ESAME DI FONDAMENTI DI INFORMATICA T-2 del 5/09/2017
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)

La nota azienda "ED Creams & Dreams" leader nella produzione del gelato artigianale nella prospera EDLandia desidera offrire alle sue gelaterie l'applicazione "EDIceCream" che permetta la gestione di ciascun negozio. "ED Creams & Dreams" è suddivisa in un centro di produzione e svariate gelaterie che si occupano della vendita al dettaglio.

L'applicazione "EDIceCream" dovrà permettere di:

- leggere da file le scorte di gelato ricevute ogni giorno dal centro di produzione;
- gestire le scorte ricevute;
- definire ed inserire ogni gelato venduto
- visualizzare a video lo stato corrente della gelateria (gelati venduti, incasso, ecc...);
- stampare su file i resi giornalieri

DESCRIZIONE DEL DOMINIO DEL PROBLEMA

"ED Creams & Dreams" desidera offrire alle sue gelaterie una applicazione per migliorare la gestione di ogni singolo punto vendita. Ogni mattina ciascun punto vendita riceve la fornitura giornaliera di gelato dal centro di produzione. Tale fornitura è memorizzata nel file IceCreamQuantity.txt, dove per ogni gusto di gelato viene indicato il peso in grammi che è stato consegnato. Inoltre, al fine di mantenere la coerenza tra i diversi punti vendita, "ED Creams & Dreams" fornisce il file Kinds.txt che dettaglia le diverse tipologie di gelato, ad esempio:

- Cono piccolo: prezzo 2,50 euro, gusti massimi 2, peso totale del gelato 60gr

L'applicazione quindi deve permettere al venditore di comporre ogni singolo gelato partendo dalla specifica del tipo di gelato (*kind*) e successivamente scegliere i diversi gusti senza violare il numero massimo di gusti possibili per la specifica tipologia. Prima della vendita di ogni gelato occorre verificare che siano disponibili tutti i gusti, ovvero che per ogni gusto sia presente il quantitativo in grammi specifico per la tipologia di gelato, ad esempio:

Cono piccolo crema e cioccolato → va verificato che siano disponibili 30 gr sia di crema, sia di cioccolato

Cono piccolo cioccolato → va verificato che siano disponibili 60 gr di cioccolato

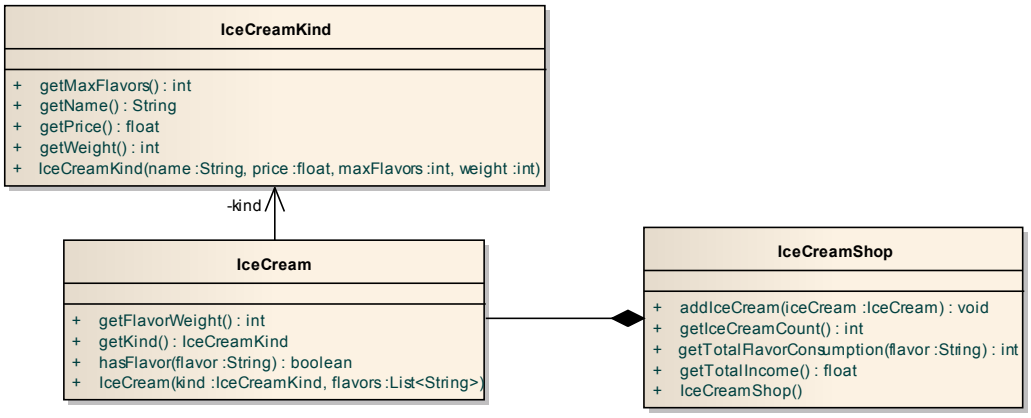
N.B. il gelato può avere meno gusti di quelli indicati nella tipologia!

