INDICE

* DESCRIZIONE STORIA
* EVENTI DEL GIOCO
* GUIDA PER IL COMPLETAMENTO DEL GIOCO
* DESCRIZIONE DELLA STRUTTURA DEL PROGETTO
* SPECIFICA ALGEBRICA
* DIAGRAMMI DELLE CLASSI

The Fedora’s Story

Johnny U Mafius ha un appuntamento al club dei suoi colleghi mafiusi, ma quella mattina si sentiva una aria diversa.

La serata al bar era come una delle solite e solo un fattore esterno avrebbe potuto sconvolgere tutto.

Un boss della malavita vede qualcosa su johnny che avrebbe davvero voluto possedere.

La fedora di Al Capone.

[Prologo](http://www.dis.uniroma1.it/~nardi/Didattica/fi12000/canale4/SecondaSettimana/QuartaLezione.htm" \l ":~:text=println()

[20 giorni prima](http://www.dis.uniroma1.it/~nardi/Didattica/fi12000/canale4/SecondaSettimana/QuartaLezione.htm" \l ":~:text=println()

[Una giornata tranquilla nella chiassosa Brooklyn, Johnny “o mafius” si sta ritirando a casa da una serena cena con i colleghi mafiosi.](http://www.dis.uniroma1.it/~nardi/Didattica/fi12000/canale4/SecondaSettimana/QuartaLezione.htm" \l ":~:text=println()

Johnny era sempre stato uno a cui piaceva apparire e piaceva farlo con stile, ma questa cosa li si ritorse contro.

Appena uscito dal ristorante del suo miglior amico e collega Vivio Marzalma,si ritrovò con le spalle al muro in un vicoletto li vicino…

Una banda di scapestrati mandati da qualcuno l aveva già accerchiato e sembrava fossero interessati solo al derubarlo anche a costo di non lasciarlo in vita.;

Davvero strano” pensò Johnny mentre guardava negli occhi i suoi malfattori, ”guardano tutti la mia fedora di al capone, saranno interessati davvero a quella?;

Non fece in tempo a pensare un alternativa per scappare che senti una botta e poi tutto nero.

Oggi:

Johnny si è sveglia con una chiamata di Vivio:”Johnny ma che cazzo stai facendo!?!”

“Hey vivio mi sono appena svegliato,sto nel letto”

“Coglione sveglia non cè tempo da perdere!!!

“Dimmi che è quello che penso…”disse Johnny con gli occhi di fuoco.”

”Paparino sta tornando!”

*Il Furto: eventi principali.*

***Presenza accendino***: l accendino lo si può trovare nel salotto(stanza 3) dopo aver acceso la luce.

***Evento guardia***: l’ evento della guardia che si muove all’interno della casa viene avviato quando il giocatore entra per la prima volta nella sala da pranzo

***Presenza padella***: la padella si potrà prendere solo nel caso in cui l evento guardia è in esecuzione. La padella può essere utile nel caso in cui ci trovassimo nella stessa stanza della guardia e fossimo nascosti in un armadio o sotto un letto. In questo caso potremmo stordire la guardia per il resto della partita.

***Luce rotta***:la stanza 11 avrà la luce rotta e per illuminarla avremmo bisogno dell’accendino.

***Chiave della cantina***: la chiave della cantina la potremmo trovare in sala da pranzo ma non sarà subito visibile, infatti essa verrà notata solo quando il giocatarore entrerà nuovamente nella sala da pranzo

***Chiave del tesoro***: si trova nella stanza del boss (stanza 10) e permette di aprire la stanza dei tesori del boss(stanza 9).

*Guida per completare il gioco*

Per completare il gioco nel meno breve tempo possibile basta seguire questi punti

* Dalla stanza di partenza entrare a nord, che vi porterò nella cucina;
* Dalla cucina procedere a nord per entrare nel bagno;
* Dal bagno entrare nella porta a ovest, entrando così nella stanza dove sono presenti le scale per il secondo piano;
* Dal corridoio entrare a ovest che vi porerà nell corridoia del secondo piano;
* Dal corridio entrare nella porta a sud, che vi porterà nella stanza del boss dove sarà presente la chiave della stanza dei tesori;
* Prendere al chiave;
* Entrare nella porta a nord;
* Entrare nella porta a est;

**NOTA :** QUESTA GUIDA EVITA DI ATTIVARE L’EVENTO DELLA GUARDIA CHE CAMMINA PER LA CASA

*DESCRIZIONE STRUTTURA DEL PROGETTO*

La struttura del progetto si basa su tre package:

* inputUtente
* outputUtente
* logica

Il package **inputUtente** contiene la classe Parser che mette a disposizione un solo metodo denominato parser.

Il metodo parser prende in input un stringa, che rappresenta il comando digitato dall’utente, e una istanza della classe di GameManager, che rappresenta il cuore del gioco.

La logica del metodo parser consiste nel modificare la comando passato in una struttura del tipo <soggetto><oggetto> (Es “apri la porta a nord” diventa “apri nord”); Questra ristrutturazione della frase è resa possibili grazie a due insiemi “paroleConcesse” e “paroleDaCancellare” (insiemi avvalorati tramite i due omonimi file .txt presenti nel package “resources”.

Dopo aver modificato la struttura del comando in base al <soggetto> e a volte anche in base a <oggetto> vengono invocati dei metodi della istanza di GameMenager che rappresentano i comandi del gioco.

Se il giocatore digita un comando che non ha implementazione nel gioco, il metodo parser restituirà una ParserException.

L’eccezione del metodo parser viene descritta nella classe ParserExceprion, presente sempre nel package inputUtente, dove tale classe erida da Exception e mette a disposizione un mettore getMessage() che restituisce una stringa.

