Restaurant Manager Report

Simone Di Cesare

July 2022

1 Introduzione

Il Progetto consiste nella realizzazione di un Gestore di un Ristorante con le seguenti funzionalità:

- Creazione e Gestione di un Menu di Portate struttrate in {ID, NOME, PREZZO}
- Gestione degli Ordini, dalla creazione alla preparazione in cucina di essi
- Gestione dei Pagamenti degli Ordini, con stampa degli Scontrini.

2 Descrizione Delle Classi

2.1 ImageButtonHighlighted

Estende JLabel.
Questa classe è usata per la creazione di bottoni che cambiano la loro icona al passare del mouse sopra il componente.

Parametri:

- **icon:** L'icona di base usata dal bottone.
- **highlightedIcon:** L'icona usata quando il mouse è sopra il bottone.
- **action:** L'azione svolta dal bottone una volta premuto su di esso.

Costruttori:

- **ImageButtonHighlighted(String, String):** Costruttore della classe che prende come parametri due stringhe, rappresentati il percorso all'interno del progetto delle immagini utilizzate come icone per il bottone.

Metodi:

- **setAction(ActionListener):** Setta l'azione del bottone

- **mouseExited(MouseEvent):** Evento chiamato all'uscita del mouse dal componente. Resetta l'icona del bottone da quella highlighted a quella base.
- **mouseEntered(MouseEvent):** Evento chiamato all'entrata del mouse dal componente. Setta l'icona del bottone da quella base a quella highlighted.
- **mouseClicked(MouseEvent):** Evento chiamato alla pressione del tasto del mouse sul bottone. Richiama il metodo **actionPerformed()** dall'oggetto **action** della classe, se esso non è nullo.

2.2 TextButtonHighlighted

Estende JLabel.
Questa classe è usata per la creazione di bottoni che cambiano il colo colore del testo al passare del mouse sopra il componente.
Costruttori:

- **TextButtonHighlighted(Color, Color)** Costruttore della classe che prende come parametri due colori, rappresentanti in ordine il colore base del testo e il colore bright del testo.

Parametri:

- **baseForeground:** Il colore del testo di base.
- **brightForeground:** Il colore del testo quando il mouse è sopra il componente.
- **action:** L'azione svolta dal bottone una volta premuto su di esso.

Metodi:

- **setAction(ActionListener):** Setta l'azione del bottone.
- **mouseExited(MouseEvent):** Evento chiamato all'uscita del mouse dal componente. Resetta il colore del testo da quello bright a quello di base.
- **mouseEntered(MouseEvent):** Evento chiamato all'entrata del mouse nel componente. Setta il colore del testo da quello base a quello bright.
- **mouseClicked(MouseEvent):** Evento chiamato alla pressione del tasto del mouse sul bottone. Richiama il metodo **actionPerformed()** dall'oggetto **action** della classe, se esso non è nullo.

2.3 MenuScopeDialog

Estende JDialog.
Questa classe è usata per la creazione e la modifica dei MenuScope.
Permette la visualizzazione e la modifica tramite JTextField del nome e del costo di una MenuScope.
Costruttori:

- **MenuScopeDialog(MenuScope, JFrame):** Costruttore della classe che prende come parametro una MenuScope e un JFrame come parente. La chiusura del MenuScope permette il ritorno della MenuScope stessa, o di null in caso l'operazione venga abortita.

Parametri:

- **scope:** La MenuScope da modificare/creare all'interno del MenuScopeDialog.
- **shouldSave:** Parametro che determina se all'uscita dal MenuScopeDialog vada ritornata la MenuScope appena creata/modificata (true), o null in caso di avorto dell'operazione (false).

Metodi:

- **showDialog():** Metodo che visualizza il MenuScopeDialog, e aspetta la sua chiusura. Alla chiusura, restituisce un parametro di tipo MenuScope in base al campo **shouldSave** della classe. Se il campo è true, il metodo ritorna la MenuScope appena creata/modificata all'interno del MenuScopeDialog, altrimenti restituisce null.

2.4 MainPanel

Estende AbstractPanel.
Questa Classe rappresenta il menù principale dell'applicazione. Al suo interno possiamo trovare i bottoni che permettono di aprire i diversi menù: Cameriere, Cuoco, Chef, Cassiere.
Costruttori:

- **MainPanel()** Costruttore che crea un ogetto di tipo MainPanel, con i bottoni per l'apertura dei menù: Cameriere, Cuoco, Chef, Cassiere; e il bottone di chiusura dell'applicazione.

Parametri:

- **waiterSelectionButton:** Il bottone che permette di accedere al menù del Cameriere.
- **cookSelectionButton:** Il bottone che permette di accedere al menù del Cuoco.

- **chefSelectionButton:** Il bottone che permette di accedere al menù dello Chef.
- **cashierSelectionButton:** Il bottone che permette di accedere al menù del Cassiere.
- **exitButton:** Il bottone che permette di chiudere l'applicazione.
- **employeeCount:** Parametro che indica quanti ruoli ci sono nel menù. Inizializzato a zero, incrementa ogni volta che viene chiamato il metodo **addEmployeeButton()**.