Se non è possibile erogare il gelato deve comparire un messaggio a video. Dopo la vendita del gelato è necessario aggiornare la fornitura di gelato scalando i quantitativi venduti per ogni gusto. Ad ogni vendita "EDIceCream" deve provvedere ad aggiornare le informazioni a video relative al numero totale dei gelati venduti, i quantitativi per ogni singolo gusto e il totale incassato. A fine giornata inoltre è necessario stampare sul file "StockGiornaliero.txt" i quantitativi resi per ogni singolo gusto.

Parte 1 (punti: 15)

Dati (namespace edicecream.model) *(punti: 5)*

Il modello dei dati deve essere organizzato secondo il diagramma UML più sotto riportato.



SEMANTICA:

a) La classe ***IceCream*** (fornita nello Start kit) rappresenta i gelati venduti: il metodo ***getKind*** permette di recuperare il tipo di gelato (***IceCreamKind***), il metodo ***hasFlavor*** permette di verificare se il gelato è composto da uno specifico gusto, ed infine il metodo ***getFlavorWeight*** restituisce il peso in grammi di ogni gusto

b) La classe ***IceCreamKind*** (fornita nello Start kit) rappresenta le diverse tipologie di gelato in termini di nome, prezzo, massimo numero di gusti e peso totale del gelato. Sono messi a disposizioni i necessari metodi accessor per recuperare questi valori

c) La classe ***IceCreamShop*** (**da realizzare**) rappresenta la collezione dei gelati venduti e offre le seguenti funzionalità:

- ***addIceCream(IceCream icecream)****:* inserisce un nuovo gelato;
- ***getTotalIncome()***: restituisce il totale incassato;
- ***getTotalFlavorConsumption(String flavor)***: restituisce un intero che rappresenta il consumo totale in grammi per il gusto che viene passato in ingresso;
- ***getIceCreamCount():*** restituisce il numero totale di gelati venduti.

Lo Start Kit contiene anche i test (da includere nel progetto) per verificare il funzionamento di queste classi.

Persistenza (edcream.persistence) *(punti 10)*

Come già anticipato, il file di testo IceCreamQuantity.txt contiene l'elenco di tutti i gusti di gelato consegnati con l'indicazione del corrispondente quantitativo.

Il file IceCreamQuantity.txt contiene una riga per ogni gusto, nella forma:

```
NomeGusto - Quantità
```

dove

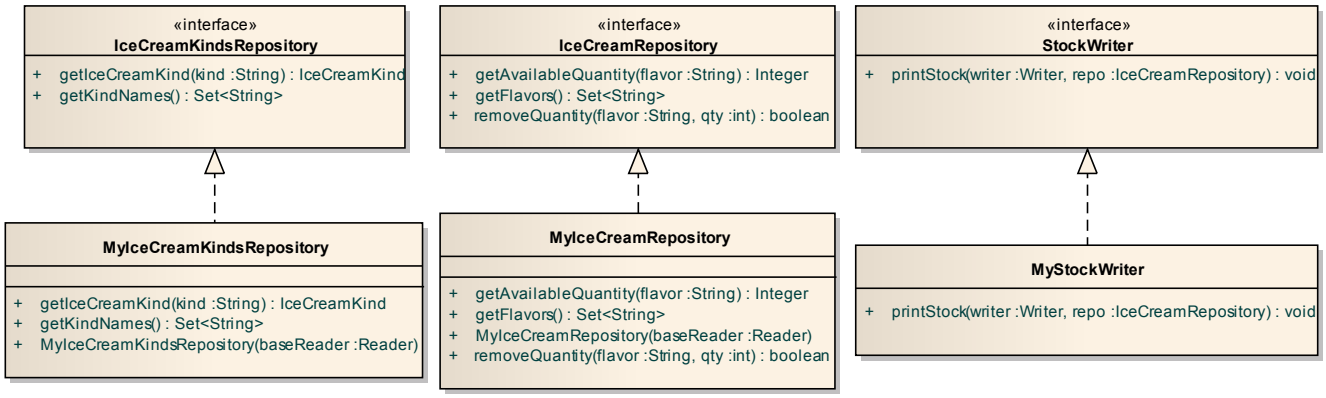
- *NomeGusto* rappresenta il nome del gusto del gelato: può contenere spazi in caso di nomi composti
- Zero o più spazi
- Un carattere '-'
- Zero o più spazi
- *Quantità* rappresenta il peso totale in grammi del gelato

