INDICE

NOTA: Il gioco è stato sviluppato mediante l'utilizzo di Intellij e provando ad aprire il progetto in un altro ide (Netbeans e Eclipse) il gioco ritona una eccezione a causa dei file non trovati, cosa che non succede su intellij

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

20 giorni prima

Una giornata tranquilla nella chiassosa Brooklyn, Johnny “o mafius” si sta ritirando a casa da una serena cena con i colleghi mafiosi.

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 giocatore 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

1. Dalla stanza di partenza entrare a nord, che vi porterà nella cucina;
2. Dalla cucina procedere a nord per entrare nel bagno;
3. Dal bagno entrare nella porta a ovest, entrando così nella stanza dove sono presenti le scale per il secondo piano;
4. Dal corridoio entrare a ovest che vi porterà nel corridoio del secondo piano;
5. Dal corridoio entrare nella porta a sud, che vi porterà nella stanza del boss dove sarà presente la chiave della stanza dei tesori;
6. Prendere al chiave;
7. Entrare nella porta a nord;
8. Entrare nella porta a est;

NOTA : QUESTA GUIDA EVITA DI ATTIVARE L'EVENTO DELLA GUARDIA CHE CAMMINA PER LA CASA, PER ATTIVARE L'EVENTO DOPO IL PRIMO PUNTO ENTRARE A OVEST E POI A NORD E RIPRENDERE DAL QUARTO PUNTO

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.

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 GameManager 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'insieme casa contiene oggetti di tipo Stanze che descrivono le varie stanze).

La classe Evento, si occupa della gestione dell'unico 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 costruttore per istanziare gli oggetti e una serie di attributi per descrivere le varie stanze

La classe enumerativa Oggetti serve per creare dei valori enumerativi che rappresentano gli oggetti sparsi nella casa, valori utilizzati nella istanzazione della classe Stanze

L'interfaccia Inseritore 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 insiemi.

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 il seguente metodo:

- parser(String comadno, GameManager partita) – Questo metodo riceve una stringa che rappresenta il comando digitato da tastiera dal giocatore e attraverso i due insiemi paroleConcesse e paroleDaCancellare riformula 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 riorganizzare con la struttura indicata sopra, il metodo lancerà una ParserException.

Classe ParserException – inputUtente

Poiché 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 il seguente metodo:

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

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 necessità 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
- erroreLetturaDescrizioneStanze() - Stampa un messaggio che comunica un errore nella lettura del file DescrizioneStanze
- erroreStream() - Stampa un messaggio che comunica un errore nella chiusura dello stream
- 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 messaggio 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 bisogno 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() - Stampa 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() - Stampa 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 fermata
- 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 stabilire 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 compongono la casa
- stanzaCorrente – un variabile intera che serve per memorizzare il numero della stanza corrente (ogni stanza ha un numero 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
- inventario – 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 nuovamente in una determinata stanza)
- casa – una lista con implementazione Arraylist, che contiene istanze della classe Stanze
- paroleConcesse, paroleDaCancellare – due insieme con implementazione HashSet che contengono stringhe
- 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 caricamento e salvataggio della partita in un db locale

La classe GameManager dispone die seguenti metodi:

- inzializzaStanzaVisitata() - invocato dal costruttore e che inizializza il vettore stanzaVisitata con il valore FALSE
- inizializzaInvetario() - invocato dal costruttore e che inizializza il vettore inventario con il valore Oggetti.VUOTO
- avvaloreParoleDaCancellare() -invocato dal costruttore e serve per avvalorare 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 costruttore e serve per avvalorare la lista casa tramite una lettura dal file DescrizioneStanze

