in mano al concorrente.

L'area di gioco si articola come segue:

- in alto, il pannello con i bottoni che rappresentano i pacchi dei partecipanti;
- di seguito, il bottone arancione che rappresenta il pacco del concorrente;
- sotto, l'area di competenza del Dottore, ulteriormente distinta in due parti:
 - il numero di pacchi ancora da aprire, decrescente fino a 0 ad ogni clic sui pulsanti del pannello;

- l'area più propriamente destinata all'interazione col Dottore, costituita dai pulsanti *Chiedi*, *Accetta*, *Rifiuta* e dal campo di testo (non scrivibile dall'utente) in cui compare l'offerta del Dottore.

Fig. 1: situazione iniziale. Territori e numeri dei pacchi variano ogni volta, così come l'associazione (ovviamente non visibile in figura) fra Pacco e Premio in esso contenuto. Anche il Pacco concorrente è estratto a sorte ogni volta.

Comportamento:

- il gioco inizia sempre dalla fase di apertura pacchi: il concorrente deve quindi aprire tanti pacchi quanti ne indica il Dottore nella riga apposita (in Fig. 1 sono quattro). Per farlo, basta che clicchi via via sui pulsanti corrispondenti: ogni pacco aperto viene svelato e il corrispondente pulsante disabilitato (Fig. 2). Contestualmente, il relativo premio viene tolto dal tabellone.
- una volta aperti tutti i pacchi richiesti, il gioco passa automaticamente alla fase INTERPELLA_DOTTORE, abilitando il pulsante "Chiedi" (Fig. 2, destra): in questa fase, ogni tentativo di aprire altri pacchi è impedito e causerebbe la comparsa di un alert (non mostrato) che spieghi la ragione del non potersi procedere
- Premendo il pulsante "Chiedi", il dottore formula l'offerta (Fig. 3, sinistra), che può essere accettata o rifiutata premendo gli appositi pulsanti "Accetta" o "Rifiuta", all'uopo abilitati: in contemporanea viene disabilitato invece il pulsante "Chiedi", per evitare che l'utente possa inavvertitamente chiedere un'altra offerta anzitempo.
- se l'offerta viene accettata (Fig. 5, destra), compare un dialogo che svela il contenuto del pacco del concorrente e il gioco termina: in contemporanea, il pulsante arancione che rappresenta tale pacco viene svelato e disabilitato.
- se invece l'offerta è rifiutata (Fig. 3, destra), il gioco torna alla fase iniziale di apertura pacchi (Fig. 4) e si procede come sopra, manche dopo manche.
- se, rifiuto dopo rifiuto, si giunge alla manche finale, il sistema propone un'ultima offerta: rifiutando anche quella, il concorrente vince il contenuto del proprio pacco, che viene quindi svelato (Fig. 5, sinistra).

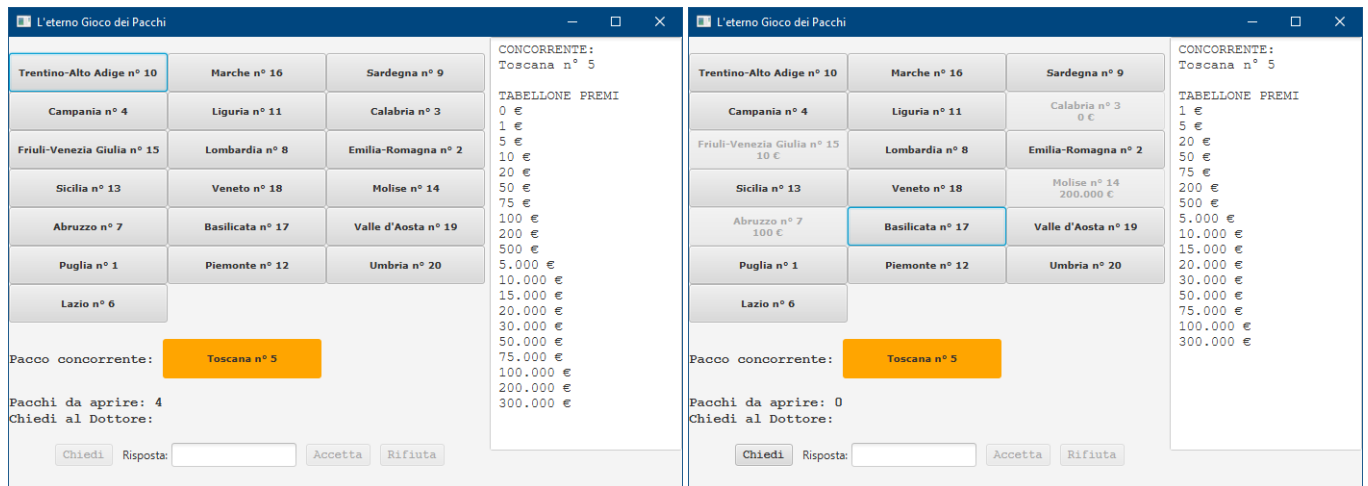


Fig. 2: a sinistra, la situazione iniziale; a destra, la GUI dopo aver aperto i 4 pacchi richiesti dal Dottore. Notare come ciò abiliti il pulsante “Chiedi” per interpellare il Dottore e attendere la sua offerta.