Il file kinds.txt contiene una riga per ogni tipologia di gelato, nella forma:

```
NomeTipologia – Prezzo – NumGusti - Peso
```

dove

- *NomeTipologia* è il nome del tipo di gelato: può contenere spazi in caso di nomi composti
- Zero o più spazi
- Un carattere '-'
- *Prezzo* è un numero reale che rappresenta il prezzo del gelato in formato **italiano** (quindi con la virgola a separare la parte decimale e il '.' come separatore delle migliaia)
- Zero o più spazi
- Un carattere '-'
- *NumGusti* è un valore intero che indica il numero massimo di gusti per la tipologia del gelato
- Zero o più spazi
- Un carattere '-'
- *Peso* è un valore intero che rappresenta il peso massimo in grammi del gelato



SEMANTICA:

a) L'interfaccia ***IceCreamRepository*** (fornita nello Start kit) dichiara i metodi messi a disposizione dal repository che gestisce le scorte di gelato.

b) La classe ***MyIceCreamRepository*** (**da realizzare**) implementa ***IceCreamRepository:***

- riceve in ingresso nel costruttore il Reader da cui leggere i dati relativi alle scorte di gelato (se il parametro è *null* lancia una **IllegalArgumentExeption**). Il costruttore recupera tutti i diversi gusti ed il loro peso e li memorizza in una opportuna struttura dati che permetta **una ricerca veloce dei gusti per nome**. In caso di problemi nella lettura del file, si lascia uscire la ***IOException*** se i problemi sono relativi al file, mentre viene lanciata una ***BadFileFormatException*** (fornita nello Start kit) se i problemi sono relativi ai dati;
- ***getFlavors*** restituisce un **Set<String>** che contiene i *nomi* di tutti i gusti:
- ***getAvailableQuantity(String flavor)*** restituisce un intero che rappresenta la quantità in grammi ancora disponibili del gusto che viene passato in ingresso;
- ***removeQuantity(String flavor, int quantity)*** riceve in ingresso il nome del gusto e il quantitativo di grammi da rimuovere ed aggiorna le scorte .

c) L'interfaccia ***IceCreamKindsRepository*** (fornita nello Start kit) dichiara i metodi messi a disposizione dal repository che gestisce le diverse tipologie di gelato.

d) La classe ***MyIceCreamKindsRepository*** (**da realizzare**) implementa ***IceCreamKindsRepository:***

- riceve in ingresso nel costruttore il Reader da cui leggere i dati relativi alle diverse tipologie di gelato (se il parametro è *null* lancia una **IllegalArgumentExeption**). Il costruttore recupera tutti i diversi gusti ed il loro peso e li memorizza in una opportuna struttura dati che permetta una **ricerca veloce delle tipologie per nome**. In caso di problemi nella lettura del file, si lascia uscire la ***IOException*** se i problemi sono

relativi al file, mentre viene lanciata una ***BadFileFormatException*** (fornita nello Start kit) se i problemi sono relativi ai dati.

- ***getKindNames*** restituisce un **Set<String>** contenente i nomi delle diverse tipologie di gelato
- ***getIceCreamKind(String name)*** restituisce un oggetto di tipo ***IceCreamKind*** che rappresenta la tipologia di gelato relativa al nome passato come argomento

e) L'interfaccia ***StockWriter*** (fornita nello Start kit) dichiara i metodi messi a disposizione per la scrittura del file StockGiornaliero.txt

