LA VILLA DI NATHAN

Team:

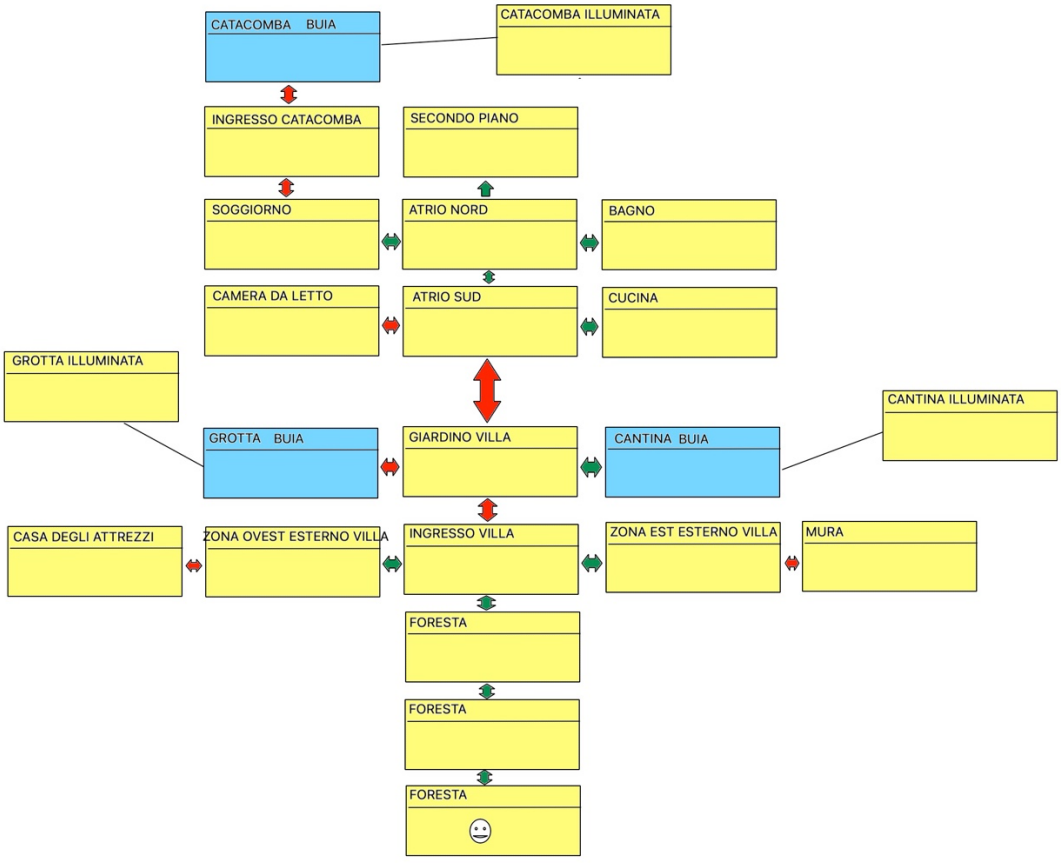
-Donato Tanieli, matricola 581524

Presentazione:

La Villa di Nathan, è un'avventura testuale di un cacciatore di tesori di nome Albert. Il protagonista si imbatterà nei vari pericoli ed enigmi nascosti nella villa alla ricerca del tesoro che nessuno aveva mai trovato. Per fare ciò il giocatore dovrà esplorare tutte le stanze e raccogliere oggetti che saranno utili nel corso della partita.

Le azioni da far compiere al protagonista dovranno essere scritte all'interno di una **JTextField** e basterà premere il tasto invio per darne la conferma.

Mappa:



Come si evince dalla mappa la maggior parte delle stanze da esplorare sono illuminate (stanze con lo sfondo giallo) ed è proprio in questa tipologia di stanza che sono presenti degli oggetti.
Oggetti che in una stanza al buio come (Cantina, Grotta, Catacomba) non risulterebbero possibili da vedere. Per cui questo è un esempio di come la stanza illuminata (LightRoom) erediti le caratteristiche della stanza buia (Room) con l'aggiunta degli oggetti. Per passare da una stanza buia all'altra sarà necessario possedere degli oggetti tali da poter illuminarla. Ogni stanza ha quattro possibili accessi rappresentati dalle frecce. La freccia rossa indica che il passaggio è bloccato e necessita di un oggetto per sbloccarlo. Ad esempio se mi trovo in INGRESSO VILLA l'accesso a nord risulta bloccato, ma una volta sbloccato posso andare e tornare tranquillamente nella stanza. Altre entità che compongono il progetto sono il **Player** che rappresenta il giocatore insieme al suo punteggio, **Inventory** che costituisce l'inventario del giocatore e **GameObject** che rappresenta l'oggetto del gioco. L'oggetto a sua volta può contenere altri oggetti e così via.