Metodi:

- **addEmployeeButton(String text):** Metodo che prende come parametro una Stringa rappresentante il nome del dipendente da aggiungere sottoforma di bottone. Restituisce un oggetto di tipo **TextButtonHighlighted**, dal colore di base Grigio Chiaro, e dal colore bright Giallo. Al suo richiamo, il parametro **employeeCount** viene aumentato.

2.5 CashierPanel

Estende AbstractPanel.
Questa Classe rappresentà il menù del Cassiere. Da qui è possibile visualizzare li scontrini dei tavoli con almeno una portata cucinata, ed in caso stampare lo scontrino per il pagamento.
Costruttori:

- **CashierPanel(Menu)** Costruttore che crea un oggetto di tipo CashierPanel, al quale viene associato un Menu, usato per l'estrapolazione dei dati delle portate dall'ordine.

Parametri:

- **menu:** Il menu associato alla cassa.
- **currentOrder:** L'ordine correntemente visualizzato all'interno della cassa.
- **receiptPanel:** Pannello per la visualizzazzione del conto.
- **receiptPanelScrollPane:** JScrollPane usata per contenere il **receiptPanel** all'interno della GUI.
- **receiptTotalLabel:** Testo che contiene e visualizza il totale del conto del tavolo.

Metodi:

- **refreshReceiptPanel():** Metodo usato per aggiornare la GUI una volta selezionato un nuovo tavolo, o una volta stampata la ricevuta di un tavolo.

Classi Interne:

- **DishPanel**
  Estende JPanel.
  Classe interna utilizzata per la visualizzazione di una portata, con il suo nome, la sua quantità, il suo costo, ed il costo totale dei piatti ordinati.

  Parametri:

  – **dish:** Il piatto visualizzato dal DishPanel.

  Costruttori:

  – **DishPanel(Dish):** Prende come parametro una variabile di tipo Dish, e crea un DishPanel che visualizza il Dish specificato dal parametro.

  Metodi:

  – **getFormattedName():** Restituisce il nome di una portata, formattato in modo da rientrare in una larghezza di 16 caratteri.

- **ReceiptPanel**
  Estende JPanel.
  Classe che contiene una lista di DishPanel caricati dai piatti cotti di un ordine.

  Costruttori:

  – **ReceiptPanel():** Costruisce un oggetto di tipo ReceiptPanel per la visualizzazione di uno scontrino.

2.6 ChefPanel

Estende AbstractPanel.
Questa Classe rappresenta il menù dello Chef. Da qui è possibile modificare, creare ed eliminare portate da un menù.
Costruttori:

- **ChefPanel(Menu):** Costruttore che crea un oggetto di tipo ChefPanel, caricando un Menu passato come parametro per poi modificarlo.

Parametri:

- **menu:** Il menù del ristorante.
- **menuPanel:** Il pannello in cui sono visualizzate le pietanze del menu, con il loro nome e costo.

Metodi:

- **addMenuScope():** Aggiunge una nuova MenuScope all'interno del menù, restituità da un MenuScopeDialog.

Classi Interne:

- **MenuScopePanel**
  Estende JPanel.
  Classe interna utilizzata per la visualizzazione di una portata del menù, con il suo nome ed il suo costo.

  Costruttori:

  - **MenuScopePanel(MenuScope)** Crea un MenuPanel per la visualizzazzione della portata rappresentata dal parametro MenuScope

  Parametri:

  - **scope:** La MenuScope visualizzata all'interno del pannello.

  Metodi:

  - **getFormattedName(String):** Restituisce la stringa passata come parametro, formattata in modo da rientrare in una larghezza di 16 caratteri.

- **MenuPanel**
  Estende JPanel. Classe interna utilizzata per contenere e visualizzare una lista di MenuScopePanel, caricati dal menu. **Costruttori**:

  - **MenuPanel()** Crea un oggetto MenuPanel per la visualizzazzione di una lista di MenuScopePanel presi dal menu.

  Metodi:

  - **addMenuScope(MenuScope):** Aggiunge una MenuScopePanel con portata la MenuScope passata come parametro.
  - **updateMenuScope(MenuScope):** Aggiorna la MenuScopePanel con portata la MenuScope passata come parametro.
  - **removeMenuScope(MenuScope):** Rimuove dal MenuPanel la MenuScopePanel con portata la MenuScope passata come parametro.

2.7 WaiterPanel

Estende AbstractPanel.
Questa Classe rappresenta il menù del Cameriere. Da qui è possibile creare e inviare ordini alla Cucina.

Costruttori:

- **WaiterPanel(Menu):** Costruttore che crea un oggetto di tipo WaiterPanel, con **menu** il Menu passato come parametro.

Parametri:

- **menu:** Il menu utilizzato dal cameriere per gli ordini.

- **currentOrder:** L'ordine corrente che il cameriere sta prendendo.
- **orderPanel:** Pannello per la visualizzazzione dell'ordine corrente.

Metodi:

- **refreshOrderPanel(JScrollPanel, boolean):** Aggiorna il pannello che visualizza l'ordine corrente del cameriere. Usato per cambiare il tipo di visualizzazzione dell'ordine, e per l'aggiornamento dell'ordine.

Classi Interne:

- **OrderPanel**
  Estende JPanel.
  Classe che contiene e visualizza una lista di DishPanel.