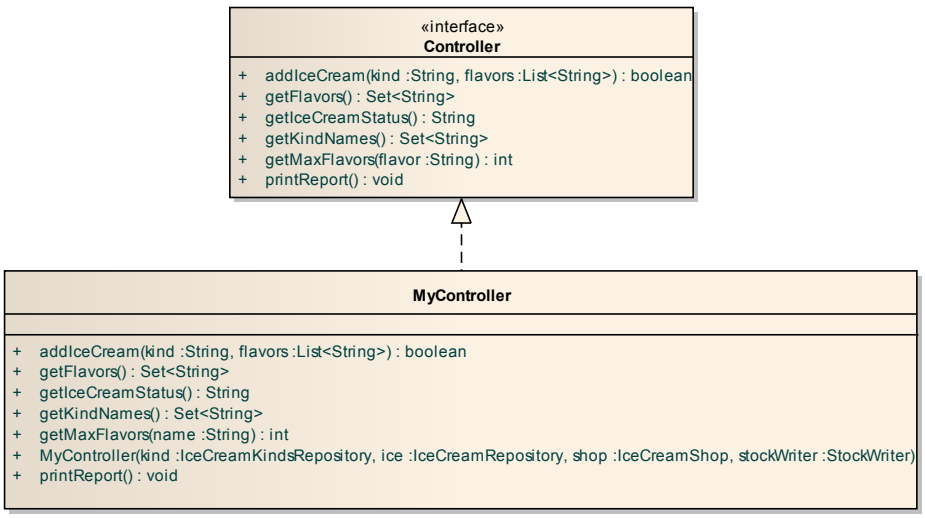
f) La classe ***MyStockWriter*** (fornita nello Start kit) implementa ***StockWriter*** e mette a disposizione il metodo ***printStock(Writer write, IceCreamRepository repo)*** che riceve in ingresso il Writer su cui scrivere i dati e il gestore delle scorte di gelato rimaste ed esegue la scrittura su file.

Lo Start Kit contiene anche i test (da includere nel progetto) per verificare il funzionamento di queste classi.

Parte 2 (punti: 15)

Controller (edicecream.controller) *(punti 7)*

Il Controller deve essere organizzato secondo il diagramma UML più sotto riportato.



SEMANTICA:

La classe ***MyController (da realizzare)*** implementa l'interfaccia ***Controller*** (fornita nello Start Kit), in particolare:

- il costruttore deve prevedere quattro argomenti di tipo ***IceCreamKindsRepository***, ***IceCreamRepository,*** ***IceCreamShop*** e ***StockWriter***
- il metodo ***getFlavors*** restituisce la lista ordinata di tutti i gusti di gelato a disposizione;
- il metodo ***getKindNames*** restituisce la lista ordinata di tutte le diverse tipologie di gelato;
- il metodo ***addIceCream(String kind, List<String> flavor)*** riceve in ingresso il nome della tipologia del gelato che si sta inserendo e la lista dei gusti richiesti, restituisce ***true*** nel caso sia possibile creare il gelato, ***false*** in caso contrario. È possibile inserire un nuovo gelato solo se sono soddisfatte due condizioni:
  - il numero dei gusti richiesti è inferiore o uguale al numero dei gusti massimi indicati nella tipologia;
  - per ogni gusto è disponibile nelle scorte il quantitativo di gelato necessario (N.B. ***IceCreamKind*** fornisce il peso massimo in grammi del gelato, non il peso per ogni gusto: quest'ultimo valore viene calcolato dalla classe ***IceCream***);

- il metodo ***printReport()*** che gestisce la stampa dei resi mediante l'uso di ***StockWriter*** e aprendo/chiudendo il file necessario. Tale metodo deve gestire eventuali errori di I/O;
- il metodo ***getMaxFlavors(String name)*** che restituisce il numero massimo di gusti relativi al nome della tipologia di gelato passato in ingresso;
- il metodo ***getIceCreamStatus()*** che restituisce una stringa contenente lo stato attuale dei gelati. La stringa deve essere della forma:

```
Totale gelati venduti: num
Venduti x gr di gusto1
Venduti y gr di gusto2
Totale incasso: xyz euro
```

