Giacomo Andrea Rotondo Cocco
X81000984
grotondococco@gmail.com

jHorseBetting

22/06/2020

Requisiti

Il sistema software simula un gioco di scommessa sui cavalli, in particolare l'utente tramite menù di gioco può scegliere di giocare, ricaricare il suo saldo, visualizzare lo storico od uscire dal gioco.

Il menù di gioco permette all'utente di puntare su un cavallo e di scegliere tra due tipi di giocate: scommettere sul primo posto oppure sul podio dei tre cavalli che arrivano per primi al traguardo. L'eventuale vincita è calcolata X3 la puntata se riguarda il primo posto e X2 la puntata se riguarda il podio. La puntata può avvenire di qualsiasi cifra, giacenza sul saldo del giocatore permettendo.

Il menù di scelta per lo storico presenta quattro alternative: ultime cinque scommesse perse, ultime cinque scommesse vinte, ultime 5 ricariche oppure ultimi 10 movimenti sul saldo del gioco tra esiti e ricariche.

Il menù di ricarica permette al giocatore di ricaricare il proprio saldo di gioco di qualsiasi cifra ed il menù di uscita permette, previa conferma, di terminare la sessione di gioco.

Progettazione

Il sistema software si articola utilizzando i design pattern: Singleton, State e Factory Method. Per la gestione del menù è usato il DP State, per regolare l'accesso ai file di testo per il saldo di gioco e per lo storico degli esiti e ricariche il DP Singleton, infine per il salvataggio degli esiti e ricariche è usato il DP Factory Method.

Main (package main)

È la classe che contiene unicamente il metodo necessario all'esecuzione e compilazione del sistema software. Le uniche due istruzioni in esso contenute consistono nell'instanziare un oggetto di tipo Menu e successiva chiamata al metodo *init* della classe Menu.

Menu (package menu)

È la classe che si occupa della stampa del menù. Essa funge da Context per il DP State, in quanto tiene il riferimento ad un oggetto di tipo MenuStatus, sul quale chiama ciclicamente (tramite metodo *init*) i metodi *print* e *changeStatus*

MenuStatus (package menu)

È l'interfaccia che ricopre il ruolo di State per il DP State. Dichiara i metodi *print*, *changeStatus*, *back* e *next*. Nelle rispettive implementazioni di tali metodi vige l'idea di costruire un percorso logico di navigazione all'interno del menù del gioco.

MainMenu (package menu)

È lo stato del menù che gestisce la navigazione sul menù principale, il quale mostra all'utente le funzioni Gioca, Ricarica, Storico ed Exit. Tramite *changeStatus* viene richiamato un ramo condizionale che dirige il prossimo menù da visualizzare sulla funzione scelta dall'utente.

ExitMenu (package menu)

È lo stato del menù che gestisce la terminazione dell'applicativo. L'utente da questo stato può tornare al menù principale oppure decidere di terminare l'applicativo. A scandire questa selezione, il metodo *print* mostra due possibili scelte: y oppure n.

RechargeMenu (package menu)

È lo stato del menù previsto per le ricariche sul saldo di gioco dell'utente. Se l'utente inserisce 0 torna al menù principale, altrimenti la cifra che inserisce verrà inserita sul saldo di gioco. Tale implementazione di MenuStatus ha un metodo aggiuntivo *recharge* per assolvere a tale incarico.

HistoryMenu (package menu)

È lo stato del menù per la visione dello storico delle scommesse e delle ricariche effettuate da parte dell'utente. Il metodo *print* mostra le quattro alternative di visione previste dai requisiti oppure una selezione apposita per tornare al menù principale.

GameMenu (package menu)

È lo stato del menù che mostra la lista dei cavalli sulla quale poter scommettere. L'utente seleziona il cavallo scrivendo su schermo il numero corrispondente al cavallo. Tale implementazione di MenuStatus ha un metodo aggiuntivo *coreGame (int ncav)* per richiamare il prossimo menù e mantenere la selezione.

GameTypeMenu (package menu)

È lo stato del menù che mostra quale tipologia di scommessa l'utente può piazzare. L'utente seleziona il tipo di scommessa scrivendo su schermo l'input corrispondente. Tale implementazione di MenuStatus ha un metodo aggiuntivo *coreGame (int ncav, int gType)* per richiamare il prossimo menù e mantenere le selezioni.

