BUGFIX

***SHIFTED DEADLINE***

01.09.2021

GRUPPO “ ***LSD***”

AUTORI

Massafra Francesco

Ricchiuti Alberto

Scavo Beatrice

# INDICE

[Titolo del progetto 1](#_Toc39047411)

[INDICE 2](#_Toc39047412)

[Analisi 3](#_Toc39047413)

1. [Descrizione del sistema 3](#_Toc39047414)

[1.1 Requisiti Funzionali 3](#_Toc39047415)

1.2 Requisiti non Funzionali…………………………………………………………………... 4

2. System Design………………………………………………………………………………...….. 5

3. OO Design………………………………………………………………………………………... 6

4. Implementazione…………………………………………………………………………………. 9

5. Specifiche Algebriche……………………………………………………………………………11

[5. Presentazione della soluzione……………………………………………………………………13](#_Toc39047424)

# Analisi

## Descrizione del progetto

~PROTAGONISTA:

Un curioso amante della tecnologia.

~IDEA:

Un uomo sconosciuto appassionato di Informatica ritrova un misterioso gioco, un’avventura testuale, in un pc acquistato dal Deep Web.

Incuriosito, l’uomo inizia a giocare e molto presto si rende conto che non si tratta della classica avventura testuale ambientata in una terra fantastica popolata da elfi, ma che tra le righe di codice di quel programma si cela un macabro segreto…

Anni prima tre giovani studenti erano in procinto di consegnare un importante progetto, nel quale avevano messo tutta la loro energia, passione e anima. I tre aspiranti programmatori erano però in ritardo per la consegna e nel raggiungere l’università, distratti dalla fretta, furono investiti da un furgone e morirono sul colpo… Il loro pc però si salvò.

Lo sfortunato nerd è quindi di fronte ad un sadico videogame, infestato dagli spiriti di Kekko4K\_Designer, beaTester e AlbooDev, al quale non si può sottrarre, dal momento in cui la posta in gioco è ciò che il videogame gli ha già sottratto: l’anima.

Concludere l’oscura avventura attraverso i circuiti infetti del suo portatile e affrontare la versione del gioco corrotta dalle menti dei tre sventurati informatici è l’unico modo che l’uomo ha per salvarsi, ma ciò che si nasconde dietro il monitor lo cambierà per sempre.

1.1 Requisiti funzionali

L’applicazione, attraverso un menu principale, permette all’utente di:

* Iniziare una nuova partita:

inserendo “nuova partita”, l’applicazione chiede di dare un nome alla nuova avventura. Questo sarà lo stesso del file sul quale verranno caricati tutti i suoi salvataggi.

Si posso giocare contemporaneamente fino ad un massimo di 5 avventure.

* Salvare una partita:

durante l’avventura ci sono dei punti, i checkpoint, in cui il gioco salverà automaticamente. Questi si trovano in corrispondenza della fine di ogni capitolo.

* Caricare un salvataggio:

il comando “carica” permette di riprendere il gioco dal punto iniziale dell’ultimo capitolo che si stava giocando, nel caso in cui si sia usciti prima di terminarlo, dell’avventura di seguito specificata.

* Visualizzare e modificare i salvataggi:  
   si ha la possibilità di rinominare o cancellare un precedente salvataggio.
* Uscire dall’applicazione:

inserendo il comando “esci”, l’applicazione termina.

Durante il gioco, l’utente può:

* Visualizzare una guida ai comandi:  
   durante l’esecuzione del programma è possibile visualizzare la guida dei   
   comandi inserendo: *“help”*.
* Interagire con gli oggetti presenti nell’ambiente:

gli oggetti si possono esaminare, raccogliere e/o usare. Ciò consente di continuare con l’avventura.

* Mostrare l’inventario:

inserendo il comando “ inventario ” si può visualizzare il contenuto del proprio inventario (lo zaino), se il personaggio ne è in possesso. In caso contrario, un messaggio avverte l’utente della sua mancanza. Il giocatore può raccogliere e/o usare gli oggetti, solo se è munito di zaino.