  Costruttri:

  - **OrderPanel(boolean):** Crea un OrderPanel con al suo interno i piatti dell'ordine corrente. Se il parametro booleano passato è true, l'OrderPanel visualizzerà tutte le portate del menù. Se il parametro booleano passato è false, l'OrderPanel visualizzerà unicamente le portate del menù aggiunte all'ordine.

  Metodi:

  - **resetOrder():** Metodo che resetta l'ordine, portando tutte le quantità delle portate del menù a zero per l'ordine corrente.

- **DishPanel**
  Estende JPanel.
  Classe usata per la visualizzazzione di un Dish, con il suo nome e la sua quantità.

  Costruttori:

  - **DishPanel(MenuScope):** Crea un nuovo DishPanel a partire da una MenuScope, creando un nuovo Dish con quantità 0 e MenuScope la MenuScope passata come parametro nel costruttore.
  - **DishPanel(Dish):** Crea un nuovo DishPanel che visualizza il Dish passato come parametro.

  Parametri:

  - **dish:** Il Dish visualizzato dal DishPanel

  Metodi:

  - **getFormattedName(String):** Restituisce una Stringa a partire dalla Stringa passata come parametro, formattata per rientrare in una larghezza di 25 caratteri.

2.8 CookPanel

Estende AbstractPanel.
Questa Classe rappresenta il menù del Cuoco. Da qui è possibile visualizzare gli ordini con pietanze ancora non preparate, e notificare la loro preparazione e spedizione al tavolo.

Costruttori:

- **CookPanel(Menu):** Costruttore che crea un oggetto di tipo CookPanel, al quale viene associato un Menu usato per la lettura degli ordini.

Parametri:

- **menu:** Il menu del ristorante usato per la lettura degli ordini.
- **currentOrder:** L'ordine corrente visualizzato dal cuoco.
- **orderPanelScrollPane:** JScrollPanel usato per contenere l'**orderPanel**
- **orderPanel:** Pannello per la visualizzazzione dell'ordine, con le sue portate.

Metodi:

- **refreshOrderPanel():** Aggiorna l'**orderPanel** alla cottura delle pietanze, o al cambio dell'ordine corrente.

Classi Interne:

- **OrderPanel**
  Estende JPanel.
  Classe che contiene e visualizza una lista di DishPanel. **Costruttori**:
  - **OrderPanel():** Crea un OrderPanel con al suo interno i piatti ancora da cucinare dell'ordine corrente.
- **DishPanel**
  Estende JPanel.
  Classe usata per la visualizzazzione di un Dish, con il suo nome e la sua quantità.

  Costruttori:
  - **DishPanel(Dish)** Crea un nuovo DishPanel che visualizza il Dish passato come parametro.

  Metodi:
  - **getFormattedName(String):** Restituisce una Stringa a partire dalla Stringa passata come parametro, formattata per rientrare in una larghezza di 25 caratteri.

2.9 AbstractPanel

Estende JPanel.
Classe astratta che permette di Creare un JPanel con una immagine di background, caricata a partire dal percorso restituito sottoforma di stringa dal metodo astratto **getBackgroundAssets()**.

Costruttori:

- **AbstractPanel():** Crea un oggetto di tipo AbstractPanel, leggendo e salvando all'interno del parametro **backgroundImage** l'immagine ottenuta a partire dal metodo astratto getBackgroundAssets().

Parametri:

- **backgroundImage:** L'immagine di background del pannello.

Metodi:

- **paintComponent():** Metodo riscritto a partire dal metodo paintComponent() della superclasse JPanel. Richiama il metodo del padre, ed in più disegna come sfondo la **backgroundImage** del pannello.
- **getBackgroundAssets():** Metodo astratto che restituisce sottoforma di String il percorso dell'immagine da caricare come **backgroundImage**

2.10 MainFrame

Estende JFrame.
Frame dell'applicazione, che visualizza un AbstractPanel alla volta, e carica il Menu del ristorante a partire da un file. Alla chiusura, il frame aggiorna il parametro lastID del menu, e salva il menu nel file specificato dal costruttore.
Costruttori:

- **MainFrame(File)** Costruttore che crea un oggetto di tipo MainFrame, inizializzando il **content** come MainPanel, e caricando il Menu a partire dal File passato come parametro.

Parametri:

- **content** L'AbstractPanel visualizzato dal MainFrame.
- **menu** Il Menu del ristorante.
- **menuFile** Il file contenente il Menu del ristorante.

Metodi:

- **saveMenu():** Salva il menu del ristorante nel File **menuFile**
- **getMenu():** Restituisce il menu del ristorante.
- **setContent(AbstractPanel):** Setta il parametro **content** della classe all'AbstractPanel passato come parametro. In più, esegue un repaint totale del MainFrame.

2.11 Order

Classe che rappresente un Ordine del ristorante.
Costruttori:

- **Order():** Crea un oggetto di tipo Order, con il numero di tavolo inizializzato a -1, e una lista vuota di piatti.
- **Order(int):** Crea un oggetto di tipo Order, con il numero del tavolo preso dal parametro passato nel costruttore, ed una lista vuota di piatti.
- **Order(int, ArrayList¡Dish¿):** Crea un oggetto di tipo Order, con parametri il numero del tavolo e la lista dei piatti ordinati.

