

Università degli studi di Modena e Reggio Emilia

Facoltà di Ingegneria “Enzo Ferrari”

Corso di laurea in Ingegneria Informatica

A.A. 2020/2021

CARDVENTURE



Pugnaloni Francesco (132159)

"Dichiaro che questo elaborato è frutto del mio personale lavoro, svolto sostanzialmente in maniera individuale e autonoma."

Francesco Pugnaloni.

INDICE

SPECIFICA DEI REQUISITI DEL SOFTWARE(SRS)

1 Introduzione	4
1.1 Obiettivo	4
1.2 Campo di applicazione	4
1.3 Definizioni, acronimi e abbreviazioni	4
1.4 Fonti	4
1.5 Struttura del documento SRS	5
2 Descrizione generale	6
2.1 Inquadramento	6
2.1.1 Interfaccia sistema/utente	6
2.1.2 Interfaccia hardware	6
2.1.3 Interfaccia software	6
2.1.4 Interfaccia di comunicazione (N/A)	7
2.1.5 Vincoli relativi all'occupazione di memoria	7
2.1.6 Operazioni(N/A)	7
2.1.7 Vincoli per l'installazione	7
2.2 Macro funzionalità	7
2.3 Caratteristiche degli utenti	7
2.4 Vincoli generali	7
2.5 Ipotesi di partenza, assunzioni e dipendenze	7
2.6 Requisiti da analizzare in futuro(N/A)	7
3 Specifica dei requisiti	8
3.1 Requisiti funzionali	8
3.1.1 Schermata iniziale	8
3.1.2 Menu principale	9
3.1.3 Selezione del personaggio	10
3.1.4 Pannello di gioco	10
3.1.5 Classifica	14
3.1.6 Tutorial	15
3.1.7 Requisiti generali	15
3.1 Requisiti non funzionali	16
4 Appendici (N/A)	16

Unified Modelling Language (UML)

5 Introduzione	—17
6 Use case diagram	—17
7 Activity diagram	—18
7.1 Partita	—18
7.2 Visualizzazione ed eliminazione punteggi	—18
7.3 Visualizzazione del tutorial	—19
8 State diagram	—19
8.1 Partita	—19
8.2 Personaggio	—20
9 Package diagram	—20
10 Class diagram	—22
10.1 Gestione delle carte (pattern “Strategy”)	—23
10.2 Gestione del Dealer (pattern “Factory method”)	—23
10.3 Gestione della partita e selezione del personaggio (tramite classe “factory”)	—24
10.4 Gestione generale	—25
11 Sequence diagram	—26
11.1 Creazione di una nuova partita	—26
11.2 Istanziamento di una nuova carta	—27
11.3 Movimento del personaggio	—27
11.4 Selezione del personaggio	—28
11.5 Salvataggio del punteggio	—28

SRS

1 Introduzione

La presente sezione ha lo scopo di riportare la visione globale dell'intero documento di specifica dei requisiti. La struttura del documento è quella suggerita dallo standard ANSI/IEEE830 noto come SRS (Software Requirements Specification).

1.1 Obiettivo

Questo documento ha lo scopo di spiegare, nel modo più preciso, consistente, non ambiguo e comprensibile, in modo che siano chiari sia al committente che ai progettisti, le funzioni del software richieste per la realizzazione di un videogioco 2D, nello specifico un gioco da tavolo survival i cui elementi saranno delle carte da gioco (le regole del gioco saranno introdotte nella sezione 2.1)

Va inoltre tenuto presente che l'approccio seguito per lo sviluppo del software è di tipo prototipale, quindi nuovi requisiti potranno essere introdotti in seguito, oltre a tutti i vincoli che fino a questo momento non sono stati individuati.

1.2 Campo di applicazione

Per il superamento dell'esame di "programmazione a oggetti" è richiesto presentare in sede di orale una applicazione funzionante per dimostrare di aver compreso i principi dell'OOP, non ci sono vincoli sulla tipologia di app da realizzare ed è stato scelto di optare per un videogioco di carattere puramente ludico che oltre a soddisfare i requisiti per il superamento dell'esame possa fungere anche da intrattenimento per un'utenza ristretta, in questo documento si farà infatti riferimento ad un'utenza generica (per le caratteristiche degli utenti vedere punto 2.3).

1.3 Definizioni, acronimi e abbreviazioni

Videogioco 2D	Videogioco sviluppato attraverso tecniche di visione bidimensionale.
Gioco da tavolo survival	Generalmente indica giochi da tavolo in cui lo scopo è sopravvivere il più a lungo possibile per poter ottenere un punteggio sempre più alto. Il software da realizzare dovrà emulare queste caratteristiche.
OOP	Abbreviazione per indicare "Object Oriented Programming", la programmazione ad oggetti.
APP	Abbreviazione di "applicazione"

1.4 Fonti

I requisiti che saranno di seguito illustrati sono stati ricavati dalle fonti seguenti:

- Documentazione che indica i vincoli per la realizzazione del progetto fornita dal docente del corso di "programmazione a oggetti"
- Partecipanti al progetto

1.5 Struttura del documento

Il documento prosegue con i capitoli seguenti:

- CAPITOLO 2: Sarà finalizzato a fornire informazioni di carattere generale sul sistema da realizzare, in particolare per prima cosa saranno trattate le caratteristiche generali che dovranno avere le varie interfacce, per poi passare ad altri argomenti come le caratteristiche dell'utenza.
- CAPITOLO 3: Sarà dedicato a rappresentare le specifiche funzionali e non funzionali dell'applicazione da produrre. Saranno perciò trattate le figure coinvolte nel sistema, i dati ad esse associati e le operazioni che possono compiere.

A seguire saranno inseriti ulteriori capitoli finalizzati a fornire una descrizione del design dell'applicazione tramite vari diagrammi UML.

2 Descrizione Generale

In questo capitolo andremo a descrivere le caratteristiche principali che riguardano l'app da produrre

2.1 Inquadramento

“Cardventure” sarà un software finalizzato ad emulare un gioco di carte survival in solitaria e nasce per finalità accademiche; nonostante questo sarà importante realizzare un’applicazione in grado di fornire intrattenimento ludico facilmente accessibile da una qualsiasi utenza.

Viste le caratteristiche inusuali del gioco che dovrà essere creato saranno qui brevemente spiegate le sue regole:

- Il tavolo da gioco sarà sempre composto da 9 carte poste in una matrice 3x3, una di queste sarà la carta “personaggio”, che rappresenta il giocatore ed ha 2 valori numerici indicati: punti vita e durabilità dell’arma.
- Ogni turno di gioco avrà una struttura molto semplice, si baserà sul muovere la carta “personaggio” in una delle caselle adiacenti, successivamente in base alla carta presente nella casella in cui ci si sposta si potranno avere diverse conseguenze (perdita punti vita, se è una carta “trappola”, o combattimento se si tratta di una carta “mostro” per esempio), nella casella lasciata vuota sarà poi inserita una nuova carta casuale.
- Lo scopo del gioco è effettuare più mosse possibili e la partita finisce quando i punti vita del personaggio diventano minori o uguali zero.

2.1.1 Interfaccia sistema/utente

Poiché si prevede un’utenza eterogenea e non necessariamente esperta in campo informatico il software dovrà avere le seguenti caratteristiche:

- **SEMPLICITÀ DELL’INTERFACCIA UTENTE**

In virtù del target variegato dell’app sarà necessario progettare un’interfaccia molto semplice e intuitiva, ma che sia allo stesso tempo accattivante, avendo il prodotto finalità ludiche.

- **USABILITÀ**

Per rendere il prodotto il più versatile possibile va creata un’interfaccia minimale che fornisca l’accesso alle funzionalità in modo rapido e intuitivo; a questo scopo dovrà essere presente la possibilità di accedere rapidamente ad un tutorial che possa guidare l’utente nell’utilizzo dell’app.

Le varie funzionalità dovranno essere accessibili dal menu principale, il quale potrà ad esempio essere composto di un insieme di bottoni premibili che indirizzano l’utente alle varie funzionalità in modo semplice. Si riterrà raggiunto un livello di usabilità sufficiente se un utente privo di particolari conoscenze informatiche sarà in grado di completare con successo una partita senza aiuti esterni.

- **NAVIGABILITÀ**

Per rispettare questo requisito è necessario che l’utente possa raggiungere tutte le funzionalità del software direttamente dal menu principale e senza dover esplorare sottomenu che potrebbero disorientare chi è privo di competenze informatiche.