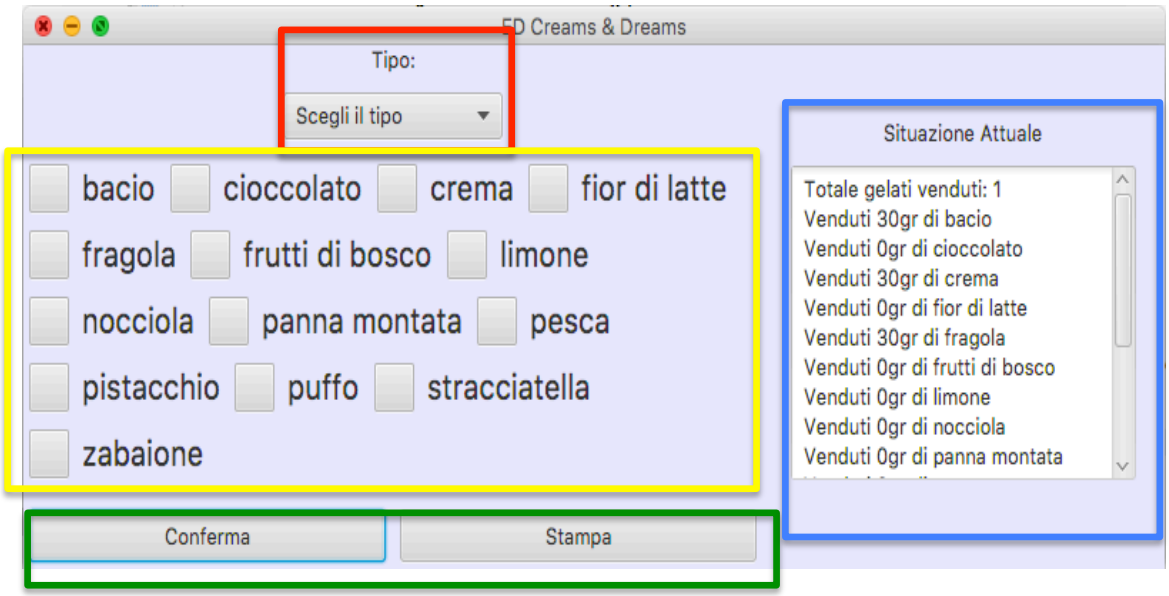
Interfaccia Utente *(punti 8)*

Studenti A.A. 2016/17 – versione con JavaFX

(package: edicecream.ui.javafx)

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

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



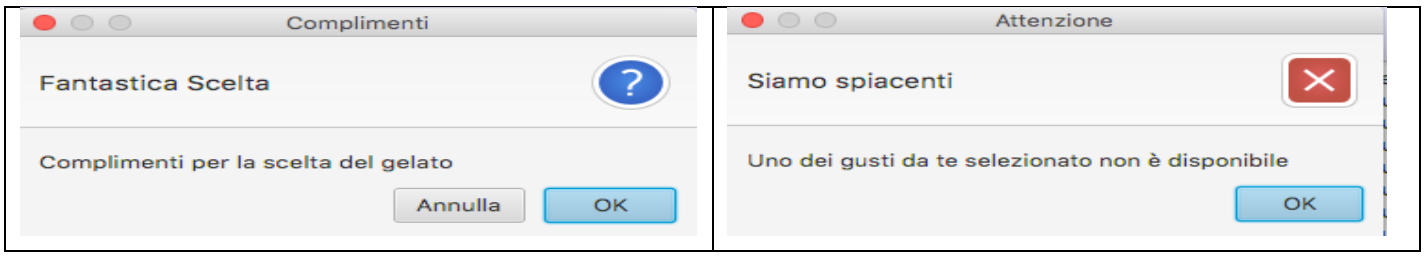
La classe ***IceCreamPane*** (**da realizzare**) deve estendere ***BorderPane***.