Parser:

Il parser rappresenta l'interprete dei comandi dell'utente ed è stato implementato in maniera tale che se si volesse ampliare l'avventura o creare una nuova avventura basandosi sulla stessa tipologia di comandi si può riutilizzare a prescindere dal gameplay. Il parser è in grado di riconoscere i comandi che sono stati scritti correttamente e che sono stati scritti nell'ordine corretto in quanto si parte dal presupposto che l'utente non scriva cose senza senso del tipo "nord martello". Si va, in primis, a scomporre le parole che l'utente ha scritto attraverso uno **StringTokenizer** usando dei delimitatori. Dopodiché si valuta la dimensione dell'ArrayList che contiene ciascuna parola e in base ad essa si intraprendono dei controlli specifici. Ad esempio se la dimensione è 1, allora si immagina che l'utente voglia andare in una certa direzione o compiere un'azione che non necessita l'interazione con uno o più oggetti. Si va a controllare a questo punto che il comando, preso singolarmente, rientri in quelli adatti a compiere quell'azione o movimento andando a verificare anche gli alias qualora il parser non trovi riscontro con il comando principale. Al contrario se la dimensione è due o più si immagina che l'utente voglia interagire con degli oggetti. In questo caso si parte sempre dalla verifica del comando principale e, se corretto, si passa alla verifica degli oggetti per capire se questi fanno parte effettivamente del gioco. Il parser può analizzare una frase composta da massimo 4 parole e, affinché sia valida, deve prevedere il comando corretto + oggetto1 + preposizione + oggetto2 (Es. accendi candela con fiammiferi). Tutto il resto risulta non comprensibile. Una volta terminati tutti i controlli, il risultato verrà incapsulato in un oggetto OutputParser che conterrà il comando con la lista degli eventuali oggetti.

Interfaccia grafica:

L'interfaccia grafica è definita all'interno del package gui ed è costituita da due classi: MenuGUI e GameGUI.
-MenuGUI contiene il menù principale tramite il quale si può cominciare una nuova partita, caricare una partita salvata, mostrare i punteggi o uscire.
-GameGUI è l'interfaccia del gioco vera e propria. Il testo viene visualizzato tramite una JTextPane, mentre il giocatore scrive il comando in una JTextField. È presente inoltre una JMenuBar in alto con due JMenu: File i cui JMenuItem sono "Nuova Partita", "Salva", "Carica", "Torna al menù principale" e Supporto che contiene un solo JMenuItem "Istruzioni".

Caricamento e salvataggio:

Il gioco prevede l'implementazione della funzione di caricamento e salvataggio di una partita. La classe che gestisce tale funzione è SaverLoaderClass che serializza/deserializza l'oggetto Game per salvarlo/caricarlo su/da un file.

Database:

Il gioco prevede anche l'utilizzo di un database per il salvataggio del punteggio del giocatore. Il salvataggio viene effettuato nel momento in cui si muore durante il gioco oppure si riesce a completarlo vincendo così la partita.
È stato sfruttato il Database Engine H2 utilizzato in modo embedded senza la necessità di installare un server.

Package game:

Il package game contiene diverse classi che rappresentano e gestiscono il gioco. Principalmente possiamo citare:
-la classe **Game** che rappresenta la struttura del gioco;
-la classe **GameManager** che costituisce la logica di tutto il gioco attraverso l'implementazione del metodo astratto executeCommand() dichiarato nella classe **GameVN;**
-la classe **GameCommunicator** che è la classe che si occupa della comunicazione tra il gioco e l'utente.

Conclusioni:

Il progetto rispecchia i canoni della programmazione ad oggetti sfruttando meccanismi di ereditarietà, incapsulamento e polimorfismo.
È stato progettato cercando di applicare quanto più possibile il concetto di astrazione in maniera tale da poter essere riutilizzato per ampliare il gioco stesso o creare un nuovo gioco rispecchiandone la stessa struttura.