2.1.2 Interfaccia hardware

Per rendere l’utilizzo del gioco il più semplice possibile l’unico controller richiesto sarà il mouse, non dovranno inoltre servire particolari livelli di prestazioni alla macchina su cui gira l’applicazione.

2.1.3 Interfaccia software

Il software dovrà essere eseguibile su qualsiasi sistema operativo con installata la corretta versione di java.

2.1.4 Interfaccia di comunicazione(N/A)

2.1.5 Vincoli relativi all'occupazione di memoria

Il software dovrà essere il più leggero possibile in modo da poterlo utilizzare su qualsiasi macchina.

2.1.6 Operazioni(N/A)

2.1.7 Vincoli per installazione

Dovrà essere semplice e limitata all'estrazione da un file compresso dell'applicazione già funzionante.

2.2 Macro funzionalità

Saranno:

- Creazione di una nuova partita
- Selezione del personaggio
- Visualizzazione dei punteggi migliori
- Inserimento di un nickname per identificare i punteggi
- Tutorial

Le varie funzionalità saranno meglio specificate nel paragrafo dedicato ai requisiti funzionali.

2.3 Caratteristiche degli utenti

Come già detto in precedenza l'utenza prevista sarà molto eterogenea e potrà avere competenze informatiche più o meno solide, ma tutti gli utilizzatori dell'app ricadranno nella stessa categoria:

- **Player:** sarà il giocatore, esso ha la possibilità di iniziare una partita e utilizzare le varie funzionalità offerte dall'applicazione, ma non può alterarla in nessun modo.

NOTA: da qui in avanti i termini player e utente sono da considerare equivalenti

2.4 Vincoli generali

Come già detto, è importante che un qualsiasi utente sia in grado di eseguire tutte le operazioni di base, le quali ovviamente non dovranno consentire di modificare dati critici che potrebbero compromettere il funzionamento dell'app.

2.5 Ipotesi di partenza, assunzioni e dipendenze

- Gli utenti potrebbero non avere un buon background informatico.

2.6 Requisiti da analizzare in futuro(N/A)

3 Specifica dei requisiti

3.1 Requisiti funzionali

Questa sezione sarà divisa in 7 parti (corrispondenti a varie sezioni dell'applicazione):

1. Schermata iniziale
2. Menu principale
3. Selezione del personaggio
4. Pannello di gioco
5. Classifica
6. Tutorial
7. Requisiti generali (si intendono le funzionalità che si riferiscono a più di una delle altre parti se non a tutte)

Questa divisione è stata fatta per esporre i vari requisiti nel modo più ordinato possibile; ogni sottosezione raccoglierà i vari requisiti funzionali associati alla macro funzionalità del sistema; alcune di esse saranno inoltre corredate da un'immagine che indica un possibile scheletro di un'interfaccia grafica che soddisfa i vari requisiti introdotti, inserita in virtù dell'approccio prototipale dato al documento.

Va inoltre detto che ogni requisito sarà introdotto tramite una tabella con la struttura seguente:

Codice requisito	Parte a cui appartiene	Nome del requisito
Input	Input atteso	
Processo	Processo da seguire	
Output	Output atteso	

3.1.1 Schermata iniziale

RF01	Schermata iniziale	Visualizza schermata iniziale
Input	Apertura dell'app	
Processo	L'utente apre l'applicazione, verrà visualizzata una schermata accattivante che dovrà stimolare il suo interesse verso il gioco	
Output	Viene aperta la schermata iniziale	

RF02	Schermata iniziale	Passa al menu principale
Input	Mouse	
Processo	L'utente effettua un click in una qualsiasi parte della schermata per passare al menu principale	
Output	La schermata viene cambiata e si passa al menu principale	

Per questa sottosezione non sarà fornita un'immagine prototipale poiché l'interfaccia grafica dovrà limitarsi ad uno sfondo accattivante corredata da una scritta che invita ad effettuare un click in un punto a caso dello schermo.

3.1.2 Menu principale

RF03	Menu principale	Inizio nuova partita
Input	Mouse	
Processo	L'utente seleziona "nuova partita" per cominciare a giocare	
Output	La schermata viene cambiata e si passa alla selezione del personaggio	

RF04	Menu principale	Visualizza high score
Input	Mouse	
Processo	L'utente seleziona "high score" per visualizzare i punteggi migliori	
Output	La schermata viene cambiata e si passa alla classifica	

RF05	Menu principale	Tutorial
Input	Mouse	
Processo	L'utente seleziona "tutorial" per ricevere consigli su come utilizzare l'app	
Output	La schermata viene cambiata e si passa al tutorial	

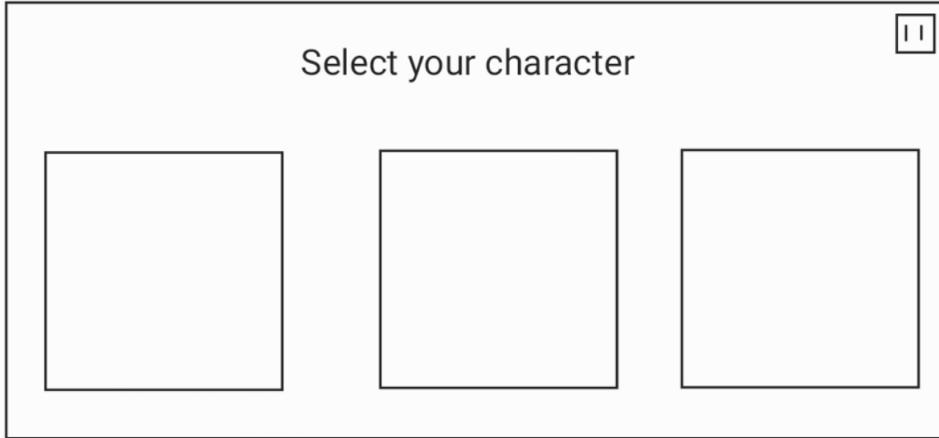
E' qui riportata una possibile interfaccia che soddisfa i requisiti, va fatto notare che, come in tutte le immagini che saranno mostrate in seguito, è stato inserito in alto a destra un pulsante "pausa", che serve a fermare la musica e risponde ad un requisito che sarà introdotto nella sezione 3.1.7.



3.1.3 Selezione del personaggio

RF06	Selezione del personaggio	Scegli personaggio
Input	Mouse	
Processo	L'utente sceglie un personaggio fra quelli disponibili, ognuno dei quali avrà delle caratteristiche specifiche di punti vita e attacco	
Output	Si passa alla schermata di gioco	

E' qui riportata una possibile interfaccia che soddisfa i requisiti (il requisito in questo caso)



I rettangoli in figura rappresentano delle carte da gioco, che saranno ovviamente i vari personaggi fra cui scegliere.

3.1.4 Pannello di gioco

RF07	Pannello di gioco	Visualizza mosse possibili
Input	N/A	
Processo	Finché la partita è in corso saranno sempre visualizzati in qualche modo i movimenti possibili per il proprio personaggio, che potranno essere: <ul style="list-style-type: none"> • moveUp: movimento verso l'alto, è possibile se nella griglia di 9 carte c'è almeno una carta al di sopra della carta personaggio • moveDown: movimento verso il basso, è possibile se nella griglia di 9 carte c'è almeno una carta al di sotto della carta personaggio • moveRight: movimento verso destra, è possibile se nella griglia di 9 carte c'è almeno una carta alla destra della carta personaggio • moveLeft: movimento verso sinistra, è possibile se nella griglia di 9 carte c'è almeno una carta alla sinistra della carta personaggio 	
Output	Le caselle in cui ci si può spostare sono in qualche modo evidenziate	

RF08	Pannello di gioco	moveUp
Input	Mouse	
Processo	L'utente effettua un click sulla casella al di sopra della carta personaggio	
Output	Se la mossa è fra quelle possibili il personaggio viene spostato nella casella selezionata, altrimenti non succede nulla	

RF09	Pannello di gioco	moveDown
Input	Mouse	
Processo	L'utente effettua un click sulla casella al di sotto della carta personaggio	
Output	Se la mossa è fra quelle possibili il personaggio viene spostato nella casella selezionata, altrimenti non succede nulla	

RF10	Pannello di gioco	moveRight
Input	Mouse	
Processo	L'utente effettua un click sulla casella alla destra della carta personaggio	
Output	Se la mossa è fra quelle possibili il personaggio viene spostato nella casella selezionata, altrimenti non succede nulla	

RF11	Pannello di gioco	moveLeft
Input	Mouse	
Processo	L'utente effettua un click sulla casella alla sinistra della carta personaggio	
Output	Se la mossa è fra quelle possibili il personaggio viene spostato nella casella selezionata, altrimenti non succede nulla	