- ricercaStanzaCorrente() - metodo che restituisci l'istanza della classe Stanza, presente nella lista casa, che ha come attributo Stanza.numeroStanza il valore dell'attributo di GameManager stanzaCorrente
- ricercaStanzaPerNumero(int numeroStanza) - metodo che restituisce l'istanza della classe Stanza, presente nella lista casa, che ha come attributo Stanza.numeroStanza il valore passato come parametro
- 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. Ogni qual volta un oggetto viene raccolto si aggiorna all'interno della lista il valore dell'istanza che rappresenta la stanza corrente, andando ad imposta l'attributo oggetto = Oggetti.VUOTO
- accendiLuce() - metodo che ricerca la stanza corrente tramite cercaStanzaCorrente() e imposta il valore Stanza.luce = TRUE. Dopo aver impostato l'attributo dell'istanza di stanza viene aggiornato anche la lista casa cosi da mantenere Stanza.luce = TRUE. Se la luce è già accesa il metodo restituirà in messaggio di errore. Se questo metodo viene invocato mentre la guardia si trova nella nostra stessa stanza il metodo imposterà il valore vivo = FALSE e il gioco terminerà
- apriInvetario() - stampa a schermo tutti gli elementi presenti nell'array inventario
- 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' inventario per entrare in una stanza). Al termine dell'aggiornamento dell'attributo stanzaCorrente, l'attributo booleano nascosto sarà impostato su FALSE. Quando ci si sposta per la prima volta nella stanza 4 il metodo avviare il thread evento e aggiornerà ad ogni spostamento il parametro evento.stanzaGiocatore
- stampaMappa() - metodo invocato da muovi() al termine di ogni spostamenti e stampa a schermo una mappa che indica la stanza attuale
- salvaPartita() - 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 istanze di Stanza presenti nella lista casa

Classe Evento – Logica

La classe Evento rappresenta 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 booleana che indica se l'evento in pausa oppure no
- stanzaGuardia – una variabile di tipo intero che indica il numero 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 – istanza della classe Connection usata da tutti i metodi della classe

La classe GestioneSalvataggi dispone dei 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 istanza 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 Parser*

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 Stanza implementa l'interfaccia Serializable poiché le istanza di questa classe devo essere caricate nel file DescrizioneStanze

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 STRUTTURA DATI: LISTA

SPECIFICA SINTATTICA

Sort: LISTA, TIPOELEM, POSIZIONE, BOOLEAN

- CREALISTA() → LISTA
- LISTAVUOTA(LISTA) → BOOLEAN
- SCRIVILISTA(TIPOELEM, LISTA, POSIZIONE)
- PRIMOLISTA(LISTA) → POSIZIONE
- SUCCLISTA(POSIZIONE, LISTA) → POSIZIONE
- PREDLISTA(POSIZIONE, LISTA) → POSIZIONE
- LIGGILISTA(POSIZIONE, LISTA) → TIPOELEM
- FINELISTA(POSIZIONE, LISTA) → BOOLEAN
- CANCLISTA(POSIZIONE, LISTA) → LISTA