Parametri:

- **tableNumber** Il numero del tavolo associato all'ordine.
- **dishes** Una lista di piatti ordinati dal tavolo.
- **cookedDishes** Una lista dei piatti ordinati dal tavolo, cotti e spediti dalla cucina.

Metodi:

- **add(Dish):** Aggiunge un piatto alla lista di piatti dell'ordine. Se un piatto con la stessa MenuScope è presente, aggiunge la quantita del piatto passato come parametro a quello preesistente nell'ordine.
- **add(MenuScope):** Aggiunge un piatto alla lista con quantità 1 e MenuScope passata come parametro.
- **add(MenuScope, int):** Aggiunge un piatto alla lista con quantità e MenuScope passati come parametri.
- **remove(MenuScope):** Rimuove un piatto da questo ordine con la stessa MenuScope passata come parametro.
- **remove(Dish):** Rimuove il piatto passato come parametro.
- **get(int):** Restituisce un piatto presente nell'ordine a partire dal suo indice nella lista di piatti.
- **get(MenuScope):** Restituisce un piatto presente nell'ordine che ha come MenuScope la MenuScope passata come parametro.
- **toString():** Metodo sovrascritto dalla classe Object per la stampa della classe sottoforma di testo.
- **indexOf(MenuScope):** Restituisce l'indice del piatto nella lista dei piatti con la stessa MenuScope passata come parametro.

- **indexOf(Dish):** Restituisce l'indice del piatto passato come parametro presente nella lista dei piatti.
- **contains(Dish):** Restituisce un valore booleano. True se il piatto passato come parametro è contenuto nella lista dei piatti, false altrimenti.
- **contains(MenuScope):** Restituisce un valore booleano. True se il piatto con MenuScope uguale a quella passata come parametro è contenuto nella lista dei piatti, false altrimenti.
- **entries():** Restituisce la quantità di piatti non cotti contenuti all'interno dell'ordine.
- **getTotal():** Restituisce il costo totale dei piatti cotti all'interno dell'ordine.
- **cook(MenuScope):** Cuoce un piatto all'interno dell'ordine con la MenuScope passata come parametro.
- **cook(Dish):** Cuoce un piatto all'interno dell'ordine uguale al piatto passato come parametro.
- **getTableNumber():** Restituisce il numero del tavolo associato all'ordine.
- **getCookedDishes():** Restituisce la lista di piatti cotti all'interno dell'ordine.
- **setTableNumber(int):** Imposta il numero del tavolo al parametro passato.
- **decrementQuantity(Dish, int):** Riduce la quantità di un piatto con la stessa MenuScope del piatto passato come parametro.
- **decrementQuantity(MenuScope, in):** Riduce la quantità di un piatto con la stessa MenuScope del parametro passato.
- **incrementQuantity(Dish, int):** Aumenta la quantità di un piatto con la stessa MenuScope del piatto passato come parametro.
- **incrementQuantity(MenuScope, int):** Riduce la quantità di un piatto con la stessa MenuScope del parametro passato.
- **getDishes():** Restituisce la lista di piatti non cotti dell'ordine.
- **addCooked(Dish):** Aggiunge alla lista di piatti cotti il piatto passato come parametro.

2.12 MenuUtil

Classe contendente unicamente metodi statici utici per la classe Menu. **Metodi**:

- **saveMenuOnFile(File, Menu):** Salva un Menu all'interno del file passato come parametro.

- **getMaxNameLength(Menu):** Restituisce la lunghezza del nome più lungo delle portate all'interno del Menu.
- **loadMenuFromFile(File):** Restituisce un Menu caricato dal File passato come parametro.

2.13 Dish

Classe Che Rappresenta un Piatto, con la sua Portata e la sua Quantità.
Metodi Statici:

- **toText(Dish):** Restituisce una stringa rappresentante il piatto formata come:
  ID;QUANTITY
- **fromText(String, Menu):** Restituisce una piatto rappresentato dalla Stringa passata come parametro, la cui portatà è all'interno del Menu passato come parametro.

Costruttori:

- **Dish(MenuScope):** Crea un oggetto di tipo Dish a partire dalla MenuScope passata come parametro, con una quantità di 0.
- **Dish(MenuScope, int):** Crea un oggetto di tipo Dish a partire dalla MenuScope passata come parametro, con una quantità definita dal secondo parametro.

Parametri:

- **scope** La portata del menù.
- **quantity** Il numero di portate.

Metodi:

- **toString():** Metodo sovrascritto dalla classe Object per la stampa della classe sottoforma di testo.
- **setQuantity(int):** Setta il parametro **quantity** al valore del parametro passato.
- **getQuantity():** Restituisce il parametro **quantity**.
- **getScope():** Restituisce il parametro **scope**.
- **decrementQuantity(int):** Riduce il parametro **quantity** di una quantià definita dal parametro passato.
- **incrementQuantity(int):** Aumenta il parametro **quantity** di una quantià definita dal parametro passato.

2.14 MenuScope

Classe Che rappresente una Portata all'interno del menu, con il suo ID, il suo nome ed il suo costo.
Metodi Statici:

- **fromText(String):** Restituisce una MenuScope a partire da una Stringa.
- **toText(MenuScope):** Restituisce una Stringa rappresentante la MenuScope.

Costruttori:

- **MenuScope(int, String, int):** Crea una MenuScope a partire dal suo ID, il suo nome ed il suo costo.