RF12	Pannello di gioco	Aggiornamento punteggio
Input	Movimento personaggio	
Processo	Il personaggio si muove	
Output	Il punteggio viene incrementato di 1	

RF13	Pannello di gioco	Visualizza stato del personaggio
Input	N/A	
Processo	Per tutta la durata della partita dovrà essere sempre visualizzato lo status del personaggio	
Output	Vengono visualizzati i punti vita del personaggio e la durabilità della sua arma	

RF14	Pannello di gioco	Visualizza punteggio
Input	N/A	
Processo	Per tutta la durata della partita sarà sempre visualizzato il punteggio attuale	
Output	Viene visualizzato il punteggio (numero di mosse fatte)	

RF15	Pannello di gioco	Raccogli arma
Input	Movimento in una casella dove è presente una carta "arma"	
Processo	L'utente muove il personaggio in una casella in cui è presente una carta "arma", che avrà indicato un proprio punteggio di durabilità	
Output	La durabilità dell'arma del personaggio viene aggiornata e diventa quella indicata nella carta arma	

RF16		Pannello di gioco	Combattimento
Input	Movimento in una casella dove è presente una carta "mostro"		
Processo	L'utente muove il personaggio in una casella in cui è presente una carta "mostro", che avrà indicato un proprio punteggio di punti vita		
Output	<p>Ci sono 2 situazioni possibili:</p> <ul style="list-style-type: none"> • La durabilità dell'arma del personaggio è maggiore o uguale ai punti vita del mostro: la durabilità dell'arma del personaggio è decrementata dei punti vita del mostro • La durabilità dell'arma del personaggio è minore ai punti vita del mostro: La durabilità dell'arma del personaggio è posta a 0 e i suoi punti vita sono decrementati della differenza fra i punti vita del mostro e la durabilità che la sua arma aveva, se i suoi punti vita diventeranno <= 0 la partita finirà 		

RF17		Pannello di gioco	Bevi pozione punti vita
Input	Movimento in una casella dove è presente una carta "Pozione punti vita"		
Processo	L'utente muove il personaggio in una casella in cui è presente una carta "Pozione punti vita", che avrà indicato un proprio punteggio di punti vita		
Output	I punti vita del personaggio sono incrementati del numero indicato nella carta "pozione punti vita"		

RF18		Pannello di gioco	Usa pozione rinforzo arma
Input	Movimento in una casella dove è presente una carta "Pozione rinforzo arma"		
Processo	L'utente muove il personaggio in una casella in cui è presente una carta "Pozione rinforzo arma", che avrà indicato un proprio punteggio di durabilità		
Output	La durabilità dell'arma del personaggio è incrementata del numero indicato nella carta "pozione rinforzo arma"		

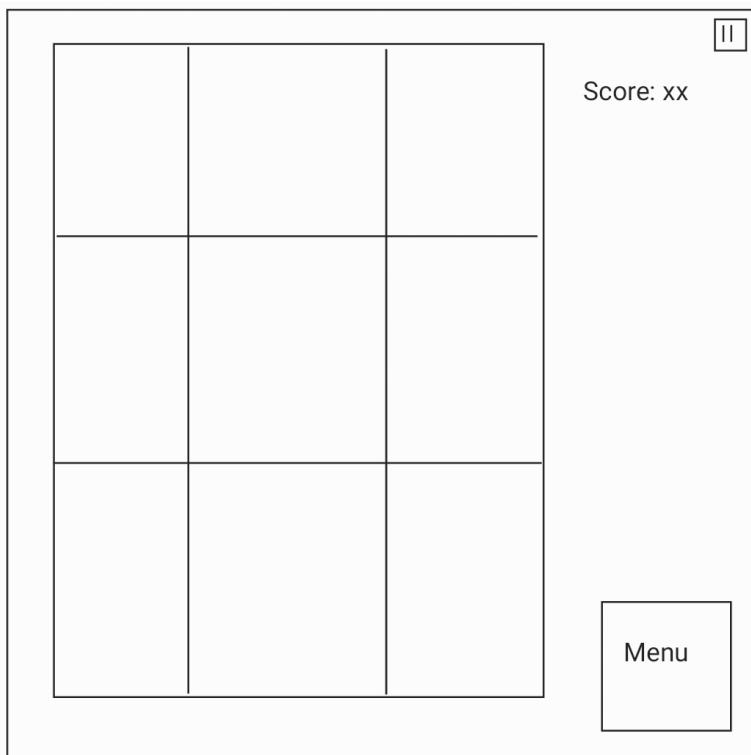
RF19		Pannello di gioco	Danneggiamento da trappola
Input	Movimento in una casella dove è presente una carta "Trappola"		
Processo	L'utente muove il personaggio in una casella in cui è presente una carta "Trappola", che avrà indicato un proprio punteggio di punti vita		
Output	I punti vita del personaggio sono decrementati del numero indicato nella carta "trappola", se in seguito a questa interazione i punti vita diventano minori o uguali a 0 le conseguenze sono le stesse del combattimento con un mostro		

RF20		Pannello di gioco	Pesca carta
Input	Movimento personaggio		
Processo	Quando il personaggio effettua un movimento viene rimossa dal tavolo da gioco la carta presente nella casella in cui si sposta e si hanno 8 carte in campo; per ritornare al numero corretto viene pescata una carta casuale fra quelle possibili e inserita nella posizione rimasta scoperta		
Output	Inserisci la nuova carta nel tavolo da gioco		

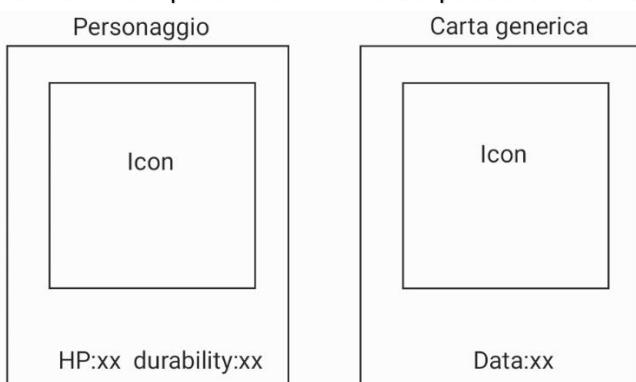
RF21		Pannello di gioco	Fine partita
Input	Punti vita del personaggio <= 0		
Processo	Il personaggio in seguito ad un combattimento o ad una trappola ha perso tutti i suoi punti vita e la partita finisce		
Output	Si passa alla schermata di salvataggio del punteggio		

RF22		Pannello di gioco	Salvataggio punteggio
Input	Username		
Processo	Va fornito lo username a cui associare il punteggio ottenuto nella partita appena conclusa		
Output	Il punteggio viene salvato e si ritorna al menu principale		

E' qui riportata una possibile interfaccia che soddisfa i requisiti, va fatto notare che in questo esempio è presente anche un tasto "menu", che soddisfa un requisito che sarà introdotto nella sezione 3.1.7, la sua funzione sarà di riportare al menu principale.



Mentre un esempio della struttura che si potrebbe dare alle carte da gioco è il seguente



Con "data" si intende un generico campo dati, poiché le carte diverse dal personaggio ne avranno solo uno.

3.1.5 Classifica

RF23	Classifica	Visualizza punteggi
Input	Apertura della sezione “classifica”	
Processo	L’utente entra nella suddetta parte dell’applicazione	
Output	Viene visualizzata una classifica dei punteggi migliori	

RF24	Classifica	Cancella punteggi
Input	Mouse	
Processo	L’utente preme il pulsante “reset scores”	
Output	I punteggi salvati sono eliminati	

E’ qui riportata una possibile interfaccia che soddisfa i requisiti:



3.1.6 Tutorial

RF25	Tutorial	Visualizza tutorial
Input	Ingresso nella sezione “tutorial” dell’app	
Processo	L’utente seleziona il tutorial	
Output	Il tutorial è visualizzato	

Per questa sezione non sarà fornita un’interfaccia di esempio poiché sarà ritenuta accettabile qualsiasi interfaccia sviluppata, purché sia chiara e facilmente navigabile.

