INDICE

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

*Descrizione Storia*

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”); .

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.

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).

La classe Evento, si occupa della gestione dell’unnico thread del progetto, che si occupa di far camminare una guardia tra le varie stanza.

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

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 valori 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(T a ), che viene usato nei metodi di GameManager per avvalorare i tre isiemi.

La classe Gioco è la classe che contiene il main del progetto e si occupa di leggere l’input da tastiera e passarlo al parser

*DESCRIZIONE DELLE SINGOLE CLASSI*

*Classe Parser – inputUtente*

La classe non contiene nessun attributo.

La classe Parser mette a diposizione i seguenti metodi:

* parser(String comadno, GameManager partita) – Questo metodo riceve una stringa che rapprenta il comando diigitato da tastiera dal giocatore e attravaerso i due insiemi paroleConcesse e paroleDaCancellare riforumula la frase secondo la struttura <soggetto><oggetto> e successivamente in base al campo <soggetto> o ad entrambi i campi <soggetto><oggetto> il metodo invoca alcuni metodi messi a disposizione da GameManager. Se il giocatore digita un comando che non è possibili riorgarnizzare con la struttura indicata sopra, il metodo lanciera una ParserException.

*Classe ParserException – inputUtente*

Poiche la classe ParserException viene usata per lanciare una eccezione, essa erediterà dalla classe Exception.

La classe non contiene nessun attributo.

La classe ParserException mette a disposizione i seguenti metodi:

* getMessage() - Questo metodo restituisce una stringa che comunica che il comando digitato non è vaido

*Classe Dialoghi – outputUtente*

La classe non contiene nessun attributo

La classe Dialoghi mette a disposizione i seguenti metodi statici:

* messaggioPresenzaChiaveCantina (Stanza stanza) – Questo metodo stampa a schermo un messaggio a schermo che comunica al giocatore la presenza di una chiave, ma solo se nell’oggetto stanza passato come parametro è presente Oggetti.CHIAVE\_CANTINA
* messaggioPresenzaAccendino (Stanza stanza) – Questo metodo stampa a schermo un messaggio a schermo che comunica al giocatore la presenza di un accendino, ma solo se nell’oggetto stanza passato come parametro è presente Oggetti.ACCENDINO
* messaggioPresenzaPadella (Stanza stanza) – Questo metodo stampa a schermo un messaggio a schermo che comunica al giocatore la presenza di una padella, ma solo se nell’oggetto stanza passato come parametro è presente Oggetti.PADELLA
* messaggioPresenzaChiaveTesoro (Stanza stanza) – Questo metodo stampa a schermo un messaggio a schermo che comunica al giocatore la presenza di un chiave, ma solo se nell’oggetto stanza passato come parametro è presente Oggetti.CHIAVE\_TESORO
* messaggioInzioEvento() - Questo metodo si occupare di stampare a schermo un messaggio che comunica l’inzio dell’evento della guardia al giocatore
* prologo() - Questo metodo si occupa di stampare a schermo il prologo della storia
* obiettivo() - Questo messaggio si occupa di stampare a schermo l’obiettivo del gioco
* oggi() - Questo metodo si occupa di stampare a schermo una introduzione per la storia del gioco
* piuTardi() - Questo metodo si occupa di stampare a schermo una introduzione per la storia del gioco
* fine() - Questo messaggio si occupa di stampre un messaggio conclusivo della storia
* attesaInput() - Questo messaggio al termine di ogni comando mostra a schermo un messaggio che chiede al giocatore il prossimo comando
* staza1() - Questo metodo si occupa di stampare a schermo alcune informazioni della stanza 1
* staza2() - Questo metodo si occupa di stampare a schermo alcune informazioni della stanza 2
* staza3() - Questo metodo si occupa di stampare a schermo alcune informazioni della stanza 3
* staza4() - Questo metodo si occupa di stampare a schermo alcune informazioni della stanza 4
* staza5() - Questo metodo si occupa di stampare a schermo alcune informazioni della stanza 5
* staza6() - Questo metodo si occupa di stampare a schermo alcune informazioni della stanza 6
* staza7() - Questo metodo si occupa di stampare a schermo alcune informazioni della stanza 7
* staza8() - Questo metodo si occupa di stampare a schermo alcune informazioni della stanza 8
* staza9() - Questo metodo si occupa di stampare a schermo alcune informazioni della stanza 9
* staza10() - Questo metodo si occupa di stampare a schermo alcune informazioni della stanza 10
* staza11() - Questo metodo si occupa di stampare a schermo alcune informazioni della stanza 11
* staza12() - Questo metodo si occupa di stampare a schermo alcune informazioni della stanza 12
* messaggioSalvataggioTerminato() - Stampa un messaggio che comunica il corretto salvataggio del gioco
* luceGuasta() - Stampa a schero un messaggio che comunica al giocatore che la luce è guasta
* luceSpenta() -Stampa un messaggio che comunica al giocatore la neccesità di accendere la luce
* erroreGestioneFile() - Stampa a schermo un messaggio che comunica un errore nella comunicazione con un file
* errore() - Stampa un generico messaggio di errore
* erroreLetturaParoleDaCancellare() - Stampa un messaggio che comunica un errore con la lettura dal file paroleDaCancellare.txt
* erroreLetturaParoleConcesse() -Stampa un messaggio che comunica un errore nella lettura del file paroleConcesse.txt
* erroreLetturaDescrizioneStaze() - Stamapa un messaggio che comunica un errore nella lettura del file DescrizioneStanze
* erroreStream() - Stamapa un messaggio che comunica un errore nella chiusura dello stram
* catturato() - Stampa un messaggio che comunica al giocatore che è stato catturato dalla guardia
* luceAccesa() - Stampa un messaggio che comunica al giocatore che la luce è stata accesa
* oggettoNonPresente() - Stampa un messaggio che comunica al giocatore che non è presente l’oggetto indicato nella stanza attuale
* oggettoNonPresenteInInvetario() - Stampa un messaggio che comunica al giocatore che non è presente l’oggetto indicato nell’inventario
* luceGiaAccesa() - Stampa un messaggio che comunica al giocatore che la luce è gia accesa
* luceGiaSpenta() - Stampa un messaggioc che comunica al giocatore che la luce è gia spenta
* necessitiDi(Stanza stanza) - Stampa un messaggio che comunica al giocatore che per accedere alla stanza passata in input ha biogno di uno specifico oggetto nell’inventario
* portaInestitente() -Stampa a schermo che non è presente nessuna porta nella direzione indicata precedentemente
* portaSbloccata() - Stampa un messaggio che comunica al giocatore che una porta è stata sbloccata
* aggiungiInvetario() - Strampa un messaggio che comunica l’inserimento di un oggetto nell’inventario
* spegniLuce() - Stampa un messaggio che comunica al giocatore che luce è stata spente
* nonPuoiUsare() - Stamapa un messaggio che comunica al giocatore che non può usare un oggetto precedentemente indicato
* guardiaAbbatuta() - Stampa un messaggio che comunica che la guardia è stata fermta
* nascostoArmadio() - Stampa un messaggio che comunica il al giocatore di essersi nascosto in un armadio
* nascostoLetto() - Stampa un messaggio che comunica al giocatore di essersi nascosto sotto il letto
* nessunNascondiglio() - Stampa un messaggio che comunica al giocatore che non ci sono nascondigli

