DOCUMENTAZIONE CASO DI STUDIO

- Giorgia Nardelli, matricola 738746

- Francesco Papagno, matricola 735596

INDICE:

1) Avventura grafica:

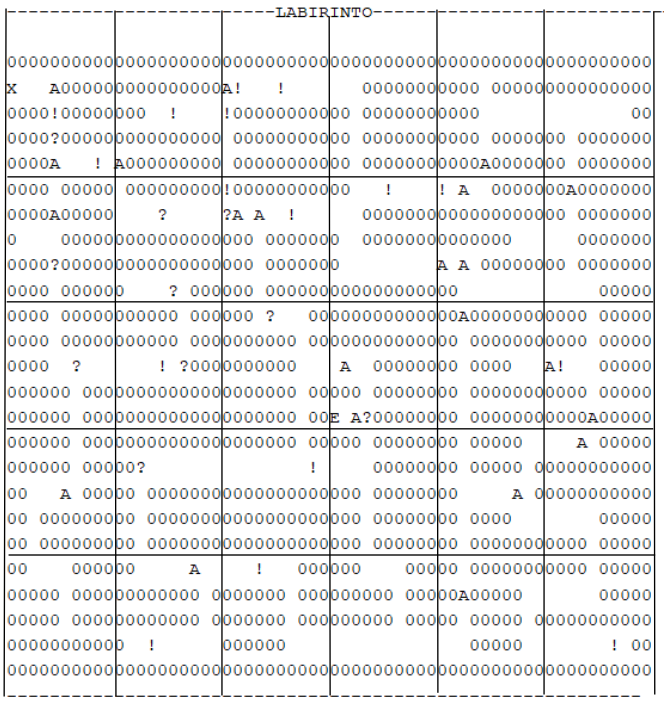
1.1) Introduzione e trama del gioco

Lo stile di gioco utilizzato è un'avventura testuale. L'avventura è svolta in prima persona e l'interazione tra il gioco e l'utente avviene mediante linea di comando, in particolare dall'inserimento di parole da parte dell'utente nel momento in cui gli viene posta una domanda. La trama è stata ideata prendendo spunto dalla serie tv "ShadowHunters". Nel gioco lo scopo del personaggio è quello di trovare la Coppa Mortale, un oggetto posizionato al centro del labirinto indicato con una 'E'. Sono presenti anche degli indovinelli e non è detto che il giocatore riesca a terminare il gioco in quanto sono presenti anche dei nemici (rappresentati con una 'A') che possono diminuire le statistiche del personaggio (stamina, salute e forza), fino anche ad azzerarle, anche se molto raramente. Inoltre, il personaggio lungo il percorso potrà trovare degli alimenti (rappresentati con '?') che, se decide di raccogliere vengono riposti nel suo inventario, se invece decide di raccoglierli e mangiarli aumenteranno la stamina e la salute del personaggio. Un altro bonus a disposizione dell'utente sono i doni che potranno aumentare la forza, la stamina o la salute se risponde correttamente all'indovinello che potrà trovare nel percorso, denotato con '!'.

Sapendo che non tutti possono essere pratici in questo tipo di giochi, abbiamo pensato di aggiungere una voce "Aiuto", disponibile solamente all'inizio del gioco, con qualche piccolo consiglio utile.

1.2) Mappa del gioco

La mappa a cui abbiamo pensato è un labirinto in cui l'utente può muoversi liberamente. Inoltre, la mappa è divisa in 30 zone.



Griglia zone

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30

1.3) Regole del gioco

Al protagonista è permesso potersi spostare all'interno del labirinto digitando 1 per spostarsi verso l'alto, 2 verso il basso, 3 verso destra e 4 verso sinistra.

2.1) Architettura del sistema

L'avventura, realizzata in Java, presenta un'architettura di sistema suddivisa in quattro parti fondamentali (package):

1. maingame: contiene il nucleo dell'avventura. In particolare, all'interno di questo package ci sono:
   - la classe Mappa in cui è definita la mappa del gioco;
   - la classe MainGame che contiene il metodo main da cui il gioco viene lanciato;
   - la classe Inventario che ha come attributi gli unici tre elementi che l'inventario può contenere, la capienza massima per ogni elemento all'interno dell'inventario è di cinque elementi;
   - la classe Giocatore che presenta come attributi la salute, la stamina, la forza e la posizione. All'interno di questa classe sono stati implementati tutti i metodi riferiti alle azioni che il giocatore può svolgere.
2. Controllo: contiene la classe Controlli al cui interno sono implementati i metodi che effettuano controlli sulle scelte fatte dall'utente.
3. Trama: contiene la classe Trama, per la maggior parte questa classe contiene stampe a video utili all'utente per conoscere dettagli narrativi.
4. ZoneMappa: contiene due classi:
   - la classe Zona è necessaria per definire le varie zone della mappa (sono 30);
   - la classe Posizione è utile per stabilire la Posizione del giocatore, difatti è anche un attributo del giocatore.