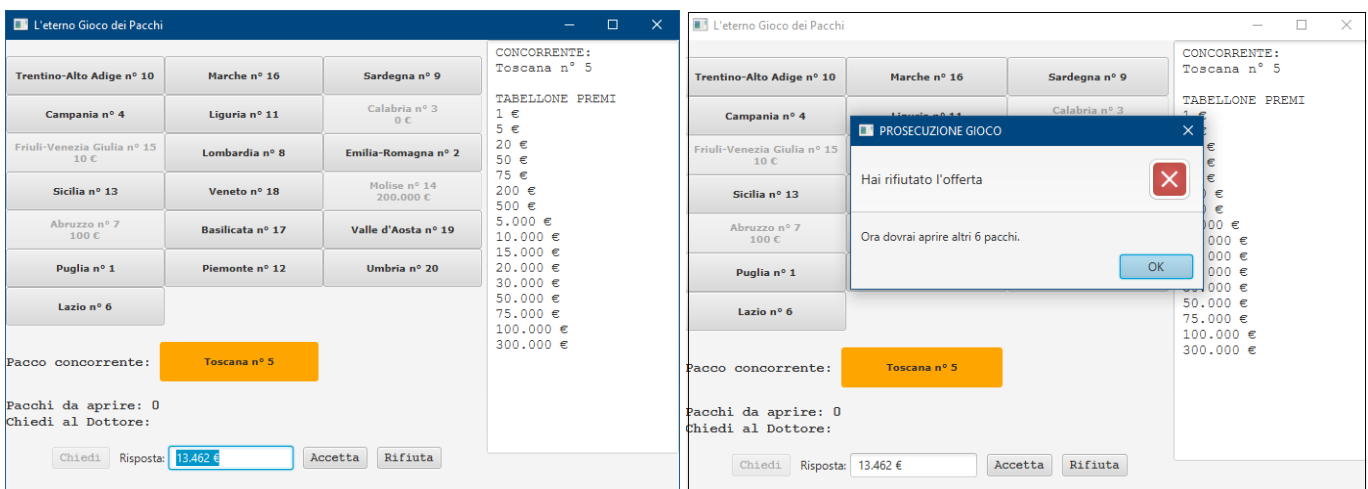


Fig. 3: a sinistra: l’offerta del Dottore è mostrata nell’apposito campo; in parallelo vengono abilitati i due pulsanti “Accetta” e “Rifiuta”, disabilitando il precedente pulsante “Chiedi”. A destra, rifiutando l’offerta appare il dialogo che conferma che il gioco prosegue e informa il concorrente di quanti pacchi dovrà aprire nella prossima manche.

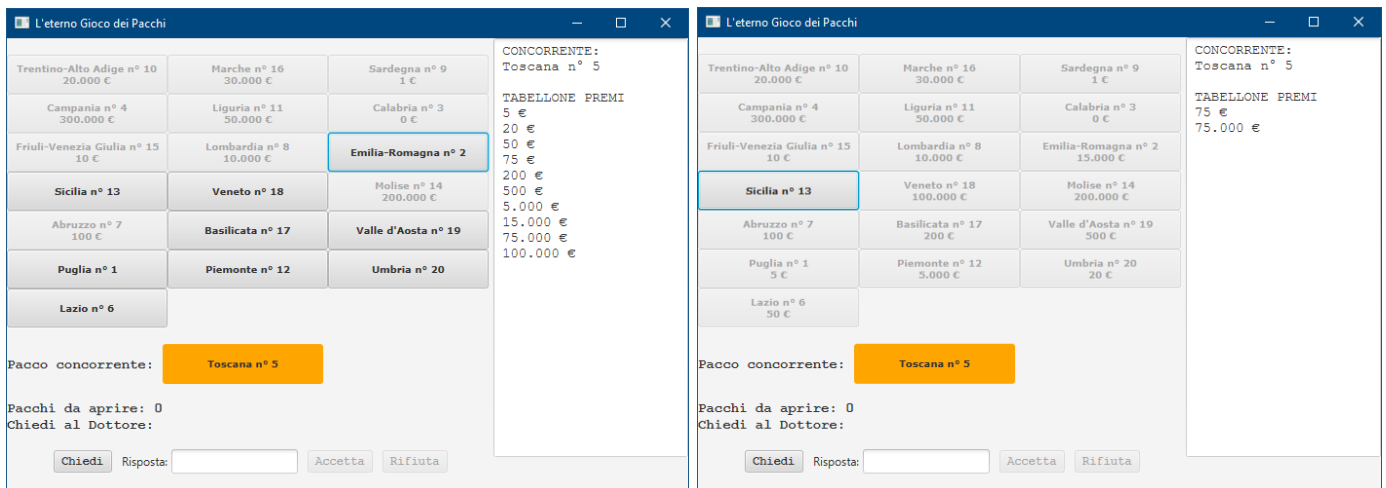


Fig. 4: a sinistra: il gioco prosegue ora come nel caso precedente, aprendo altri pacchi e chiedendo poi la successiva offerta; a destra: continuando a rifiutare tutte le offerte, si giunge infine alla manche finale.