Parametri:

- **id:** L'ID della Portata. Univoco all'interno di un Menu.
- **name:** Il nome della portata.
- **cost:** Il costo della portata. Il costo è rappresentato in un numero intero per la velocità dei calcoli, rendendolo un float solo nel momento della visualizzazzione. Questo è possibile poiché la valuta di rappresentazione (il dollaro) non va oltre il centesimo di precisione.

Metodi:

- **equals(Object):** Metodo sovrascritto dalla classe Object per il confronto della classe con un'altra.
- **toString():** Metodo sovrascritto dalla classe Object per la stampa della classe sottoforma di testo.
- **getName():** Restituisce il parametro **name** della portata.
- **getId():** Restituisce il parametro **ID** della portata.
- **setName(String):** Setta il parametro **name** della portata.
- **getCost():** Restituisce il parametro **cost** della portata.
- **setCost(int):** Setta il parametro **cost** della portata.

2.15 Menu

Classe che rappresenta il Menu di un ristorante.
Costruttori:

- **Menu():** Crea un nuovo Menu, con una lista vuota di portate e l'ultimo valido ID settato a 0.

Parametri:

- **menu** La lista delle portate del menu.
- **lastID** L'ID dell'ultima portata aggiunta + 1.

Metodi:

- **add(MenuScope):** Aggiunge al menu una nuova portata.
- **remove(MenuScope):** Rimuove dal menu una portata.
- **toString():** Metodo sovrascritto dalla classe Object per la stampa della classe sottoforma di testo.
- **entries():** Restituisce il numero di portate all'interno del menu.
- **getMenuScope(int):** Restituisce la portata a partire dal suo indice all'interno della lista di portate del menu.
- **setLastID(int):** Setta il parametro **lastID** del menu.
- **getLastID():** Restituisce il parametro **lastID**.
- **getMenu():** Restituisce la lista delle portate del menu.
- **getMenuScopeFromName(String):** Restituisce la portata all'interno del menu con nome uguale alla stringa passata come parametro.
- **getMenuScopeFromID(int):** Restituisce la portata all'interno del menu con nome uguale alla stringa passata come parametro.

2.16 OrderUtil

Classe che contiene unicamente metodi statici per la classe Order. **Parametri**:

- **ORDER_SEPARATOR** Valore della Stringa che separa i piatti cotti da quelli non cotti all'interno dei file rappresentati gli ordini.

Metodi:

- **saveOrder(Order):** Salva un ordine non vuoto all'interno del file "orders/OrderI.txt", dove I è il numero del tavolo associato all'ordine.
- **getActiveOrders(Menu):** Restituisce una lista di Ordini associati al Menu, che non hanno una lista di piatti non cotti vuota.
- **sendOrder(Order, Menu):** Invia un Ordine associato ad un Menu alla cucina.
  Se un ordine è già stato mandato, i due ordini verranno sommati insieme e salvati all'interno del file dell'ordine già esistente.
  Altrimenti crea il file "orders/OrderI.txt", dove I è il numero del tavolo associato all'ordine.

- **availableOrder(Order):** Restituisce un valore booleano che indica se il file di un ordine esiste o meno.
- **loadOrderFromFile(File, Menu):** Carica e restituisce un Ordine associato ad un menu da un File.
- **getOrderFromTable(int, Menu):** Carica e Restituisce un Ordine associato ad un menu a partire dal numero di tavolo.
- **getCompletedOrders(Menu):** Restituisce una lista di ordini che hano almeno 1 piatto cotto al loro interno.
- **stampAndDeleteOrder(Order):** Salva un file rappresentante lo scontrino dell'ordine, ed elimina il file dell'ordine.
- **saveOrderOnFile(File, Order):** Salva all'interno di un file un ordine.
- **getMaxNameLength(Order):** Calcola la lunghezza massima dei nomi delle portate all'interno dell'ordine

2.17 Main

Classe contenente il main del programma e metodi per l'interpretazione delle opzioni.

Metodi:

- **main(String[]):** Il main del programma. Legge le opzioni ricevute da linea di comando, e avvia il programma.
- **getMenuPathFromOptions(String[]):** Metodo che restituisce il Percorso del file del Menu a partire dalle opzioni passate al programma.
  Se non dovesse essere l'opzione "-m/-menu", restituirà null.
- **getOptionValue(String, String[]):** Restituisce il valore dell'opzione passata come primo parametro all'interno degli argomenti ricevuti come secondo parametro. Se l'Opzione non è presente, restituirà null.

3 Funzionalità

- **Creare e Modificare un Menù**
  Il menù è contenuto in un file .txt con il seguente formato:

Listing 1: Menu.txt

```
[lastValidID]
ID;Scope_Name;Scope_Cost
ID;Scope_Name;Scope_Cost
...
...
```

dove [lastValidID] è il prossimo ID che verrà associato all'aggiunta di una nova portata. Questo viene incrementato ad ogni aggiunta di portata, e viene ricalcolato alla chiusura del programma, trovando l'ID più grande assegnato alle portate ed incrementandolo di 1. Successivamente è salvata una lista di portate con i prori ID, Nomi e Costi separati da un ";"
Tramite il menù dello chef, è possibile aggiungere una nuova portata inserendone il nome e il costo all'interno di un JDialog personalizzato.
Ed è anche possibile modificarne/eliminarne una già esistente, cliccando con il tasto destro del mouse e selezionando la voce "Edit"/"Remove" nel Pop-up menù aperto.
Il menù è letto da un file all'avvio del programma, tramite il metodo **loadMenuFromFile()** all'interno della classe **MenuUtil**.
Il file in questione è, di default il file "Menu.txt" all'interno della stessa cartella in cui è situato il programma, altrimenti è il file passato da riga di comando tramite l'opzione "-m/-menu [path]".