3.1.7 Requisiti generali

RF26	Requisiti generali	Metti musica in pausa/accendi musica
Input	Mouse	
Processo	L’utente preme un bottone che svolge la funzione desiderata	
Output	2 situazioni possibili: <ul style="list-style-type: none">• Se la musica è accesa viene messa in pausa• Se la musica è in pausa viene accesa	

NOTA: il requisito RF26 dovrà essere implementato in tutte le schermate dell’applicazione

NOTA: all’apertura dell’app la musica dovrà essere accesa

RF27	Requisiti generali	Torna al menu principale
Input	Mouse	
Processo	L’utente preme un bottone che svolge la funzione desiderata	
Output	Si passa alla schermata del menu principale	

NOTA: il requisito RF27 dovrà essere implementato in tutte le schermate dell’applicazione fatta eccezione per quella di selezione personaggio, la schermata principale e il menu principale stesso.

3.2 Requisiti non funzionali

In questa sezione saranno sintetizzate le caratteristiche non funzionali emerse dalle direttive del docente e dai vari incontri fra i partecipanti al progetto, queste caratteristiche ispireranno lo stile di realizzazione dell'applicazione

Va inoltre detto che i vari requisiti non funzionali saranno introdotti tramite una tabella con la struttura seguente:

Codice requisito	Categoria di appartenenza	Nome del requisito
Descrizione	Descrizione del requisito	

Non sarà inoltre fatta una distinzione in categorie "stretta" come lo è stato per i requisiti funzionali, poiché il numero di requisiti non funzionali emersi è inferiore e di natura più eterogenea.

RN01	Codice e linguaggio	Linguaggio usato
Descrizione	E' richiesto di realizzare l'applicazione utilizzando il linguaggio java	

RN02	Codice e linguaggio	LOC (lines of code)
Descrizione	E' richiesto di realizzare un'applicazione di dimensione discreta (fra le 2000 e le 5000 righe di codice) per dimostrare la comprensione della programmazione ad oggetti	

RN03	Salvataggio dati	Database usato
Descrizione	Non è necessario rendere i dati disponibili in rete, sarà quindi sufficiente un qualsiasi database che li mantiene salvati in locale	

RN04	Requisiti prestazionali	Tempi di risposta
Descrizione	L'applicazione deve rispondere prontamente ai vari input da parte dell'utente per rendere il più piacevole possibile l'esperienza di gioco	

RN05	Attributi del sistema	Mantenibilità
Descrizione	E' necessario seguire la politica di individuazione degli elementi critici, che devono essere trattati in modo adeguato in modo da poter estendere l'applicazione, ovvero: <ul style="list-style-type: none"> • Tipologie di nemici • Tipologie di trappole • Tipologie di pozioni • Tipologie di armi • Tipologie di carte in generale (in futuro si potrebbe voler aggiungere per esempio la tipologia "animale", che avrebbe un comportamento diverso dalle carte finora presenti) 	

RN06	Attributi del sistema	Portabilità
Descrizione	L'applicazione deve poter funzionare in qualsiasi sistema provvisto della corretta versione di java	

4 Appendici (N/A)

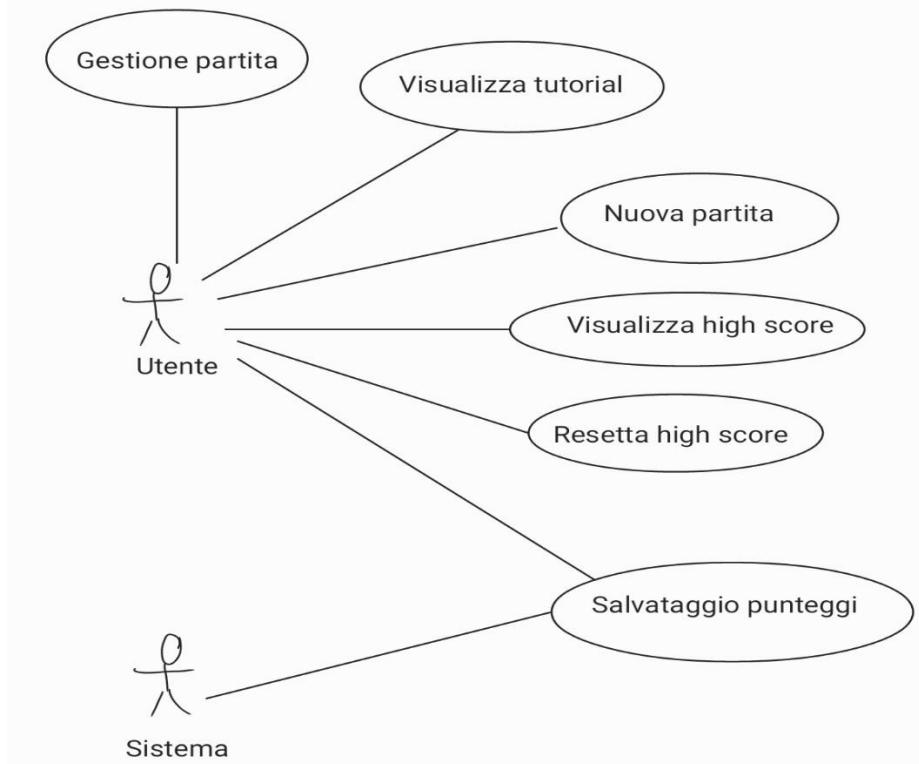
UML

5 Introduzione

Questa seconda parte del documento, a differenza della precedente, sarà orientata verso il design che sarà dato all'applicazione. Verranno inizialmente introdotti dei diagrammi finalizzati a descrivere cosa deve fare l'app, questo sarà fatto tramite use case diagram, activity diagram e state diagram; successivamente sarà descritta la struttura da dare all'app partendo dalle relazioni tra vari package, usando un package diagram, per poi trattare le classi coinvolte nelle principali operazioni richieste all'applicazione tramite dei class diagram. Per concludere saranno inseriti dei sequence diagram per spiegare come le operazioni più complesse saranno effettuate.

6 Use case diagram

Lo scopo di questo diagramma sarà mostrare gli use case a cui gli utenti del sistema sono collegati in modo da dare una panoramica generale su quello che l'app dovrà poter fare

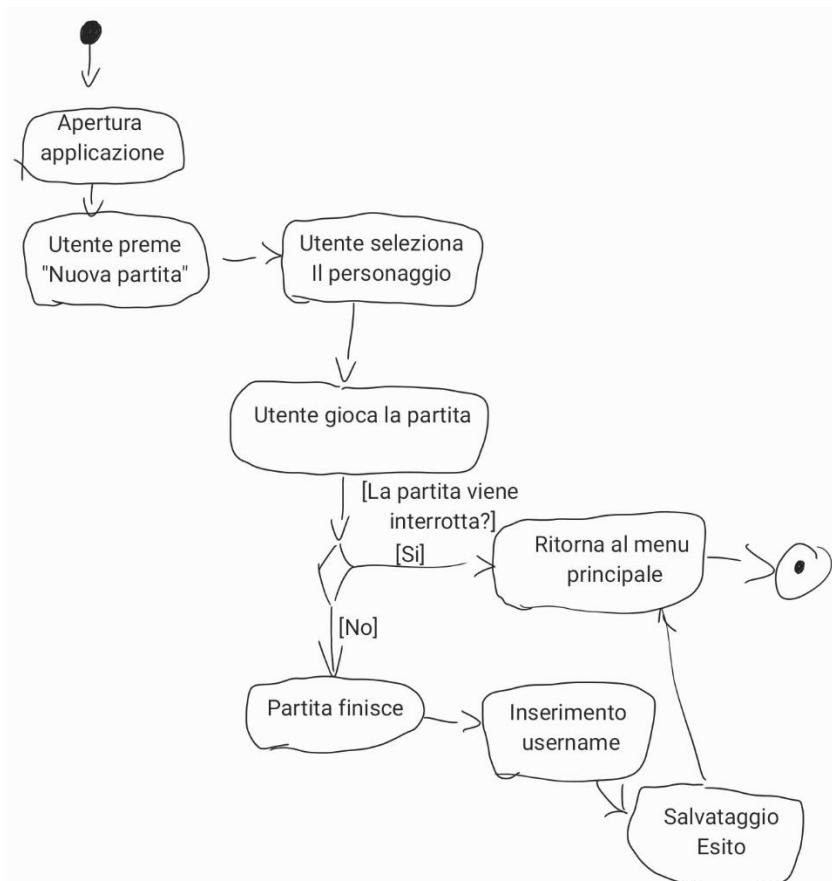


7 Activity diagram

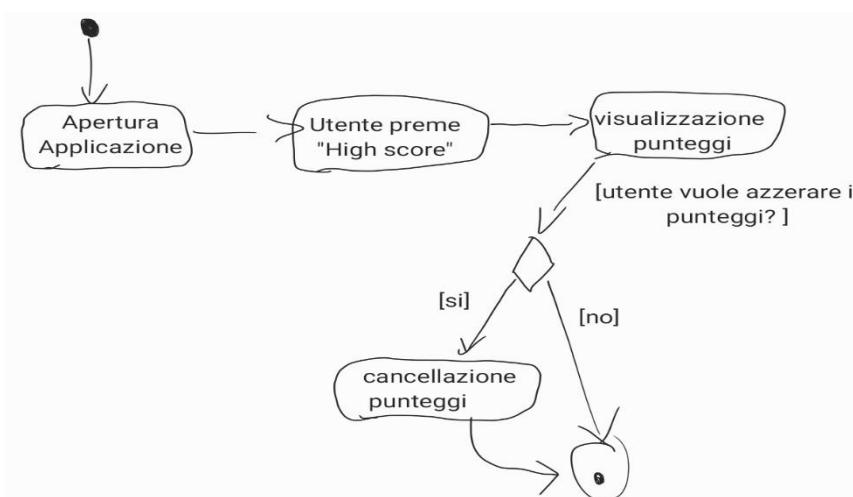
Saranno ora riportati degli activity diagram finalizzati a mostrare il flusso di attività seguito dalle operazioni più frequenti nel sistema.