La GUI è organizzata in quattro sezioni:

1) Sezione inserimento tipologia gelato (parte sinistra in alto, evidenziata in rosso):
   - Una Label

- una ***ComboBox*** pre-popolata con i diversi tipi di gelato. Attenzione che quando cambia la tipologia di gelato va cambiato il valore del massimo numero di gusti selezionabili nel ***FlavorPanel*** (vedi dopo). All'avvio e dopo l'inserimento di ogni gelato la ***ComboBox*** deve presentare un *hint* con la scritta "Scegli il tipo " (usare il metodo **setPromptText** per settare il valore all'avvio).

2) Sezione scelta dei gusti (parte sinistra centrale, evidenziata in giallo):
   - Un ***FlavorPane*** (fornito nello Start kit) che permette la selezione dei gusti del gelato. Se si tenta di inserire un numero di gusti maggiore rispetto a quello massimo viene visualizzato un messaggio di errore. Tale classe fornisce le seguenti funzionalità:
     - Costruttore: riceve in ingresso un insieme di stringhe che rappresentano i diversi gusti del gelato. Il costruttore imposta a 0 il massimo numero di gusti selezionabili in modo tale che non sia possibile selezionare gusti finché non è stato selezionato il tipo di gelato.
     - ***setMaxSelected*** permette di impostare il massimo numero di gusti selezionabili.
     - ***getSelected*** restituisce una ***List*** dei gusti selezionati
     - ***reset*** permette di resettare il pannello deselezionando tutte le **checkbox**
3) Sezione risultato (parte destra, evidenziata in blu)
   - ***Label*** che specifica il contenuto della ***TextArea***
   - ***TextArea*** che si aggiorna ad ogni inserimento di gelato.
4) Sezione pulsanti (parte sinistra in basso, evidenziata in verde):
   - Un ***Button*** per la conferma del gelato da inserire. La pressione di tale bottone deve mostrare a video una **Alert** con il risultato dell'inserimento del gelato come indicato nelle figure sottostanti e deve aggiornare il testo della ***TextArea*** che rappresenta lo status attuale della gelateria: totale gelati venduti, grammi venduti per ogni gusto e totale incasso.



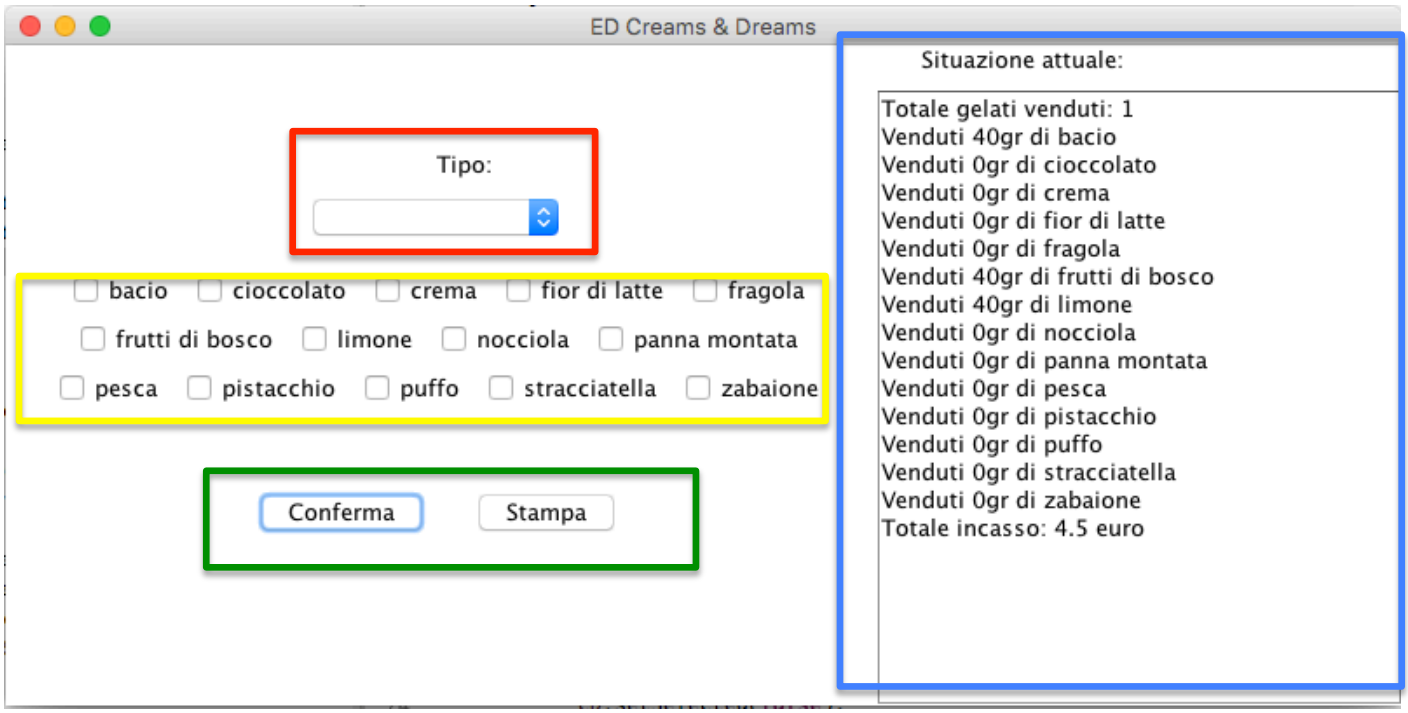
- Un ***Button*** stampa la cui pressione deve permettere di stampare su file il reso giornaliero.