- **Creare ed inviare un Ordine**
  Gli ordini sono contenuti all'interno della cartella "orders" all'interno della stessa cartella contenente il programma, e sono salvati in file .txt avendo nome: "OrderI.txt", dove I è il numero del tavolo. Il formato dei file di ordini è il seguente:

Listing 2: Order3.txt

```
[tableNumber]
ID;Scope_Quantity
ID;Scope_Quantity
...
...
==COOKED==
ID;Scope_Quantity
ID;Scope_Quantity
...
...
```

dove [tableNumber] è il numero del tavolo associato all'ordine, seguito da una lista di piatti ordinati ma non ancora cotti, salvati con il proprio ID e la propria quantità. La stringa di separazione "==COOKED==", che divide i piatti non cotti da quelli cotti e spediti al tavolo, con a seguire la lista delle portate cotte e spedite al tavolo con i prori ID e quantità.
Il file è leggibile e convertibile in un oggetto di classe Order tramite il metodo **loadOrderFromFile()** della calsse **OrderUtil**.
Questo oggetto è utilizzato nel Menà del Cameriere, del Cuoco e della Cassa.

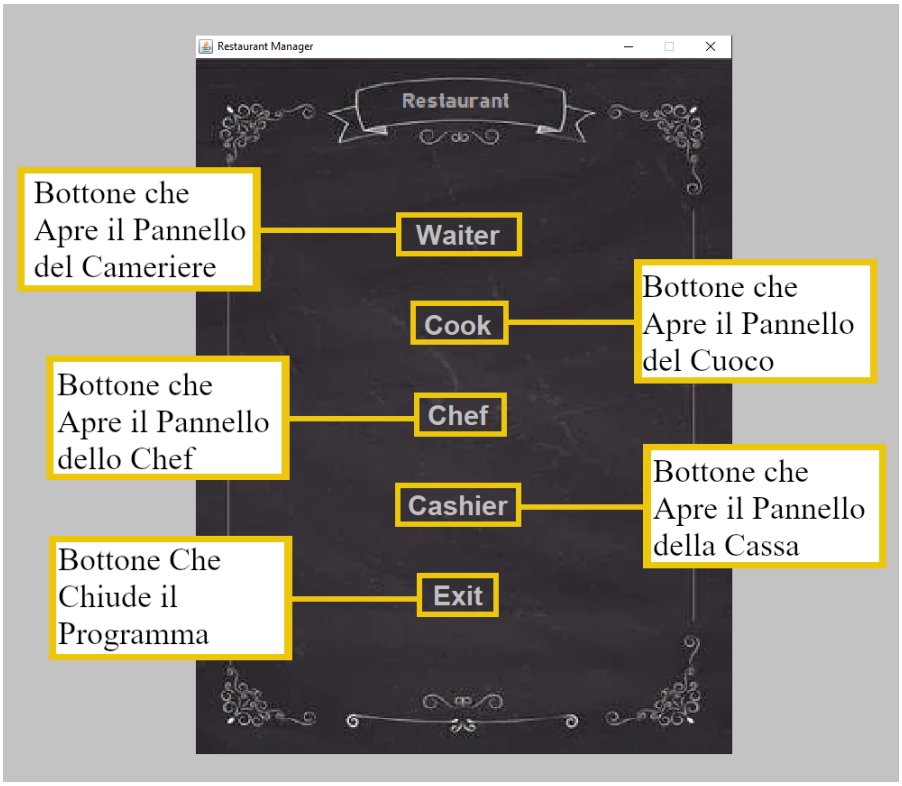
- **Cuocere Portate di un Ordine**
  Questa operazione è svolta dal menù del Cuoco, il quale può visualizzare una delle portate non cucinate di un tavolo, ed in caso notificare l'avvenuta

preparazione.
La classe, tramite il metodo **getActiveOrders()** della class **OrderUtil**, inizializza la lista di tavoli con ordini in sospeso, e permette la loro visualizzazione e modifica tramite cottura delle portate.
Quando una portata viene cotta, la quantità della portata non cotta all'interno dell'ordine viene decrementata di 1, e viene aumentata di 1 la quantità della portata cotta nell'ordine.

- **Pagare l'ordine alla cassa e Stampa Scontrino**
  Questa operazione è svolta dal menù del Cassiere, il quale può visualizzare il conto di un tavolo che abbia consumato almeno 1 portata, e stamparne lo scontrino.
  La classe, tramite il metodo **stampAndDeleteOrder()** della classe **OrderUtil**, inizializza la lista di tavoli con ordini che hanno almeno una pietanza sotto a sezione delle pietanze cotte e spedite. Oltre alla visualizzazione, permette di stampare lo scontrino tramite un pulsante, il quale richiama il metodo della classe , che salva lo scontrino dell'ordine, ed elimina l'ordine corrente dalla cartella "orders".