*Classe GameManager – Logica*

La classe GameManager dispone dei seguenti attributi:

* nascosto – una variabile booleana per stabiliere se il giocatore è nascosto oppure no
* DIMENSIONE\_INVETARIO – una costante usate per definire la grandezza dell’array inventario
* NUMERO\_STANZA – una costante usata per definire il numero di stanza che compongnon la casa
* 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 GameManager dispone die seguenti metodi:

* nzializzaStanzaVisitata() - invocato dal costruttore e che inzializza il vettore stanzaVisitata con il valore FALSE
* inizializzaInvetario() - invocato dal costruttore e che inizlizza il vettore inventario con il valore Oggetti.VUOTO
* avvaloreParoleDaCancellare() -invocato dal costruttore e serve per avvalore l’insieme paroleDaCancellare tramite una lettura dal file paroleDaCancellare.txt
* avvaloreParoleConcesse() - invocato dal costruttore e serve per avvalorare l’insieme paroleConcesse tramite una lettura del file paroleConcesse.txt
* avvaloreCasa() - invocato dal cotruttore e serve per avvalorare l’insieme casa tramite una lettura dal file DescrizioneStanze
* ricercaStanzaCorrente() - metodo che restituisci l’istanza della classe Stanza, presente nell’insieme casa, che ha come attributo Stanza.numeroStanza il valore dell’altributo di GameManager stanzaCorrente
* ricercaStanzaPerNumero(int numeroStanza) - metodo che restituiscie l’instanza della classe Stanza, presente nell’insieme casa, che ha come attributo Stanza.numeroStanza il valore passato come paramtro
* raccoltaOggetto(String oggetto) – metodo che inserisce nell’array invetario l’oggetto indicato tramite la stringa passata come parametro. Se l’oggetto non è presente restituirà un messaggio di errore
* accendiLuce() - metodo che ricerca la stanza corrente traminte cercaStanzaCorrente() e imposta il valore Stanza.luce = TRUE. Dopo aver impostato l’attributo dell’istanza di stanza viene aggiornato anche l’insieme casa cosi da mantere Stanza.luce = TRUE. Se la luce è gia accesa il metodo restiturà in messaggio di errore
* apriInvetario() - stampa a schermo tutti gli elementi presenti nell’array invetario
* muovi(String direzione) – in base alla stanza corrente e alla direzione indicata tramite la stringa in input, il metodo modifica l’attributo stanzaCorrente effettuando tutti i vari controlli necessari (come ad esempio la presenza di una chiave nell’ invetario per entrare in una stanza)
* stampaMappa() - metodo invocato da muovi() al termine di ogni spostamente e stampa a schermo una mappa che indica la stanza attuale
* salvaPartitaa() - inserisce alcuni attributi della classe in un db (attraverso il metodo inserimentoSalvataggioInTabella) e infine cancella il file DescrizioneStanze e ne crea uno nuovo (attraverso il metodo creaNuovoFileDescrizioneStanze) andando a inserire, in questo nuovo file, le instanse di Stanza presenti nell’insieme casa