FinalSelectionMenu (package menu)

È lo stato del menù dove l'utente inserisce la puntata. Il metodo *changeStatus* si occupa di controllare se la puntata non supera la giacenza sul saldo. Tale implementazione di MenuStatus ha un metodo aggiuntivo *coreGame (int ncav, int gType, double euroS)* per richiamare il prossimo menù e mantenere le selezioni.

CoreGame (package menu)

È lo stato del menù che mostra all'utente l'esito della gara. Tale classe è corredata di metodi aggiuntivi per comunicare l'esito della scommessa quali *win* e lost. L'ulteriore metodo *updateSaldo* è usato a priori da entrambi per aggiornare il saldo di gioco tramite classe adibita allo scopo.

Cavalli (package game)

È la classe che accede al file .txt contenente l'elenco dei cavalli in gara. Tale classe restituisce tramite metodo statico *getCavalli* la lista dei cavalli in gara. Gli altri due metodi *loadCavalli* e *updateNumCavalli* completano le funzioni della classe per la gestione delle informazioni sui cavalli utili al sistema software.

Core (package game)

È la classe che si occupa di generare gli esiti delle gare. I metodi statici *getEsit1P* e *getEsit3P* restituiscono dei boolean relativi all'esito. La gara è simulata tramite il metodo *newGame* che mescola una lista di numeri interi. Infine, il metodo *gameEsit* restituisce la lista di interi sulla quale si basa l'esito della gara.

Inputchecker (package game)

È una classe che si compone di tre metodi statici adibiti al rilevamento dell'input da parte dell'utente. Il metodo *checkInput* è adibito agli input di caratteri sui menù. Delle selezioni numeriche sui menù se ne occupa *numberInput*, infine per gli input che riguardano il denaro *doubleInput*.

Saldo (package game)

È una classe strutturata seguendo il principio del DP Singleton. Essa si occupa di accedere al file contenente il saldo di gioco e di predisporre le operazioni ad esso correlate *addSaldo* e *subSaldo*. Tale scelta progettuale risiede sull'esigenza del sistema software di monopolizzare la lettura e la scrittura su un singolo file.

ResultHandler (package game)

È un'altra classe Singleton. Essa si occupa di accedere al file contenente i risultati delle scommesse e le ricariche effettuate. L'operazione *addRecord* permette di aggiungere un record al file, mentre gli altri metodi, tramite programmazione funzionale, forniscono l'output per il menù adibito alla visione dello storico

EsitCreator (package game)

È una classe che funge da ConcreteCreator per il DP Factory Method. Tale classe restituisce implementazioni di Esito a seconda del caso. Se si tratta di una ricarica restituirà una nuova istanza di Rec, se si tratta di una scommessa restituirà, a seconda del boolean di esito, un'istanza di Win o Lost.

Esito (package game)

È l'interfaccia che copre il ruolo di Product per il DP Factory Method. Tale interfaccia dichiara i metodi: *recordEsit*, *toString*, *gTypeToString* per fornire, tramite le relative implementazioni, le funzionalità relative alla generazione del record da registrare sul file e del suo conseguente salvataggio.

Win (package game)

È una classe con ruolo ConcreteProduct per il DP Factory Method. Essa gestisce la formattazione della stringa contenente la vincita da inserire sul file dello storico tramite *toString* e *gTypeToString* e predispone l'aggiunta al file tramite metodo *recordEsit*.

Lost (package game)

È una classe con ruolo ConcreteProduct per il DP Factory Method. Essa gestisce in caso di perdita la formattazione della stringa da inserire sul file dello storico tramite *toString* e *gTypeToString* e predispone l'aggiunta al file tramite metodo *recordEsit*.

Rec (package game)

È una classe con ruolo ConcreteProduct per il DP Factory Method. Essa gestisce la creazione di una stringa contenente la causale dell'operazione (Ricarica) e l'importo ricaricato tramite *toString* e *gTypeToString* e predispone l'aggiunta al file tramite metodo *recordEsit*.