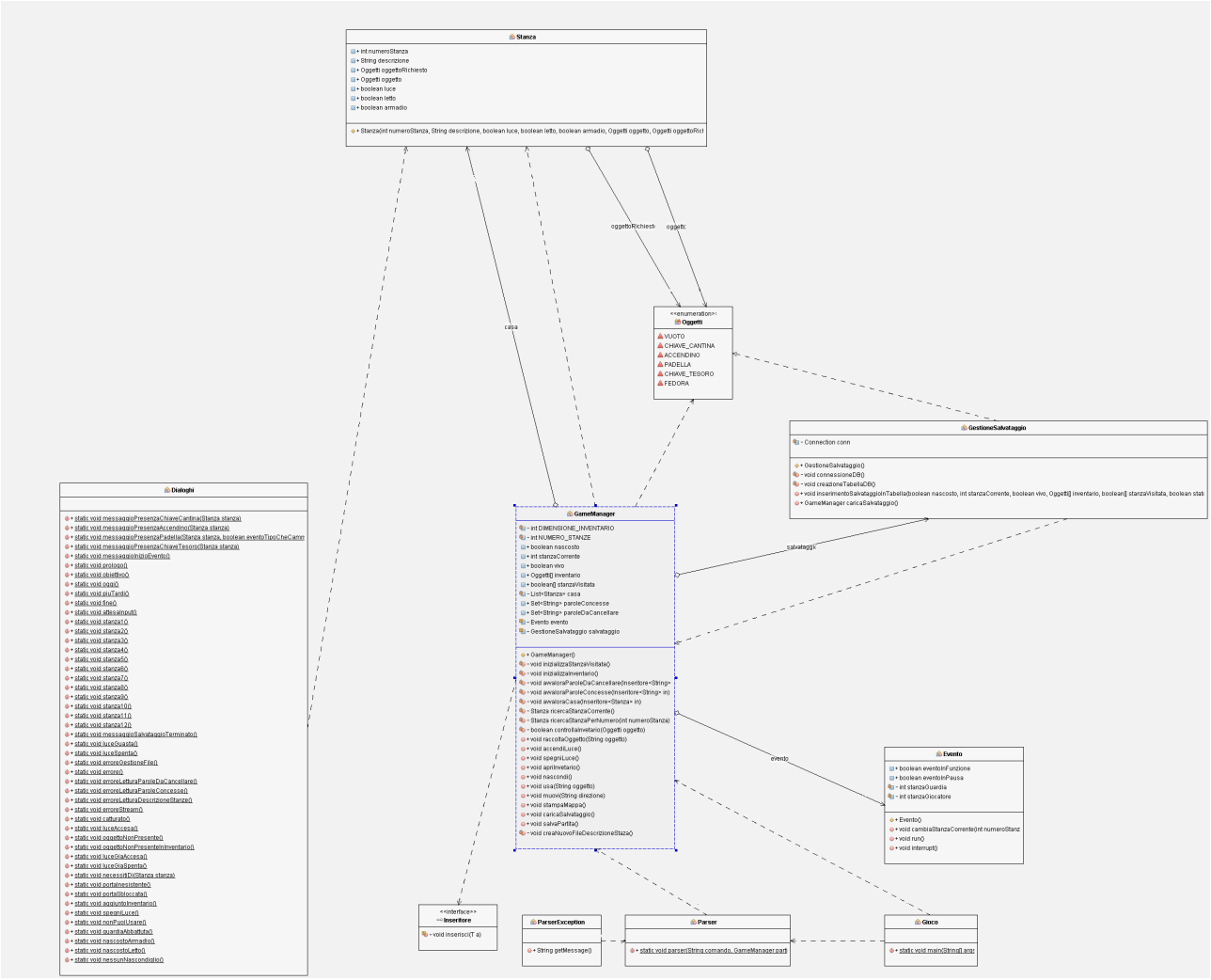
SPECIFICA SEMANTICA

Declare: LISTA : LIST , I: TIPOELEM, POS : POSIZIONE

- LISTAVUOTA(CREALISTA) = TRUE
- LISTAVUOTA(SCRIVILISTA(I, LIST, POS)) = FALSE
- PRIMOLISTA(SCRIVILISTA(I, LIST, POS)) = IF LISTAVUOTA(LIST) THAN POS ELSE PRIMOLISTA(LIST)
- SUCCLISTA (SCRIVILISTA(I, LIST, POS)) = IF FINELISTA(POS', LIST') THAN ERROR ELSE POS'+1
- PREDLISTA (SCRIVILISTA(I, LIST, POS)) = IF POS' = PRIMOLISTA(LIST') THAN ERROR ELSE POS'-1
- CANCLISTA (POS' , LIST') = IF LISTAVUOTA(LIST') THAN CREALISTA()
  ELSE SCRIVILISTA (I, CANCLISTA(POS', LIST) , POS))
- LEGGILISTA(POS', LIST') = IF LISTAVUOTA(LIST) AND POS'=POS THAN I ELSE LEGGILISTA(POS',LIST)

OSSERVATORI	COSTRUTTORE DI LIST'	
	CREALISTA()	SCRIVILISTA (I, LIST, POS)
LISTAVUOTO(LIST')	TRUE	FALSE
PRIMOLISTA(LIST')	ERROR	IF LISTAVUOTA(LIST) THAN POS ELSE PRIMOLISTA(LIST)
SUCCLISTA(POS', LIST')	ERROR	IF FINELISTA(POS',LIST') THAN ERROR ELSE POS'+1
PREDLISTA(POS', LIST')	ERROR	IF POS' = PRIMOLISTA(LIST') THAN ERROR ELSE POS'-1
FINELISTA(POS', LIST')	ERROR	IF SUCCLISTA(POS',LIST') != ERROR THAN FALSE ELSE TRUE
CANCLISTA(POS', LIST')	ERROR	IF LISTAVUOTA(LIST') THAN CREALISTA() ELSE SCRIVILISTA(I, CANCLISTA(POS',LIST), POS)
LEGGILISTA(POS', LIST')	ERROR	IF LISTAVUOTA(LIST') AND POS'=POS THAN I ELSE LEGGILISTA(POS', LIST)

Diagramma delle classi



PIANTINA DELLA MAPPA DI GIOCO