* Uscire dall’avventura in corso:

per uscire durante l’esecuzione del gioco si deve inserire il comando “*#esci*”. Dopo la conferma di tale decisione, si viene riportati nel menu iniziale.

1.2 Requisiti non funzionali

L’applicazione dovrà essere eseguita attraverso il file.bat “Bugfix\_run”, che si trova all’interno della cartella del progetto.

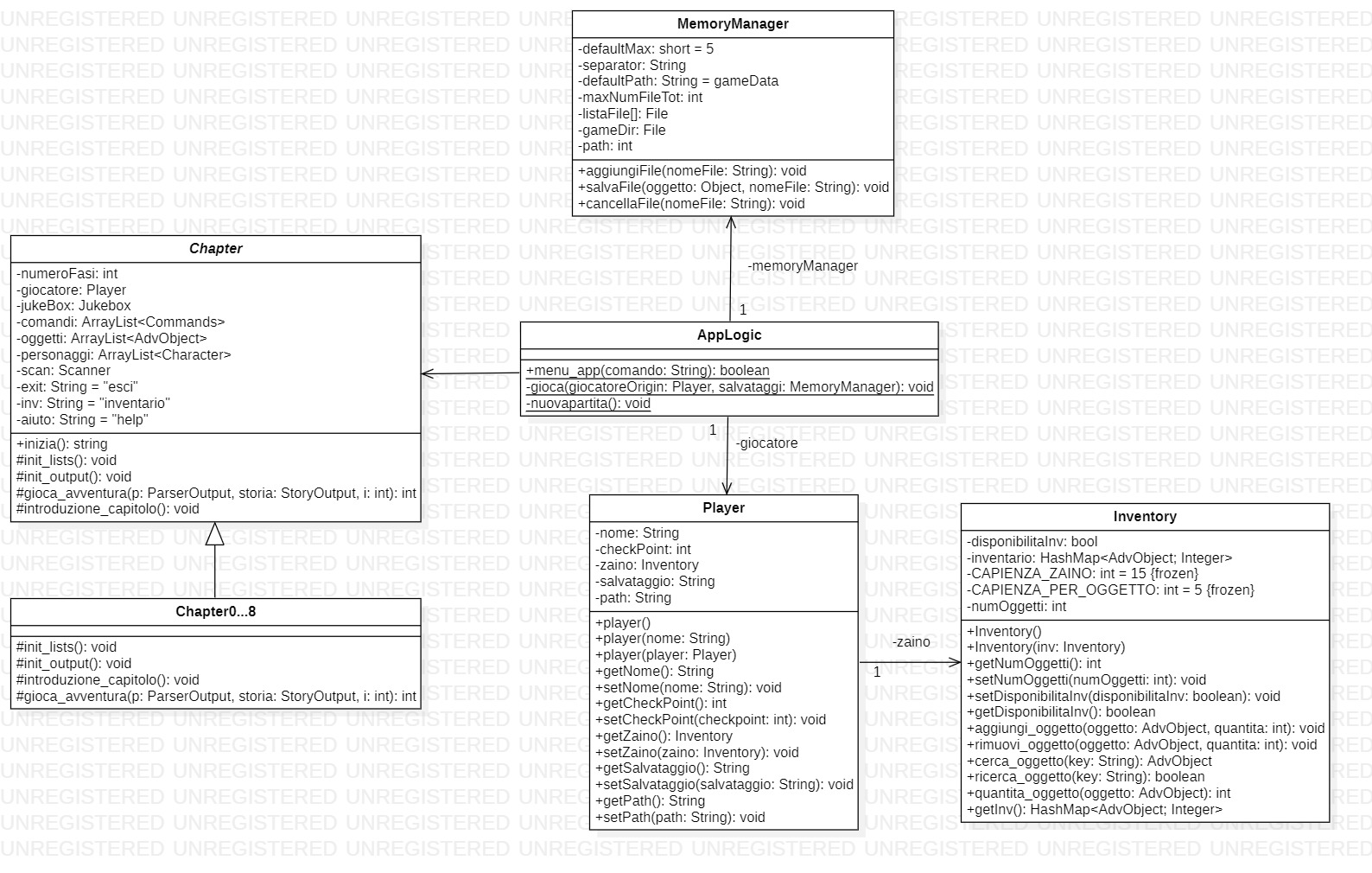
**2. System Design**

Suddivisione in packages

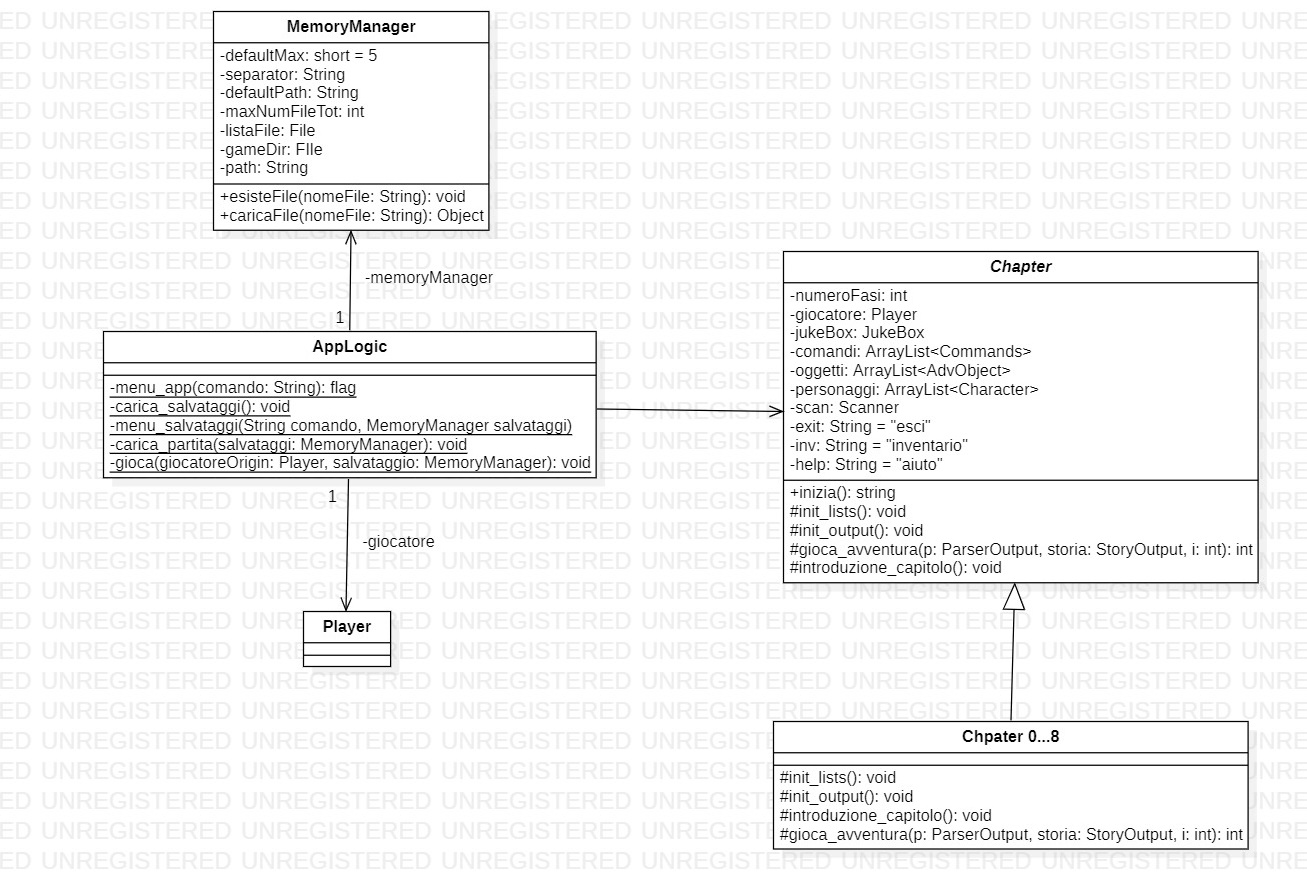
* Il package ***boundary*** contiene tutte le classi di tipo boundary, che quindi si occupano dell’interazione diretta con l’utente. All’interno sono presenti le classi:
  + *AppInput*, raccoglie dall’utente gli input utili all’applicazione e al gioco (quando non devono essere analizzati dal parser);
  + *AppOutput*, stampa a video gli output dell’applicazione;