Design pattern usati

Design pattern Factory Method

Nome Ruolo	Nome Classe o Interfaccia
Product	Esito
ConcreteCreator	EsitCreator
ConcreteProduct	Win
ConcreteProduct	Lost
ConcreteProduct	Rec

Design pattern State

Nome Ruolo	Nome Classe o Interfaccia
Context	Menu
State	MenuStatus
ConcreteState	MainMenu
ConcreteState	ExitMenu
ConcreteState	RechargeMenu
ConcreteState	HistoryMenu
ConcreteState	GameMenu
ConcreteState	GameTypeMenu
ConcreteState	FinalSelectionMenu
ConcreteState	CoreGame

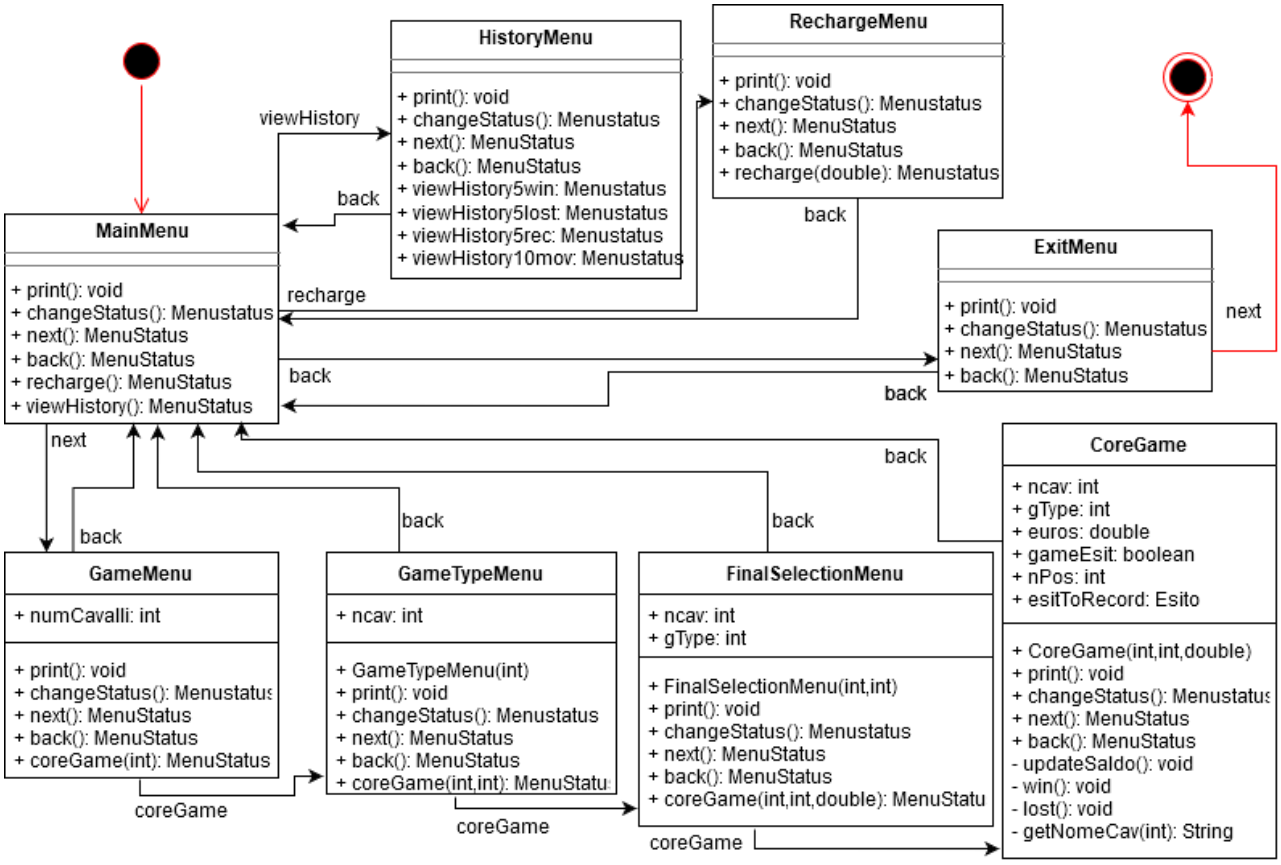
Design pattern Singleton

Nome Ruolo	Nome Classe o Interfaccia
Singleton	Saldo

Design pattern Singleton

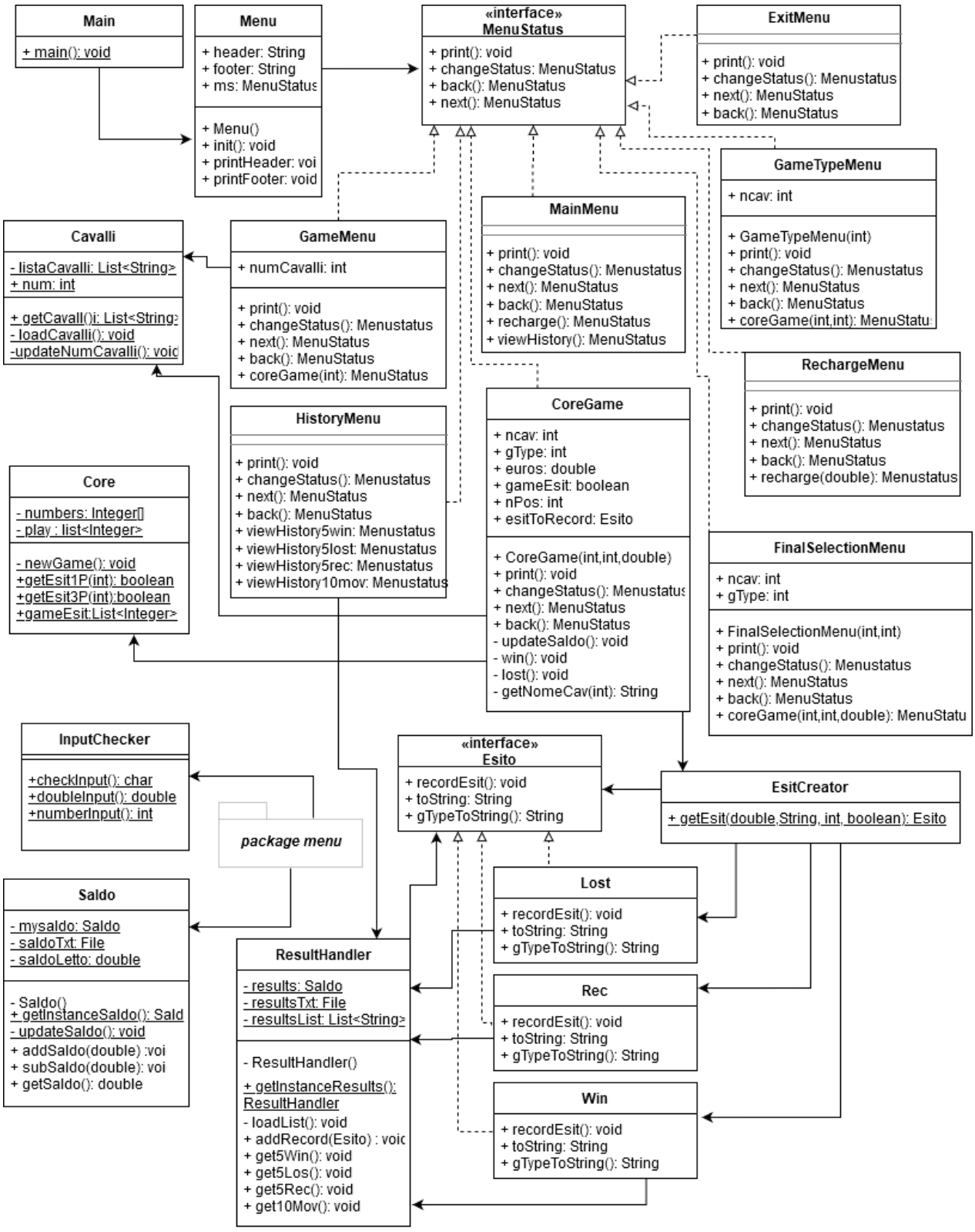
Nome Ruolo	Nome Classe o Interfaccia
Singleton	ResultHandler