Il package **outputUtente** contiene solo la classe Dialoghi, una classe priva di attributi e costruttore che mette a disposizione una serie di metodi statici per stampare a schermo alcuni messaggi durante il gioco.

Il package **logica** contiene tutte le classi core del gioco.

La classe GameManager mette a disposizione un costruttore, dei metodi che rappresentano l’implementazione dei comandi di giochi (come ad esempio muovi(), raccoltaOggetto(), …) e alcuni metodi per inizializzare gli insiemi “paroleConcesse”, “paroleDaCancellarre” e “casa” (l’insime casa contiene oggetti di tipo Stanze che descrivono le varie stanze).

Per quanto riguarda gli attributi di GameManager, abbiamo:

* nascosto – una variabile booleana per stabiliere se il giocatore è nascosto oppure no
* stanzaCorrente – un variabile intera che serve per memorizzare il numero della stanza corrente (ogni stanza ha un nummero associato)
* vivo – una variabile boolean per indicare se il giocatore è vivo e quindi il gioco può continuare. Se vivo assume valore FALSE il gioco termina
* invetario – un array che contiene valori enumerativi messia a disposizione dalla classe enumerativa Oggetti
* stanzeVisitata – un vettore di booleani dove l’elemento i-esimo assume valore TRUE se la stanza numero i+1 è stata visitata, FALSE altrimenti . (vettore utile per mostrare alcuni oggetti solo dopo essere entrato nuvamente in una determinata stanza)
* casa – un insieme con implementazione HashSet, che contiene istanze della classe Stanze
* paroleConcesse, paroleDaCancellare – un insieme con implementazione HashSet
* evento – una istanza della classe Evento,utile per la gestione dell’vento della guardi che cammina per la casa
* salvataggio – una istanza della classe GestioneSalvataggio per la gestione della carica e salvataggio della partita in un db locale

La classe Evento, si occupa della gestione dell’unnico thread del progetto infatti essa eredita da Thread.

Evento mette a disposizione una metodo run() che contiene tutto il codice del thread, in paritoclare questo thread si occupa di far muovere una guardia per l’intera casa traminte una serie di scelte generate in modo casuale; sa la guardi si trova nella nostra stessa stanza stamperà un messaggio che avviserà il giocatore dell’arrivo della guardia.

Per comunicare la posizione del giocatore al thread si fa uso del costruttore, per la posizione non appena viene attivato il thread, e nelle fasi succesive il comando muovi() di GameManager attraverso l’istanza di Evento chiamera il metodo cambiaStanzaCorrente() cosi il thread saprà sempre la posizione del giocatore

Oltre il metodo run() la classe Evento mette a disposizione il metodo interrupt() che serve per terminare il thread e quindi terminare l’evento; il metodo interrupt() verrò chiamato solo quando il giocatore sarà nascosto è usare il comando “usa padella” per stordire la guardia

La classe GestioneSalvataggio si occupare di carica e leggere da un database locale il salvataggio della partita corrente.

GestioneSalvataggio mette a dispozione un metodo connesioneDB() per la connesione al db locale, utilizzato da tutti i restanti metodi definiti nella classe.

Il metodo creazioneTabellaDB() è un mettodo che si collega al db attraverso il metodo connesioneDB() e crea una tabella attraverso una query. La query usata è struttura il modo tale che se la tabella è gia presente non deve essere create nuovamente.

Il metodo inserimentoSalvataggioInTabella() prende in unput alcuni attributi dell’ istanza di GameManagaer e attraverso un praret statment carica questi dati nella tabbella presente sul db. La gestione della tabella segue la seguente logica: la tabella ha sempre una solo riga, quindi quando viene invocato questo metodo si è deciso di cancellare e creare nuovamente la tabella e poi inserire i dati

Il metodo caricaSalvataggio() si occupa invece di leggere tutti i dati presenti nella tabella; tale metodo restituisci un’oggetto di tipo GameManager contenente tutte le informazioni lette dal db.

La classe Stanze è una classe le cui istanze servono per rappresentare le varie stanze che formano la casa. Stanze mette a disposizione solo un cotruttore per istanziare gli oggetti e una serie di attributi per descrivere le varie stanze

La classe enumeriva Oggetti serve per creare dei vaolori enumerativi degli oggetti sparsi nella casa, valori utilizzati nella istanzazione della classe Stanze

L’interfaccia Iseritore è una interfaccia generica e allo stesso tempo un’interfaccia funzionale. Inseritore mette a disposizione un unico metodo generico inserisci() metodo che viene usato nei metodi di GameManager per avvalorare i tre isiemi.

*SPECIFICA ALGEBRICA DELLA CLASSE GESTIONE SALVATAGGIO*

Per quanto riguarda specifica algebrica delle classi del progetto, è possibile realizzarla solo per la classe GameManager, poiché la maggior aprte della logica consiste nel modificare attributi dell’oggetto o interagire con altri elementi (db, file) quindi la maggior parte dei metodi definiti nel progetto sono di tipo void.

Nella classe GameManager è presente un solo metodo che può essere considerato un osservatore, ossia “controllaInvatario”, che ci fornisce informazione sul attributo invetario dell’istanza di GameMangaer.

Il metodo restiruisce vero se l’ooggetto passato come paramentro è presente nel vettoreinventario, falso altrimetni

|  |  |
| --- | --- |
| *OSSERVATORI* | COSTRUTTORI DI **GAMER** |
| controllaIInvetario(GAMER,oggetto) | IF GAMER.invetario contiene oggetto than TRUE, else FALSE |

*Diagramma delle classi*