Studenti A.A. 2015/16 (e precedenti) – versione con Swing

(package: zannofantasyfootball.ui.swing)

La classe ***EDIceCreamMain*** (fornita) costituisce il main per Swing che si occupa di aprire i file, creare il controller e incorporare il ***IceCreamFrame*** (**da realizzare**). Per consentire di collaudare la GUI anche in assenza della parte di persistenza, è possibile avviare l'applicazione mediante la classe ***GUITest.***

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



La classe ***IceCreamFrame*** (**da realizzare**) deve estendere ***JFrame***.

La GUI è organizzata in quattro sezioni:

5) Sezione inserimento tipologia gelato (parte sinistra in alto, evidenziata in rosso):
   - Una ***JLabel***
   - una ***JComboBox*** pre-popolata con i diversi tipi di gusti. Attenzione che quando cambia la tipologia di gelato va cambiato il valore del massimo numero di gusti selezionabili nel ***FlavorPanel*** (vedi dopo).
6) Sezione scelta dei gusti (parte sinistra centrale, evidenziata in giallo):
   - Un ***FlavorPanel*** (fornito nello Start kit) che permette la selezione dei gusti del gelato. Se si tenta di inserire un numero di gusti maggiore rispetto a quello massimo viene visualizzato un messaggio di errore. Tale classe fornisce le seguenti funzionalità:
     - Costruttore riceve in ingresso un insieme di stringhe che rappresentano i diversi gusti del gelato. Il costruttore imposta a 0 il massimo numero di gusti selezionabili in modo tale che non sia possibile selezionare gusti finché non è stato selezionato il tipo di gelato.
     - ***setMaxSelected*** permette di impostare il massimo numero di gusti selezionabili
     - ***getSelected*** restituisce un ***List*** dei gusti selezionati
     - ***reset*** permette di resettare il pannello deselezionando tutte le checkbox
7) Sezione risultato (parte destra, evidenziata in blu)
   - ***JLabel*** che specifica il contenuto della ***JTextArea***
   - ***JTextArea*** che si aggiorna ad ogni inserimento di gelato.
8) Sezione pulsanti (parte sinistra in basso, evidenziata in verde):
   - Un ***JButton*** per la conferma del gelato da inserire. La pressione di tale bottone deve mostrare a video un JOptionPane con il risultato dell'inserimento del gelato come indicato nelle figure sottostanti e deve aggiornare il testo della ***JTextArea*** che rappresenta lo status attuale della gelateria: totale gelati venduti, grammi venduti per ogni gusto e totale incasso.

- Un ***JButton*** stampa la cui pressione deve permettere di stampare su file il reso giornaliero