Diagramma UML degli stati



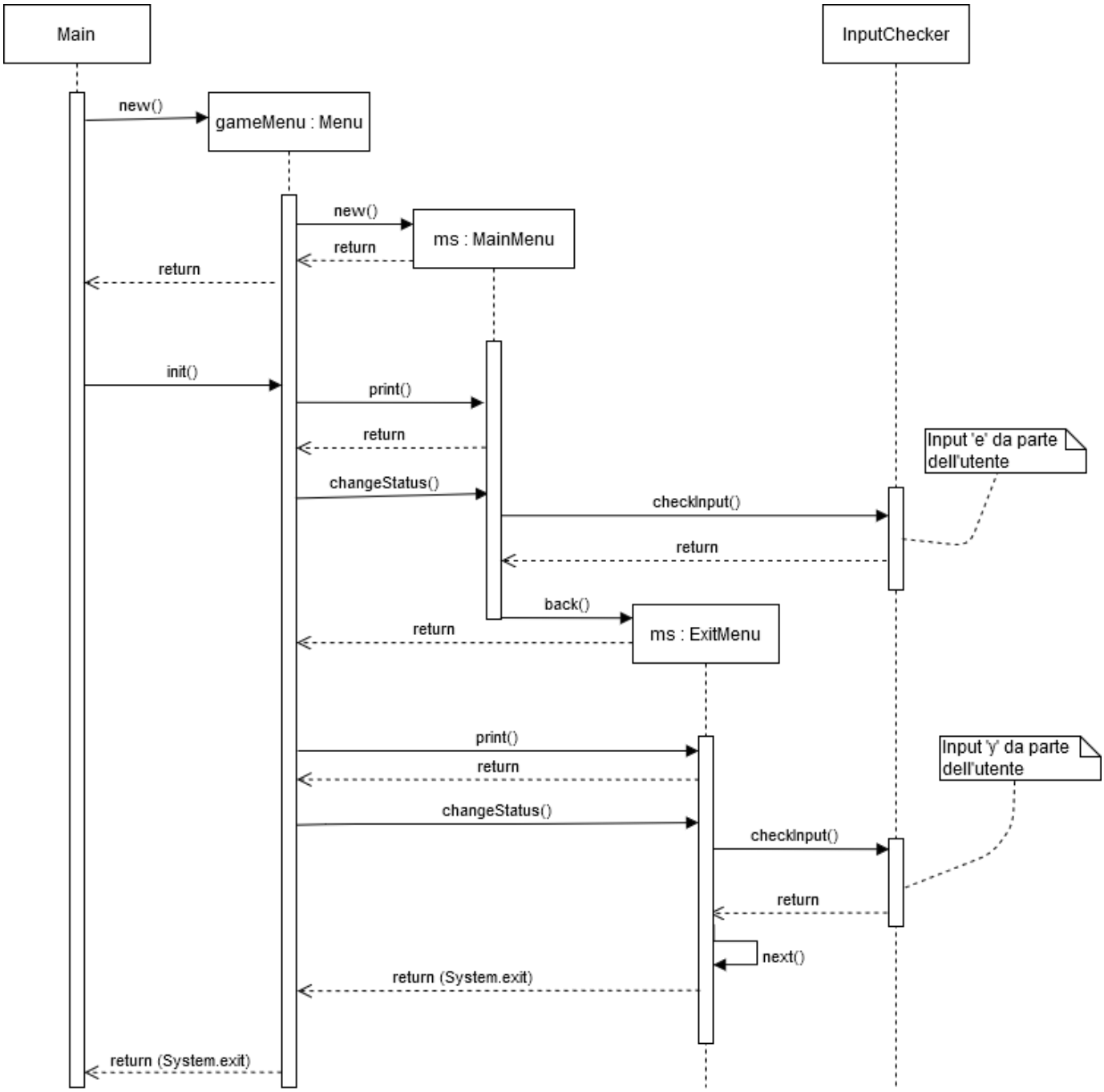
Lo stato iniziale è MainMenu, il quale è raggiungibile da tutti gli altri stati. I metodi *print*, *changeStatus*, *back* e *next* assumono diversi comportamenti a seconda dello stato in cui il Menu si trova. I rami condizionali dovuti all'iterazione con l'utente sono inseriti dentro l'implementazione di *changeStatus* in ciascuna classe, che di conseguenza provvederà a manipolare, con opportune chiamate ai metodi statici di supporto contenuti all'interno della classe *InputCheker*, i dati inseriti dall'utente.

Diagramma UML delle classi



Tutte le implementazioni di MenuStatus fanno uso delle classi *Saldo* e *InputChecker*, le relazioni ad esse sono evidenziate in basso a sinistra del diagramma, per maggiore visibilità. La parte in alto a destra mostra l'implementazione del Design Pattern State, la parte in basso a destra mostra l'implementazione del Design Pattern Factory Method e sulla sinistra sono mostrate le classi di supporto al problema, tra le quali si distinguono *Saldo* e *ResultHandler* poiché seguono le linee guida del Design Pattern Singleton.

Diagramma UML di sequenza



Il Diagramma UML di sequenza mostra l'esecuzione del codice secondo le seguenti condizioni: l'utente avvia il programma e si trova nella schermata iniziale tramite menù principale del sistema software. L'utente inserisce 'e' sulla riga di comando, seguendo le istruzioni per terminare l'applicativo. A seguito viene mostrato il messaggio di conferma, che richiede all'utente di inserire 'y' oppure 'n'. L'utente inserisce 'y' ed il sistema software termina la sua esecuzione.

Link al codice

Repository GitHub: https://github.com/coccojack/jHorseBetting