  + *GameOutput*, stampa a video gli output generali del gioco, come ad esempio la guida ai comandi, i messaggi dell’inventario e le possibili schermate;
  + *MinigameOutput*, si occupa di stampare gli output relativi ad enigmi, ostacoli ed altri elementi dei minigame delle tre bossfight;
  + *StoryOutput*, fa visualizzare i dialoghi della storia dell’avventura.
* Il package ***control*** contiene la classe *AppLogic,* che gestisce la logica dell’applicazione. Quindi della creazione, caricamento e salvataggio delle partite.
* Nel package ***control.game*** c’è l’avventura del gioco. *Chapter* è la classe astratta da cui ereditano i capitoli in cui viene suddivisa l’avventura: *Chapter0*, *Chapter1*, *Chapter2*, *Chapter3*, *Chapter4*, *Chapter5*, *Chapter6*, *Chapter7*, *Chapter8*.
* Il package **entity** contiene le classi di tipo entità, ovvero le classi che si occupano della persistenza dei dati, che sono:
  + *AdvObject*, rappresenta un oggetto nell’avventura;
  + *Character,* rappresenta i personaggi che si possono incontrare durante l’avventura;
  + *CommandType,* rappresenta i tipi di comandi che è possibile utilizzare;
  + *Commands*, rappresenta il comando;
  + *Inventory*, rappresenta l’inventario;
  + *Player*, rappresenta il giocatore.
* Il package ***utils*** contiene le classi con funzioni specifiche per il corretto uso del programma. Sono presenti i seguenti file:
  + *Jukebox*, consente la riproduzione della musica;
  + *MemoryManager*, gestisce i file e i salvataggi;
  + *Parser* e *ParserOutput*, analizzano i comandi immessi dall’utente.
* Il package ***windows*** contiene interfacce grafiche realizzate con le librerie Swing: *Agreement*, *Balance*, *BurningCastle*, *Collapse*, *Drink*, *Home*, *Padlock*, *SimulatedError*.