7.1 Partita

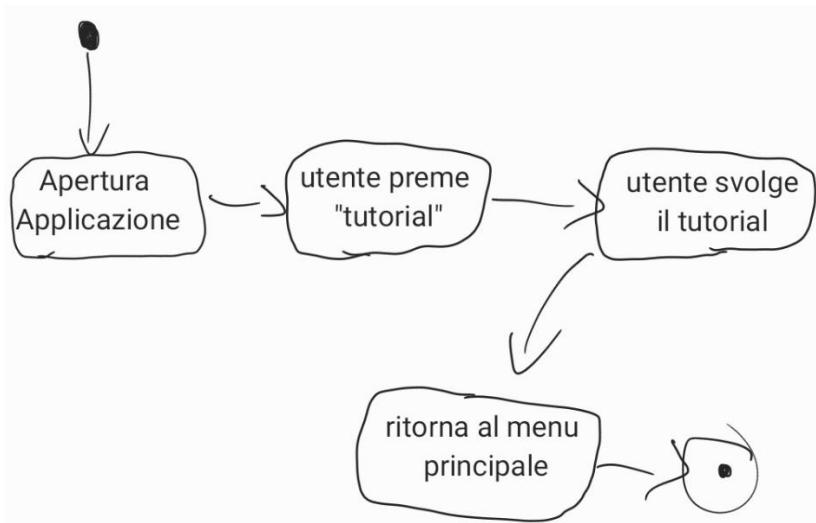
Il diagramma seguente si propone di spiegare le varie operazioni che saranno eseguite durante la partita, dalla sua creazione alla sua fine, ma senza scendere troppo nel dettaglio, i vari stati in cui una partita può trovarsi saranno analizzati nel capitolo 7



7.2 Visualizzazione ed eliminazione punteggi



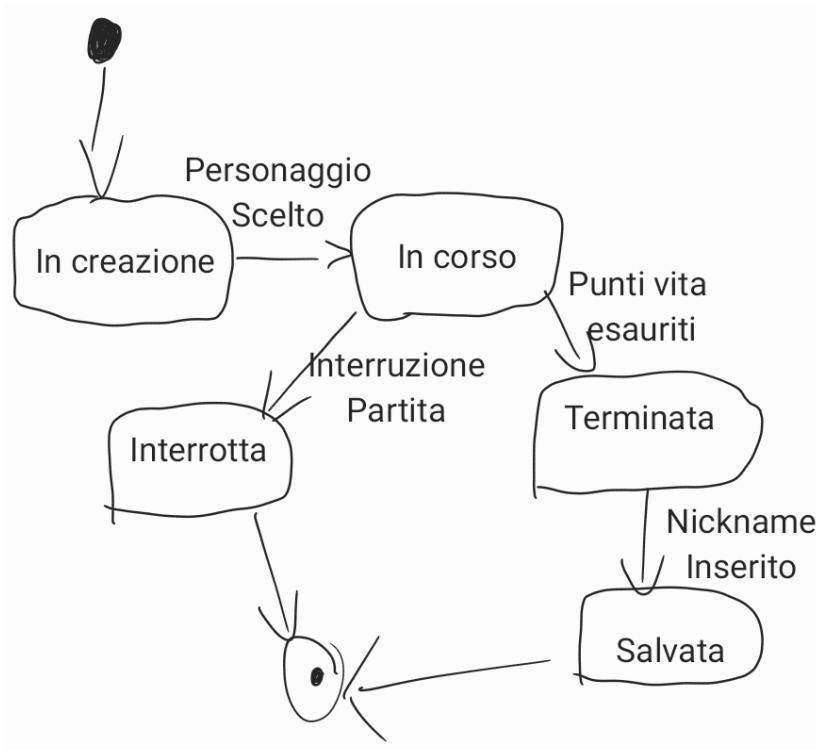
7.3 Visualizzazione del tutorial



8 State diagram

Saranno ora riportati due state diagram finalizzati a facilitare la comprensione dei vari stati in cui possono trovarsi una partita ed un personaggio.

8.1 Partita

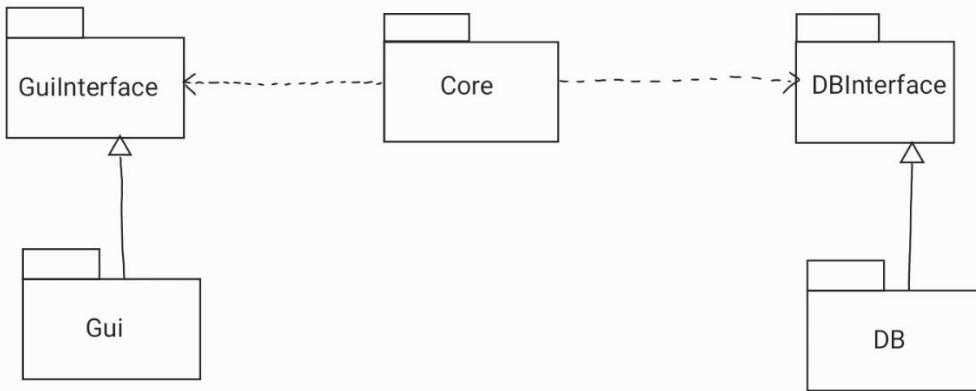


8.2 Personaggio



9 Package diagram

Il diagramma che sarà ora riportato è finalizzato a fornire un'idea dei moduli in cui sarà divisa l'applicazione; in questa sezione non andremo nel dettaglio ad analizzare le singole classi in essi contenute, ma daremo una descrizione generale su come sarà strutturato il software e analizzeremo le varie funzionalità dei vari package introdotti.



- **Core**: sarà il package in cui andremo ad inserire la “logica” dell’applicazione e saranno qui presenti tutte le classi che implementano le sue business rules, non ci sarà però nulla legato all’interfaccia grafica o all’accesso ai database; questi essendo componenti di natura più volatile saranno utilizzati dalle classi core tramite gli appositi package di interfaccia.
- **GuilInterface**: è un package che fa da interfaccia fra le classi core e l’implementazione della gui. Questa scelta è stata fatta per invertire le dipendenze e quindi per facilitare il riuso del codice, infatti in questo modo le classi core saranno disaccoppiate da ciò che concerne l’interfaccia grafica e in futuro, se necessario sarà possibile apportare modifiche a quest’ultima senza influenzare le classi nei package più stabili.
NOTA: questa pratica favorisce anche un possibile approccio multipiattaforma, che magari in futuro potrebbe rivelarsi necessario
- **Gui**: è il package in cui sarà implementata la gui vera e propria, esso dipende dal package “GuilInterface” ed implementa tutte le funzionalità necessarie per il funzionamento dell’app dal punto di vista grafico, pur restando separato dal package “core”
- **DBInterface**: svolge una funzione analoga a quella del package **GuilInterface**, e si occupa di invertire le dipendenze fra i package **Core** e **DB**
- **DB**: sarà l’implementazione della **DBInterface**

NOTA: nel capitolo seguente si andranno ad introdurre le classi coinvolte nelle operazioni principali che avvengono nel sistema; per rendere questo design il meno vincolante possibile NON saranno presenti classi provenienti dai package “**Gui**” e “**DB**” per via della loro natura particolarmente volatile e poiché nel caso della **gui**