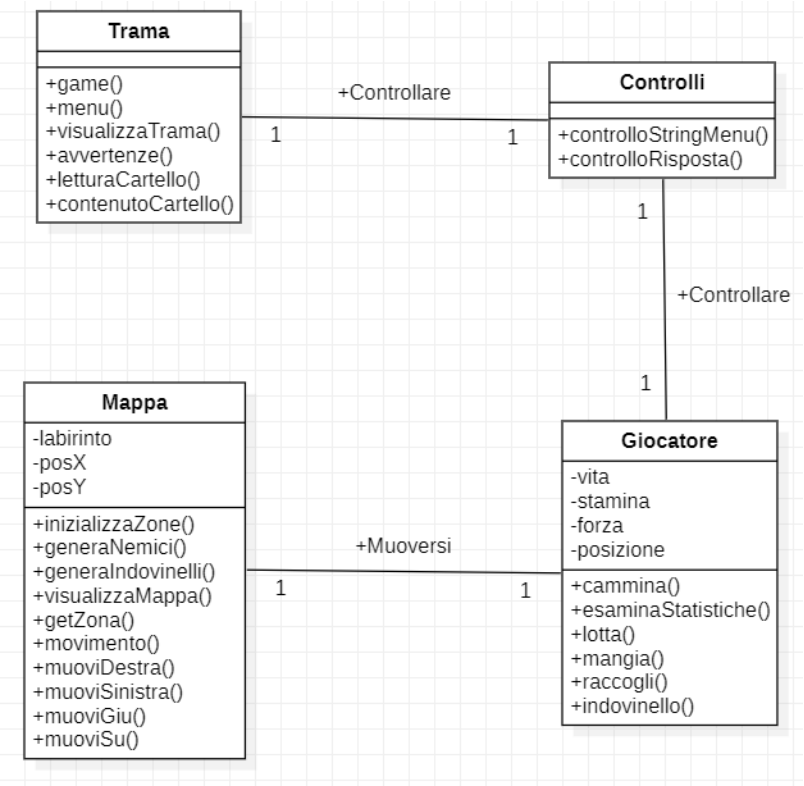
2.2) Dettagli di implementazione

All'interno del programma abbiamo deciso di utilizzare un file che memorizzi il contenuto della sezione *aiuto* del menù iniziale, mediante l'utilizzo della classe File di Java.

Inoltre, sono state utilizzate espressioni regolari all'interno di alcuni metodi della classe *Giocatore* e all'interno di entrambi i metodi della classe Controlli mediante l'utilizzo della classe *Pattern*. In particolare, è stato utilizzato il metodo compile() in cui il primo parametro di tale metodo indica quale modello viene cercato e il secondo parametro ha un flag per indicare che la ricerca non deve fare distinzione tra maiuscole e minuscole (*CASE_INSENSITIVE*).

In aggiunta, per evitare che un errore potesse bloccare e far terminare in modo anomalo il programma abbiamo utilizzato le eccezioni. Un'eccezione è un'istanza di java.lang.Throwable o di una sua sottoclasse. Le due sottoclassi più comuni sono java.lang.Error e java.lang.Exception. Una Exception indica una situazione che può essere gestita. Nel nostro caso abbiamo quindi utilizzato java.lang.Exception.

2.3) Diagramma delle classi



2.4) Specifica algebrica e struttura dati

LIST

Una List è un'interfaccia di Collection è una sequenza di elementi; quindi, ogni elemento ha una sua posizione. Inoltre, sono ammessi duplicati.

Oltre ai metodi previsti da Collection ci sono metodi per: accedere agli elementi tramite la loro posizione, cercare un oggetto e ottenere la posizione in cui si trova, iterare sulla sequenza, creare delle sottoliste.

Noi abbiamo usato, in particolare la struttura dati ArrayList, ovvero un'implementazione di List.

ArrayList è una struttura dati che fa parte di Collections Framework e può essere vista come simile a array e vettori.

ArrayList può essere percepito come un array dinamico che ti consente di aggiungere o rimuovere elementi dinamicamente.

L'ArrayList è classe utilizzata per la creazione dinamica di un array che contiene i riferimenti agli oggetti. Questo array potrebbe crescere di dimensioni come e quando richiesto.

SPECIFICA SINTATTICA:

TIPI: ArrayList, Index, elemento, boolean.

OPERATORI:

ArrayList () -> ArrayList

isEmpty (ArrayList) -> boolean

Contains (elemento) -> boolean

Add (index, elemento, ArrayList) -> ArrayList

Get (Index) -> elemento

Remove (Index) -> boolean

OSSERVAZIONI:

ArrayList: Restituisce un ArrayList vuoto.

isEmpty: per verificare se l'ArrayList è vuoto.

Contains: è usato per verificare se l'elemento è presente nell'ArrayList. E restituisce vero o falso.

Add: aggiunge l'elemento all'ArrayList

Get: è utilizzato per ottenere l'elemento all'indice specificato dell'ArrayList.

Remove: è utilizzato per rimuovere l'elemento all'indice specificato dell'ArrayList.

SPECIFICA SEMANTICA:

TIPI:

ArrayList: array dinamico che ti consente di aggiungere o rimuovere elementi dinamicamente.

Index: indica la posizione di un elemento all'interno dell'ArrayList.

Elemento: elemento contenuto nell'ArrayList

booleano = insieme dei valori di verità {vero, falso}.

isEmpty (ArrayList ())	TRUE
isEmpty (Add (i, Get(i), ArrayList ()))	FALSE
Contains (Get(i))	If e ∈ A then TRUE else FALSE
Add (i, Get(i), ArrayList ())	ArrayList()

SPECIFICA ALGEBRICA:

OSSERVATORI	COSTRUTTORI	
	ArrayList	Add (Index, element, ArrayList)
Remove (ArrayList, Index)	ERROR	ERROR
isEmpty (ArrayList)	TRUE	ERROR
Contains (element)	ERROR	ERROR
Get (Index)	ERROR	ArrayList