**3. OO Design \***

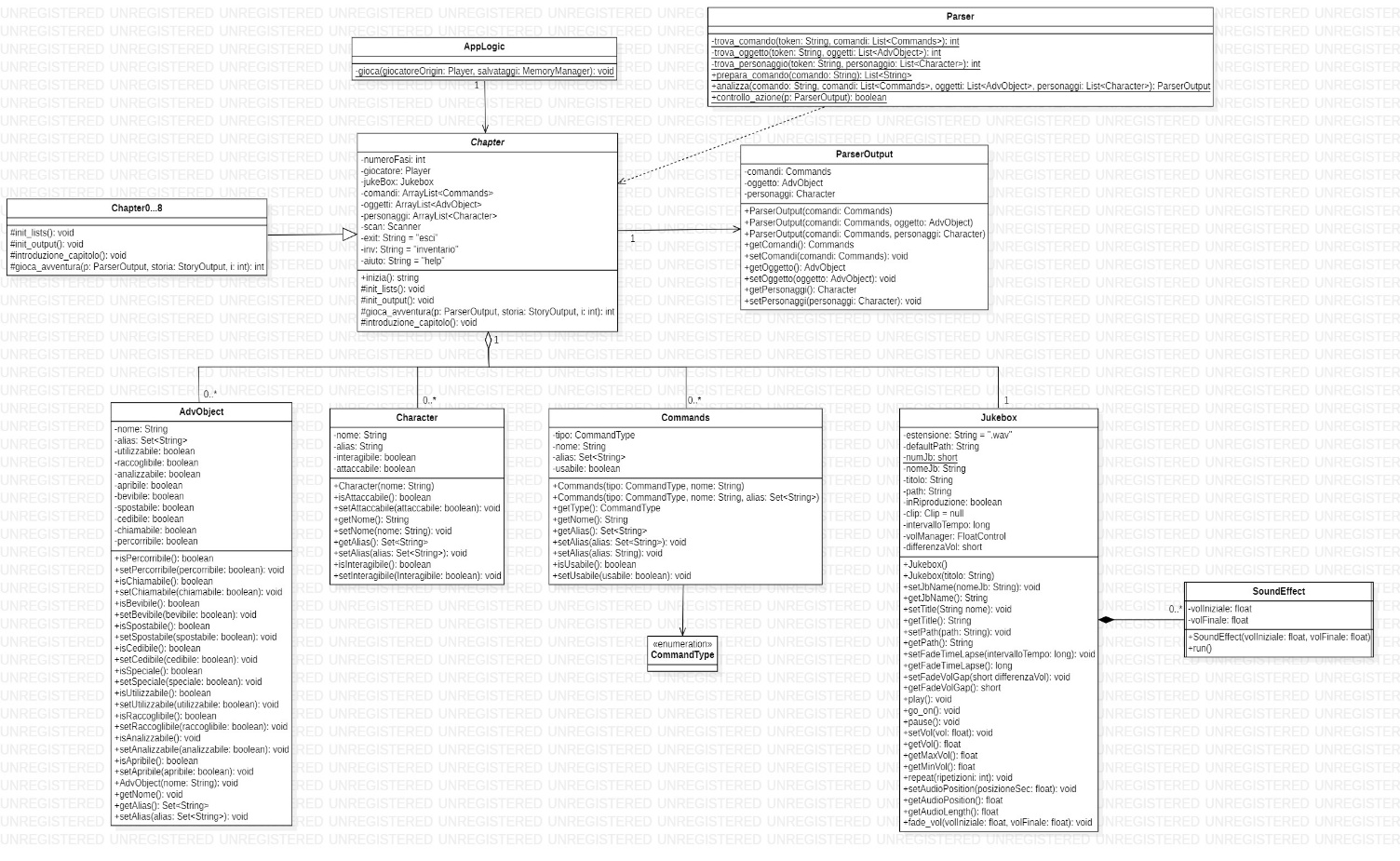
* Nuova partita



* Carica partita



* Gioca



**\* le immagini sono salvate nella cartella “diagrammi”**

**4. Implementazione**

Scelte progettuali

La storia di “Bugfix” segue un’idea ben definita:

il giocatore viene “forzato” a partecipare a qualcosa a cui non vuole prendere parte e costretto a seguire quello che è un cammino determinato a priori, con regole a lui completamente sfavorevoli che mirano a farlo perdere a tutti i costi.

L’implementazione del gioco rispetta questo principio:

l’assenza di una mappa e l’impossibilità di scelta di muoversi liberamente tra una stanza e l’altra, ricreano l’idea di prigionia e fatalismo.

Inoltre sono stati inseriti appositamente in più capitoli eventi che portano alla morte del protagonista (riprendendo, in tale caso, dall’inizio del capitolo interessato) e, se non vengono raccolti specifici oggetti, anche ad uscire sconfitto dal gioco. Per questo è stata inserita la possibilità di giocare più avventure in contemporanea, con file rispettivamente dedicati.

Dettagli implementativi

Ogni capitolo eredita ed implementa dalla classe astratta *Chapter*, tutti i metodi relativi alla loro inizializzazione e svolgimento.