è già stata data un'idea generale su come dovrà essere strutturata nel capitolo 3.1. Al più dove necessario si inseriranno le interfacce facenti parte dei package “`GuilInterface`” e “`DBInterface`”; il package su cui ci si concentrerà maggiormente sarà “`Core`”, essendo in esso contenuta la maggior parte della logica del programma. Saranno inoltre tralasciati in questi diagrammi:

- Schermata Iniziale: poiché di natura prettamente grafica
- Gestione della musica: poiché considerato un requisito di importanza secondaria e non indispensabile al funzionamento dell'app

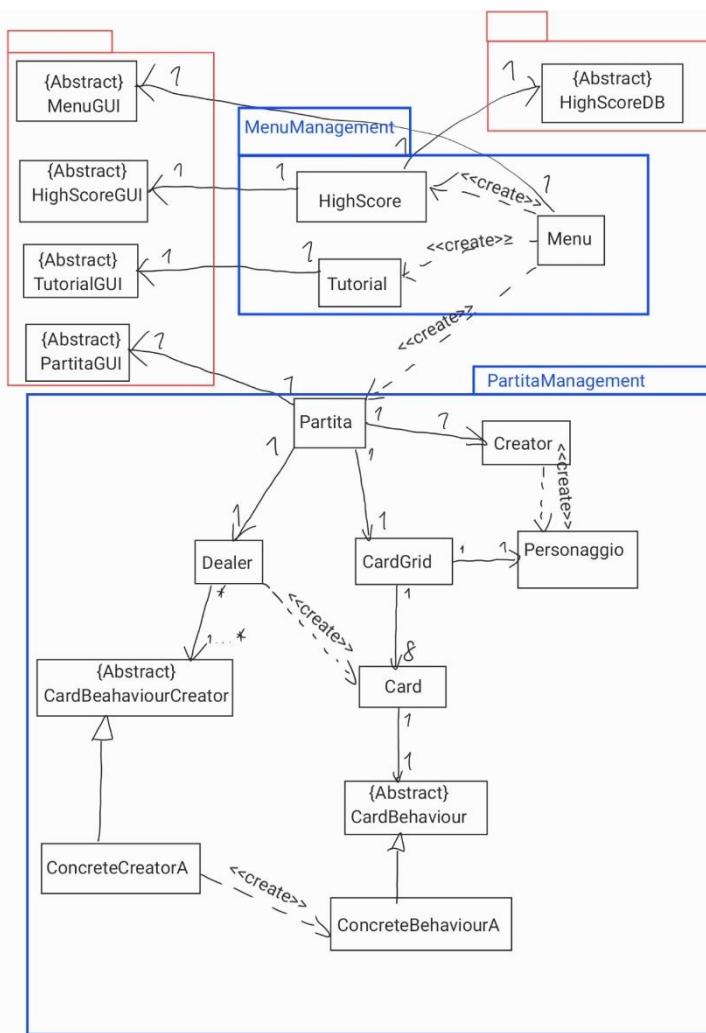
10 Class diagram

Andremo ora a descrivere il design da dare all'applicazione concentrandoci, come già detto, sul package "core", che conterrà le classi più importanti.

Introdurremo una divisione ulteriore di questo package in 2 parti:

- MenuManagement: gestisce il menu principale e l'istanziazione delle classi che rappresentano le 3 sezioni dell'applicazione
- PartitaManagement: contiene le classi che saranno usate durante una partita

Sarà ora fornita una "mappa generale" di come sarà strutturato il design; essa si presenterà come un package diagram dove ogni package contiene dei class diagram, successivamente andremo ad analizzare nel dettaglio le varie classi chiave tramite diagrammi più dettagliati.



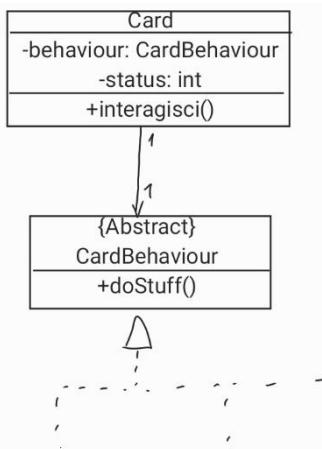
NOTA: i package in rosso si riferiscono a GUIInterface e DBInterface, come è facilmente intuibile.

10.1 Gestione delle carte (pattern “strategy”)

La carta “personaggio” sarà una classe separata dalle altre in virtù delle sue caratteristiche particolari (contiene 2 attributi numerici invece che 1), mentre per quelle “base” è stato scelto di creare una classe unica, classe che tramite il design pattern “strategy” sarà configurabile in modo da potersi comportare in modi diversi in base alla strategia assegnata.

La scelta di distinguere le varie tipologie di carte in funzione di un loro attributo potrebbe sembrare controtuitiva, poiché non si avranno effettivamente carte diverse; il vantaggio sarà però grande se si deciderà di estendere il software con nuove tipologie di carte; basterà infatti estendere la classe “CardBehaviour” e si potrà effettuare la modifica senza intaccare il resto del software rispettando l’open-close principle

Il class diagram che la rappresenta situazione descritta è il seguente:



Andiamo ora ad analizzare la struttura data:

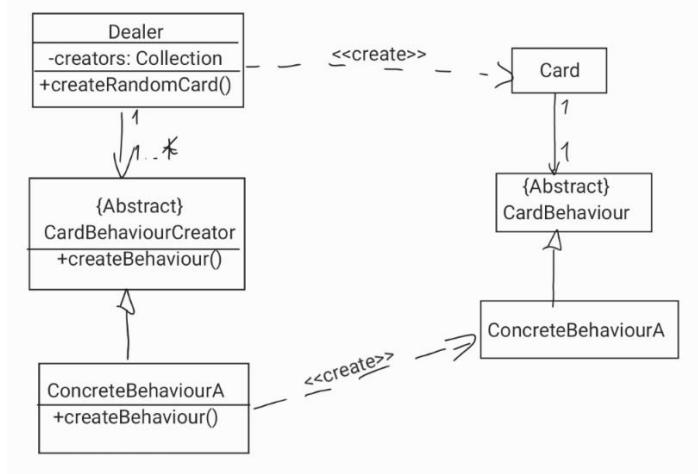
- **Interagisci():** questo metodo verrà chiamato quando il personaggio si sposta sulla casella che contiene la carta in questione e si occuperà di effettuare una chiamata al metodo “doStuff()” del suo behaviour, che gestirà l’interazione in funzione della sua tipologia (combattimento se è un mostro o ricarica hp se è una pozioni per esempio)
- **Status:** questo attributo è inserito nella classe Card poiché tutte le carte all’infuori del “personaggio” hanno un solo parametro di tipo numerico e l’unica cosa che cambia fra di loro è come esso viene utilizzato durante le interazioni
- **doStuff():** è il metodo che si occupa di gestire l’interazione

NOTA: ovviamente la classe CardBehaviour dovrà essere estesa per poter essere utilizzata (una volta per ogni tipologia di carta)

10.2 Gestione del “Dealer” (pattern “factory method”)

La classe “Dealer” sarà quella che si occuperà di fornire in modo randomico le carte da inserire nel tavolo da gioco. Per l’istanziazione dei vari behaviour (che differenzieranno appunto le varie carte) sarà utilizzato il pattern “factory method”, il quale ci permetterà di far fronte all’aggiunta di nuove tipologie di carte in modo efficiente, poiché per creare loro un creator basterà estendere quello astratto.