4 Manuale della GUI

4.1 Menù Principale

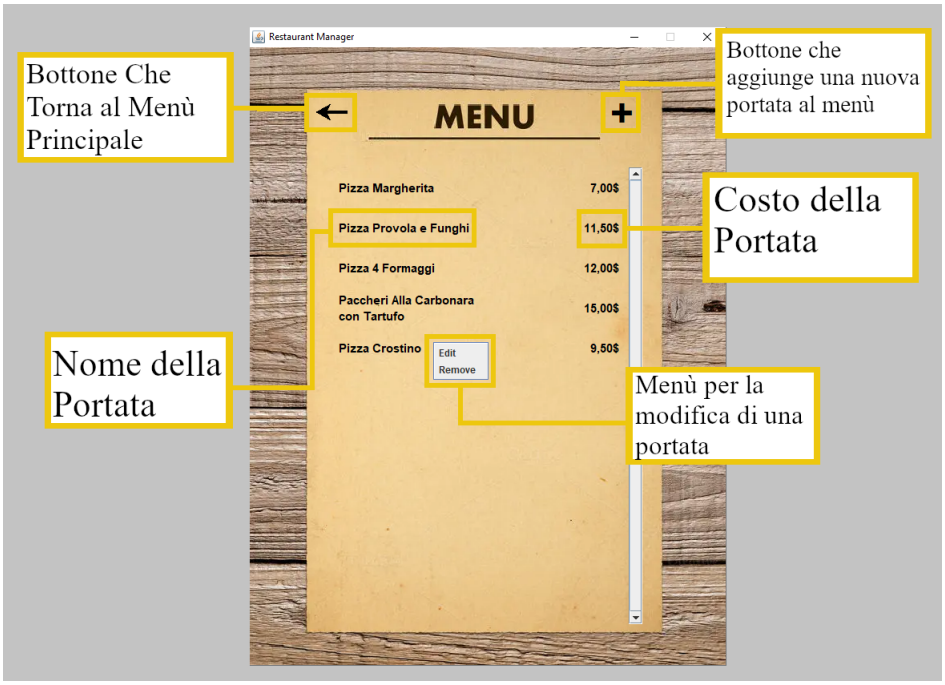


Il menù principale è la schermata visualizzata all'apertura del programma. Da qui è possibile accedere alle diverse schermate del: Cameriere, Cuoco, Chef e Cassa.
Inoltre è possibile chiudere il programma tramite l'apposito bottone "Exit" posizionato in fondo.

4.2 Menù del Cameriere

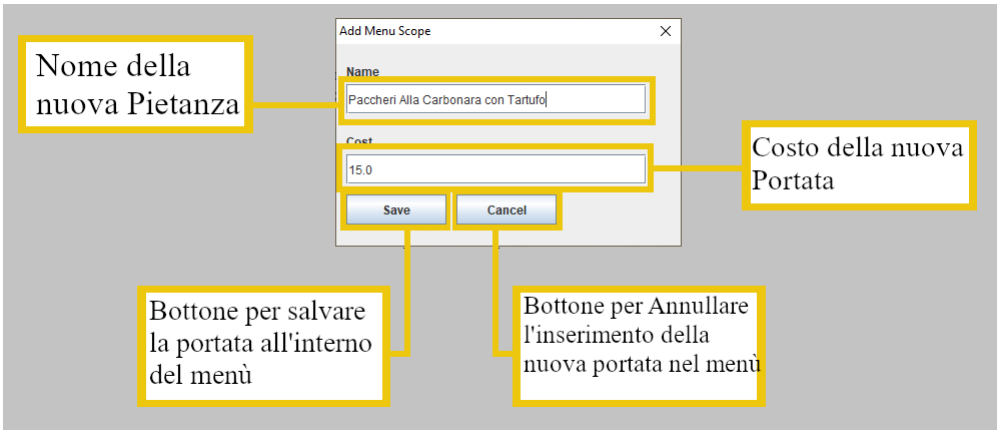
La schermata del Cameriere.
Da qui è possibile selezionare un tavolo, creare un ordine ed inviarlo alla cucina tramite l'apposito bottone posizionato in fondo "Send Order To Kitchen"

4.3 Menù del Cuoco



La schermata del Cuoco.
Da qui è possibile modificare, eliminare e aggiungere portate al menù del ristorante. Cliccando con il tasto destro del mouse su una portata, è possibile aprire un menù pop-up grazie al quale si può modificare o eliminare una portata presente nel menù.