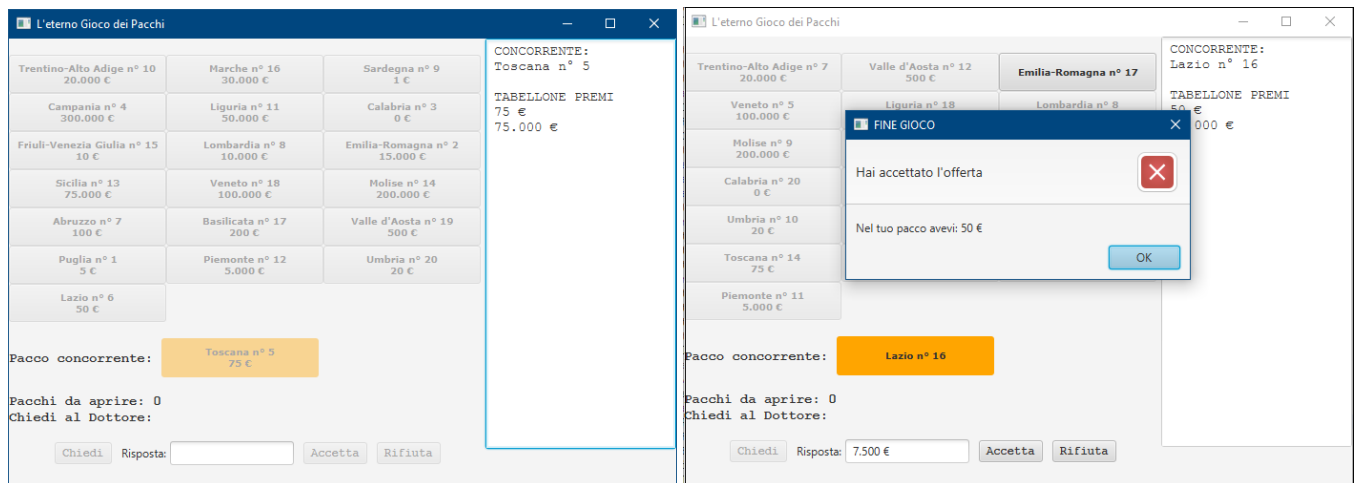


Fig. 5: a sinistra: rifiutando anche l'ultima offerta si vince il contenuto del proprio pacco (in questo caso, 75€); a destra: in alternativa (in un'altra partita) si può in qualsiasi momento accettare un'offerta. Il dialogo mostra quanto c'era nel pacco concorrente, poi la GUI viene aggiornata (non mostrato in figura).

Il MainPane è fornito *parzialmente realizzato*: è presente quasi tutta la parte strutturale ed è già implementata la gran parte dei metodi. Rimangono da realizzare solo tre funzionalità:

- 1) il popolamento iniziale del pannello con gli N-1 bottoni che rappresentano i pacchi dei partecipanti: il compito è facilitato dal metodo privato *creaPaccoButton* (fornito) che provvede anche ad agganciare già ai bottoni il relativo gestore degli eventi, costituito dal metodo privato *apriPacco* (anch'esso fornito)
- 2) il metodo privato *interpellaDottore*, che gestisce il pulsante "Chiedi"
- 3) il metodo privato *sintetizzaOutput*, che produce la stringa (articolata in righe come da figure) che popola il tabellone di gioco (sulla destra nelle figure).

In particolare, il metodo privato *interpellaDottore* deve:

- preliminarmente, verificare che il controller sia nella fase INTERPELLA_DOTTORE: se così non è, deve emettere (tramite il metodo alert del Controller) un messaggio d'errore e ritornare senza fare nulla
- se la fase di gioco è quella corretta, interpellare il Dottore e tramite l'omonimo metodo del controller, e mostrare la risposta nel campo di testo all'uopo previsto
- disabilitare il tasto "Chiedi" e ri-abilitare i due tasti "Accetta" e "Rifiuta"
- reimpostare il controller sulla fase ACCETTA_RIFIUTA, così da predisporre al giro successivo

Invece, il metodo privato *sintetizzaOutput* deve sintetizzare la stringa di seguito specificata:

- in alto, dopo un'intestazione come "CONCORRENTE", deve indicare territorio e numero del pacco in mano al concorrente (senza ovviamente svelarne il premio!)
- di seguito, dopo una riga bianca e un'intestazione come "TABELLONE PREMI", deve elencare i premi ancora disponibili (cioè quelli contenuti nei pacchi ancora da aprire, incluso ovviamente quello del concorrente), in ordine di valore crescente, utilizzando il formato valuta italiano standard: ovviamente, non deve dire in che pacchi siano contenuti! ☺

IMPORTANTE:

LEGGERE BENE LA CHECKLIST DI CONSEGNA ALLA PAGINA SEGUENTE!

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?