*Classe Evento – Logica*

La classe Evento rapprenta l’unico thread del gioco. In questo caso si è scelto di far ereditare la classe dalla classe Thread

La classe Evento dispone dei seguenti attributi:

* eventoInFunzione – una variabile booleana che indica se l’evento è in funzione oppure no
* eventoInPausa – una variabile boolenana che indica se l’evento in puasa oppure no
* stanzaGuardia – una varibile di tipo intero che indica il nuemero della stanza in cui si trova la guardia
* stanzaGiocatore – una variabile di tipo intero che indica il numero della stanza in cui si trova il giocatore

La classe Evento dispone dei seguenti metodi

* run() - metodo che si occupa di far cambiare stanza al guardia aggiornando l’attributo stanaGuardia in base ad una generazione casuale di un numero. Se la guardia entra nella stessa stanza in cui ci troviamo il metodo si mette in “pausa” e aspetta la mossa del giocatore
* Evento() - costruttore della classe
* cambiaStanzaCorrente(int numeroStanza) – metodo che aggiorna la variabile stanzaGiocatore in base al valore passato come parametro
* interrupt() - metodo che serve per terminare il thread

*Classe GestioneSalvataggio – Logica*

La classe GestioneSalvataggio dispone di un solo attributo:

* conn – instanza della classe Connection usata da tutti i metodi della classe

La classe GestioneSalvataggi dispone die metodi:

* GestioneSalvataggio() - costruttore della classe
* connessioneDB() - che si occupa di creare una connessione con il DB locale
* creazioneTabellaDB() - che si occupa di creare una tabella nel DB locale
* inserimentoSalvataggioInTabella(“parametri”) - si occupa di caricare nella tabella del DB i parametri passati in input al metodo. 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
* caricaSalvataggio() - metodo che restituisci una instanza di GameManager e che si occupa di leggere i dati presenti nella tabella del DB locale

*Classe Gioco – Logica*

*La classe Gioco non dispone di nessun attributo*

*La classe Gioco dispone del seguente metodo:*

* *main(Sting[] args) – metodo che si occupa di avviare il gioco e si occupare di leggere l’input da tastiera e passarlo alla classe Parse*

*Inerfaccia Insetitore - Logica*

L’interfaccia inseritore, è una interfaccia generica che mette a disposizione un solo metodo:

* inserisci(T a )

*Classe Oggetti – Logica*

La classe oggetti è una classe enumerativa che mette a disposizione i seguenti valori:

* VUOTO
* CHIAVE\_CANTINA
* ACCENDINO
* PADELLA
* CHIAVE\_TESORO
* FEDORA

*Classe Stanza - Logica*

La classe Stranza implementa l’interfaccia Serializable poiche le istanza di questa classe devo essere caicate nel file DescrizioneStaze

La classe Stanza dispone dei seguenti attributi:

* numeroStanza – una variabile di tipo intero che indica il numero della stanza
* descrizione – una Stringa che contiene la descrizione di una stanza
* oggettoRichiesto – variabile di tipo Oggetti che indica l’oggetto necessario per accedere alla stanza
* oggetto – variabile di tipo Oggetti che indica l’oggetto presente nella stanza
* luce – variabile di tipo booleano che indica lo stato della luce (ON = TRUE OFF = FALSE)
* letto – variabile di tipo booleano che indica la presenza o meno del letto
* armadio - variabile di tipo booleano che indica la presenza o meno di un armadio

*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*