Il class diagram che rappresenta la situazione descritta è il seguente:

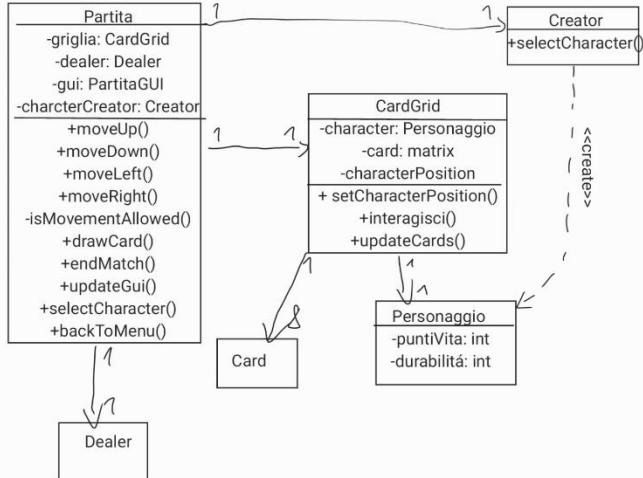


Andiamo ora ad analizzare la struttura data:

- **creators**: sarà una generica collection contenente un numero di **CardBehaviourCreators** pari a quello dei **ConcreteBehaviours** presenti, ogni creator sarà dedicato alla creazione di uno specifico behaviour
- **createRandomCard()**: crea una carta e poi randomicamente sceglie uno dei vari creator nella collection, per poi assegnare il behaviour creato alla nuova carta.
- **createBehaviour()**: banalmente crea il **concreteBehaviour** a cui è associato

10.3 Gestione della partita e selezione del personaggio (tramite classe “factory”)

Andremo ora ad analizzare nel dettaglio la struttura delle classi che gestiscono la partita:



Analizziamo la struttura:

- metodi di movimento: controllano con la funzione apposita se il movimento è consentito e poi chiamano il metodo `“setCharacterPosition”` e il metodo `“interagisci”` sulla carta presente nella posizione in cui ci si sposta.
- `drawCard()`: dopo un movimento crea e aggiunge una carta nella casella rimasta vuota
- `endMatch()` chiamato quando i punti vita del personaggio diventano minori o uguali a zero, chiude la partita e gestisce il salvataggio dei dati
- `updateGui()`: usa l'attributo **gui** per richiedere un update dell'interfaccia grafica
- `updateCards()`: si usa per inserire una nuova carta nella griglia
- `selectCharacter()`: il metodo deve aprire un'interfaccia che permette all'utente di selezionare il proprio personaggio, non è necessario utilizzare un vero proprio `“factory method”`, poiché al contrario delle carte

base, che dovranno avere comportamenti diversi fra loro, le uniche differenze fra i vari personaggi saranno i valori iniziali degli attributi "hp" e "durability", non serviranno perciò classi distinte.

E' comunque una buona pratica delegare a una classe diversa da "partita" l'istanziazione delle carte "Personaggio" per rendere più semplice l'aggiunta di nuovi personaggi in e non sovraccaricare di responsabilità la classe suddetta.

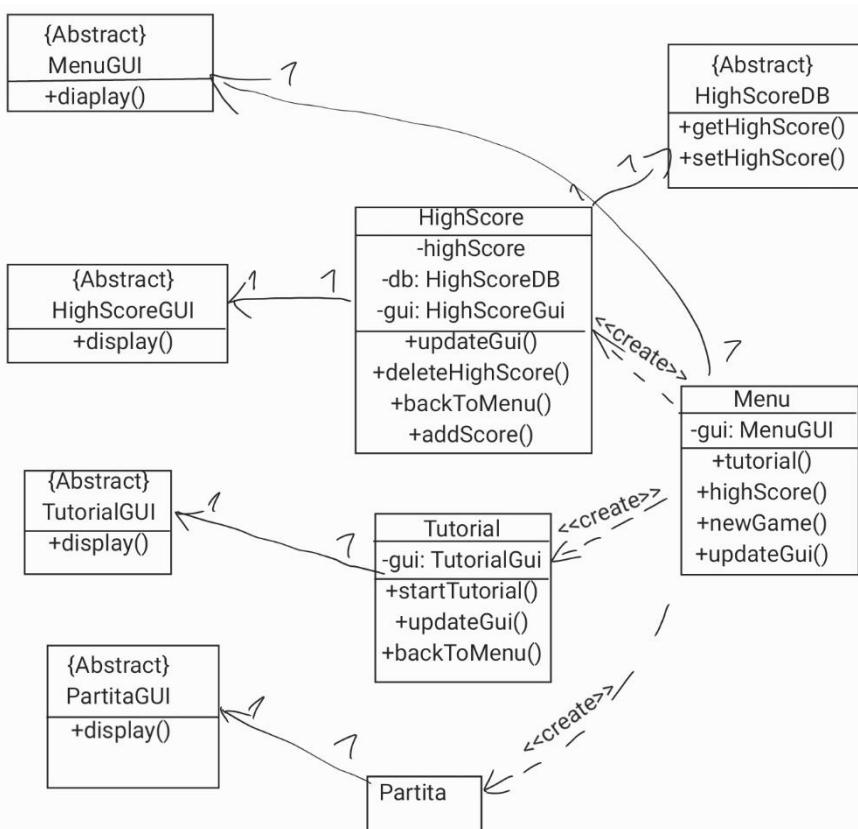
In sostanza il metodo "selectCharacter()" della classe partita si limiterà ad invocare quello del characterCreator, il quale aprirà il menu di selezione e gestirà l'istanziazione vera e propria comportandosi come un "simple factory"

NOTA: in generale il simple factory ritorna una classe astratta, qui non è necessario, poiché la classe creator ha l'unica utilità di incapsulare l'istanziazione degli oggetti, che però apparterranno tutti alla stessa classe. Inoltre se in futuro sarà necessario aggiungere nuovi personaggi questa sarà l'unica classe da modificare, poiché l'unica differenza fra le varie tipologie è data dai valori di inizializzazione.

- backToMenu(): interrompe la partita e riporta al menu principale

10.4 Gestione generale

Analizziamo più nel dettaglio la struttura delle classi che rappresentano il menu e quelle appartenenti ai package di interfaccia:



Analizziamo ora la struttura:

- gui: attributo presente in Menu, Tutorial, HighScore e Partita, rappresenta l'interfaccia con la gui
- backToMenu(): riporta al menu principale
- display(): metodo presente nelle interfacce del package gui, va implementato nelle classi del package Gui, si occupa della grafica

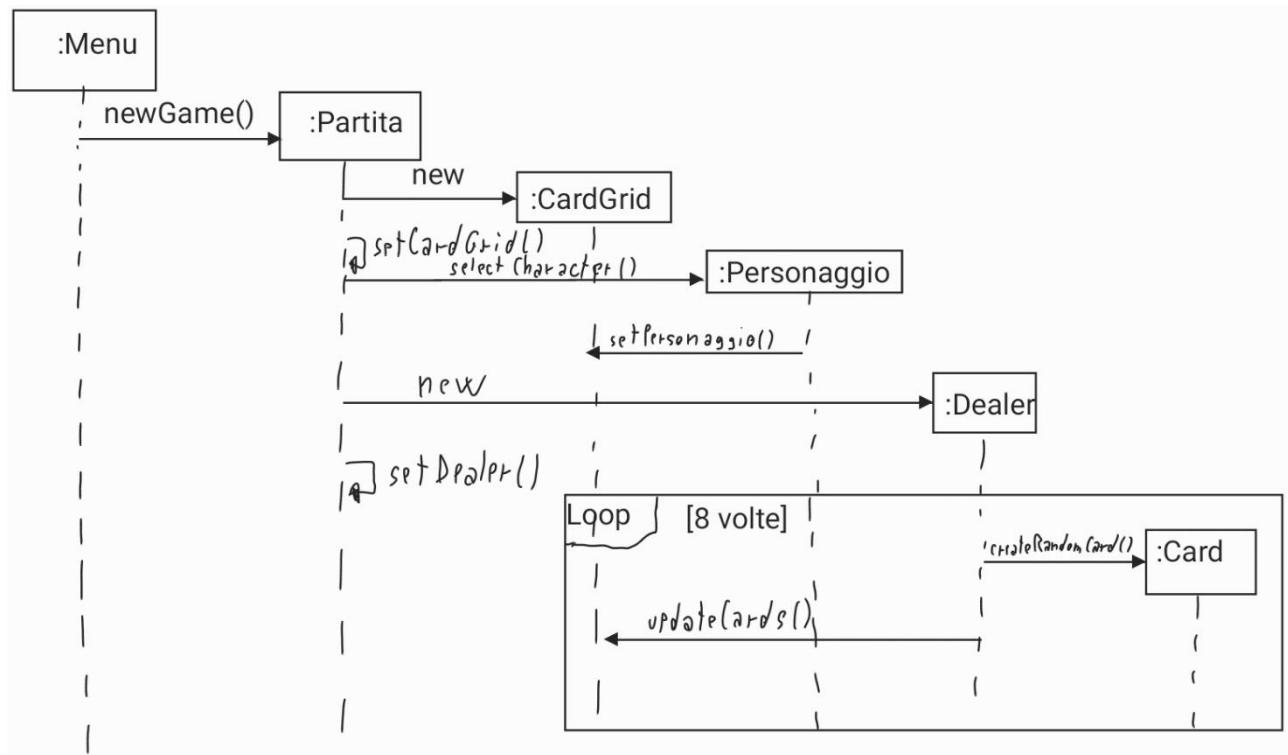
- updateGui(): si occupa di chiamare il metodo display() con i parametri corretti
- HighScoreDB: i metodi di questa interfaccia permettono di gestire i dati salvati in un database
- Menu: i suoi metodi si occupano di instradare verso le funzionalità dell'applicazione
- addScore(): serve ad aggiungere al database il punteggio di una partita terminata

11 Sequence diagram

In questa sezione saranno forniti dei sequence diagram finalizzati a chiarire la sequenza di operazioni da eseguire per dare luogo alle funzionalità più complesse del sistema.