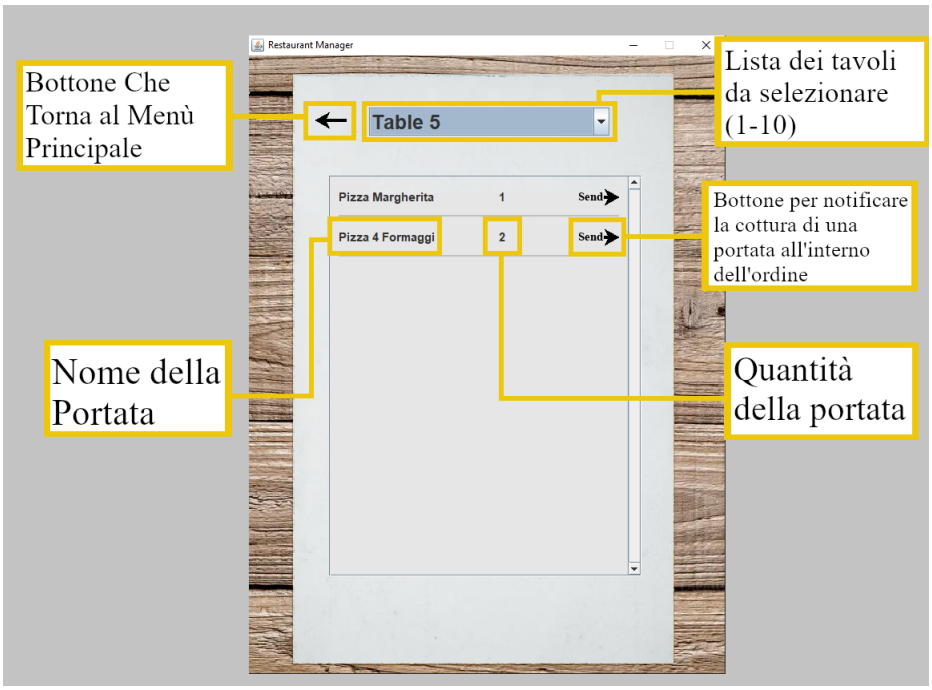
4.3.1 Dialodo di Creazione/Modifica di una portata



Questo menù, accessibile dalla schermata del cuoco, permette di modificare il

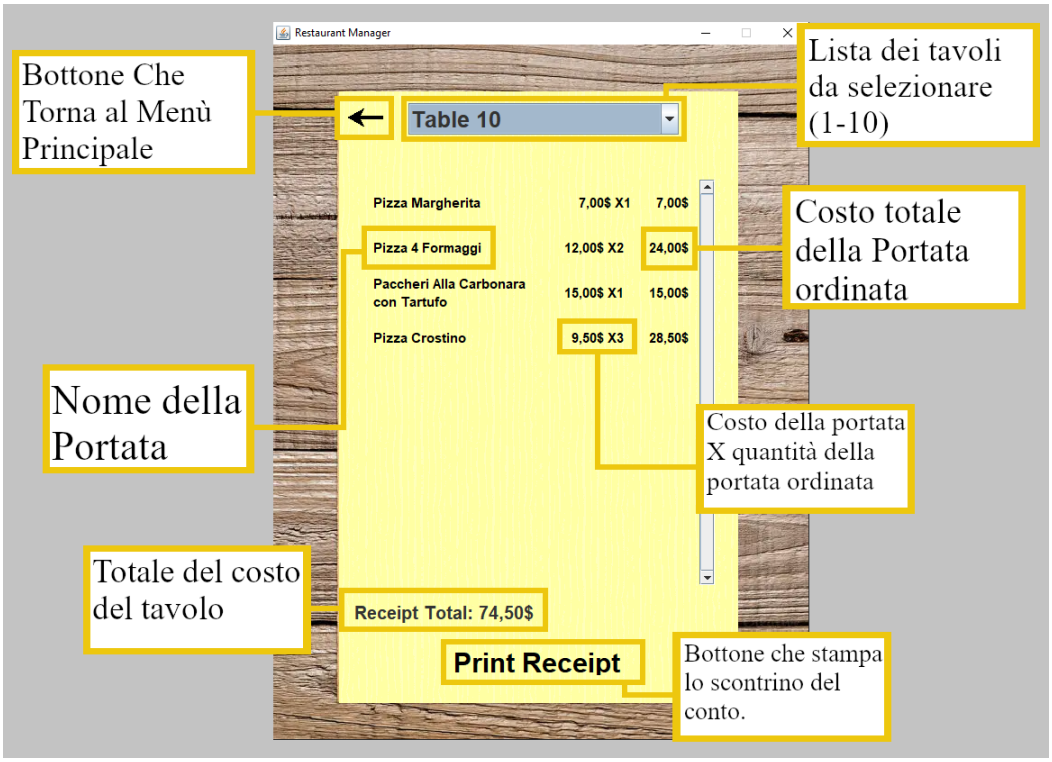
nome e il costo di una portata, o crearne una da zero. Tramite il bottone "Save", è possibile salvare la portata e le sue modifiche all'interno del menù.
Tramite il bottoce "Cancel", è possibile annullare le modifiche/la creazione della portata.

4.4 Menù della Cucina



La schermata della Cucina.
Da qui è possibile visualizzare l'ordine di un tavolo, selezionabile dalla lista posizionata in alto, e notificare la preparazione dei piatti tramite il bottone "Send" posto alla destra delle portate.

4.5 Menù della Cassa



La Schermata della Cassa.
Da qui è possibile visualizzare il conto di un tavolo, selezionabile dalla lista posizionata in alto, e stampare lo scontrino tramite il bottone "Print Receipt" in fondo alla schermata.
Quando viene stampato lo scontrino, l'ordine del tavolo viene cancellato. Dando la possibilità di riutilizzare il tavolo e creare un nuovo ordine.

5 Referenti di sviluppo

L'intero progetto è stato sviluppato da: Simone Di Cesare (1938649)