11.1 Creazione di una nuova partita

Qui viene analizzata la sequenza di operazioni principali da eseguire per la creazione di una nuova partita, va fatto notare la selezione del personaggio verrà spiegata in modo riduttivo e si limiterà alla chiamata del metodo "selectCharacter()" da parte della classe "Partita"; questa operazione sarà spiegata nel dettaglio in seguito.

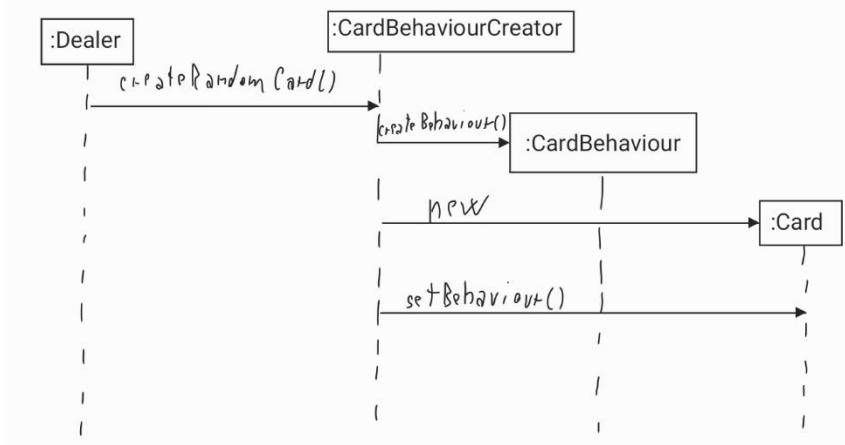


NOTA: l'operazione createRandomCard() prevede l'utilizzo del factory, qui non è stato inserito, poiché il prossimo sottocapitolo sarà dedicato al modo in cui il dealer istanzia le carte

11.2 Istanziazione di una nuova carta

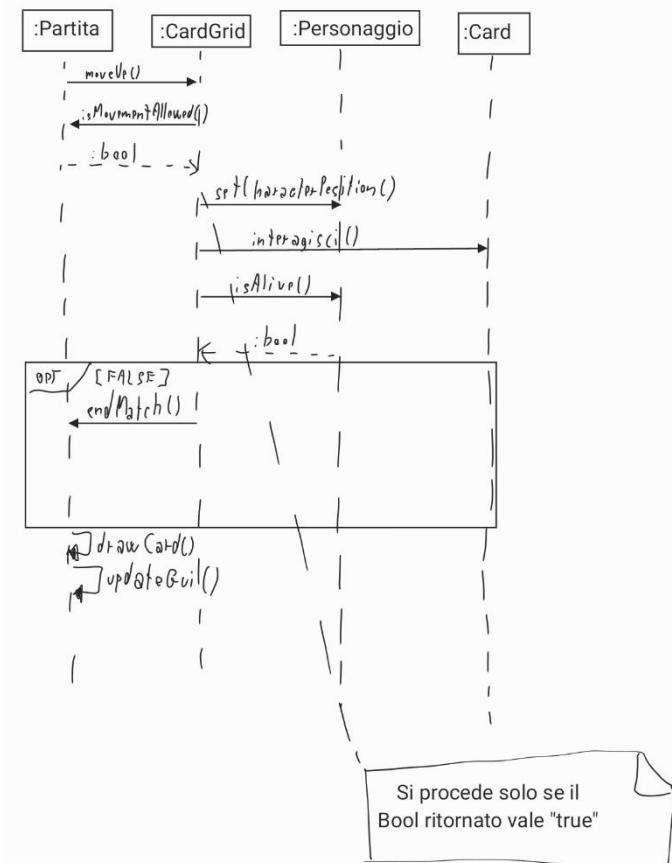
L'obiettivo del diagramma seguente è spiegare la sequenza di operazioni da eseguire durante la creazione di una nuova carta; in particolare ci concentreremo sulle operazioni che saranno effettuate dal dealer quando viene chiamato il metodo "drawCard()"

NOTA: l'unica altra situazione in cui il metodo createRandomCard() è utilizzato è in fase di creazione di una nuova partita



11.3 Movimento del personaggio

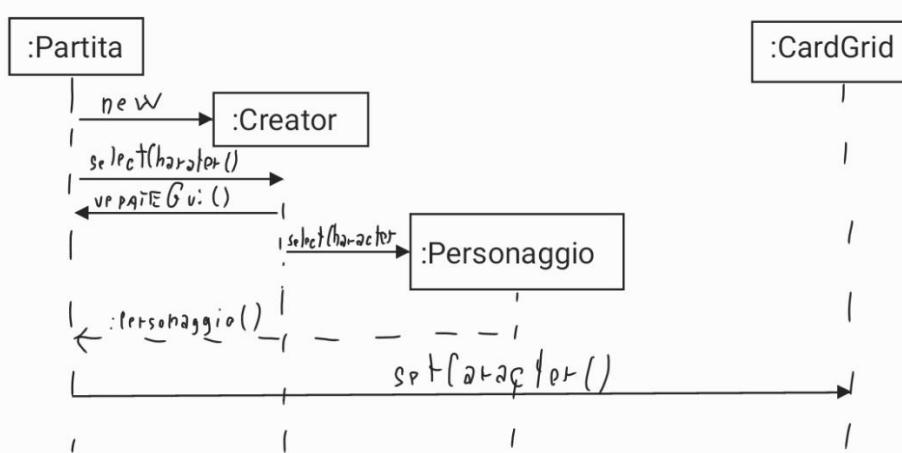
Nel diagramma che segue il metodo utilizzato sarà "moveUp", la sequenza di operazioni sarà la stessa anche per tutti gli altri movimenti.



NOTA: il metodo endMatch() avvia la procedura di salvataggio del punteggio effettuato, ciò che avviene sarà spiegato nel dettaglio in seguito

11.4 Selezione del personaggio

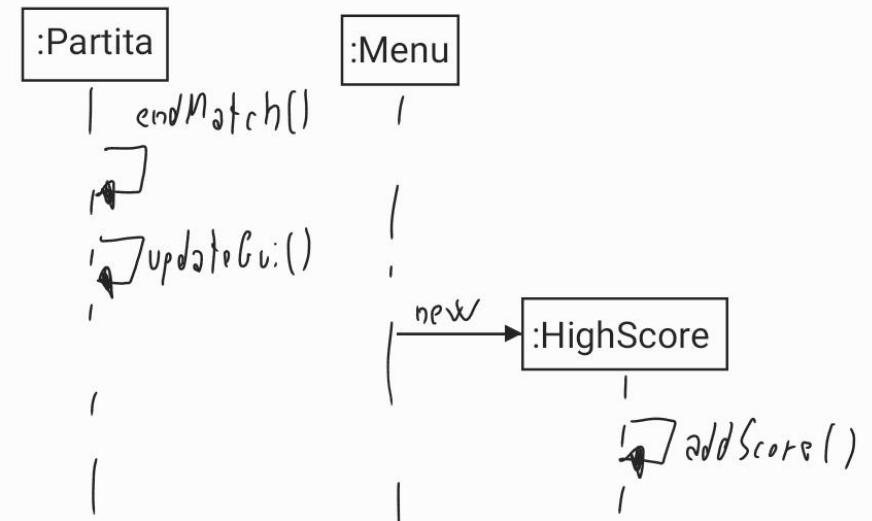
Questo paragrafo è finalizzato ad analizzare le operazioni necessarie alla creazione del personaggio che erano state tralasciate in 11.1



Come si può vedere dal grafico riportato la selezione del personaggio prevede prima la creazione del Creator e poi la selezione vera e propria, l'update della gui sarà necessario per mostrare all'utente le varie opzioni fra cui scegliere

11.5 Salvataggio del punteggio

Saranno mostrate in questo paragrafo le varie operazioni da effettuare durante il salvataggio del punteggio della partita in seguito alla chiamata del metodo “endMatch()”:



NOTA: la chiamata ad endmatch() prevede anche la richiesta di un username, che sarà quello associato al punteggio ottenuto.