Alcuni capitoli presentano al loro interno classi innestate, come il *Chapter3*.

Nel *Chapter3*, nella classe *Riddle*, le classi *SolutionGetter* e *Hourglass* estendono *Thread*. La prima si occupa di interrompere il tempo in caso di risposta corretta, la seconda di interrompere il gioco se si è passato il tempo limite per rispondere.

Il *Chapetr7* è dedicato quasi esclusivamente alla narrazione dei fatti della storia e allo svolgimento del minigioco (BossFight), che si va ad inserire nello sviluppo del capitolo generale.

La classe *Jukebox* eredita da *Thread*, senza che venga implementato il metodo *run*(). Utilizza inoltre una classe innestata che a sua volta eredita da *Thread*, per l’effetto dissolvenza della colonna sonora del gioco e delle musiche aggiuntive, al fine di farla eseguire in contemporanea con il gioco.

I File sono stati utilizzati in *MemeoryManager* per raccogliere e caricare i dati delle avventure in corso.

Per quanto riguarda la struttura dell’inventario, è stato deciso di utilizzare un Hashmap<*AdvObject*, *Integer*>, dove la chiave, *AdvObject,* è l’oggetto dell’avventura, mentre il valore *Integer*, indica la quantità di cui se ne è a disposizione. La stampa è stata implementata attraverso un’Espressione Lambda, nella quale viene usata l’interfaccia funzionale *BiConsumer*, che accetta in input due argomenti e che rispetto ad altre interfacce funzionali opera attraverso il side\_effect.

Per l’assegnazione degli alias degli oggetti, dei personaggi e dei comandi è stato utilizzato, invece, un Set<String>.

Tra le finestre implementate, i disegni di *Balance*, *BurningCastle*, *Collapse*, *Drink* e *Home* sono state realizzati dal team.

*Agreement* e *SimulatedError* hanno il compito di simulare:

il primo, l’accettazione dei termini e condizioni per usufruire dei servizi offerti dal gioco;

il secondo, il caricamento e la risoluzione degli errori generati da un programma in esecuzione.

Entrambe non possono essere chiuse.

*Padlock* invece rappresenta il lucchetto che va aperto con la giusta combinazione per poter proseguire nel gioco.

Dove è stato possibile, sono state inserite Espressioni Lambda negli ActionListener dei jButton.

**5. Specifiche algebriche**

SPECIFICA SINTATTICA

Tipi:

Set, Object, booleano, intero

Operazioni:

*new*() Set Crea un nuovo Set

*setObject*(Set, Object)  Set Aggiunge un elemento al Set

*clear*(Set)  Set Elimina tutti gli elementi dal Set

*contains*(Set, Object)  booleano Stabilisce se un elemento è nel Set

*isEmpty*(Set)  booleano Stabilisce se il Set è vuoto

*removeObject*(Set, Object)  Set Elimina un elemento dal Set

*dimension*(Set)  intero Restituisce il numero di elementi del Set

*equal*(Set, Set)  booleano Stabilisce se due Set sono equivalenti

SPECIFICA SEMANTICA

DeclareS,S1: Set; o, e : Object; i: integer

*dimension*(*new*()) = 0;

*dimension*(*setObject* (S,e)) = *dimension*(S)+1;

*isEmpty*(*new*()) = vero;

*isEmpty*(*setObject* (S,e)) = falso;

*remove*(*new*(),o)=errore;

*removeObject*(*setObject* (S,e),o) = if *isEmpty*(S) then (if (e=o) then *new*() else errore) else (if(e=o) then S else *removeObject*(S,o));

*contain*(*new*(),o) = falso;

*contain*(*setObject*(S,e),o) = if (o=e) then vero else *contain*(S,o);

*clear*(*new*()) = *new*();

*clear*(*setObject*(S,e)) = *new*();

SPECIFICA SEMANTICA

|  |  |  |
| --- | --- | --- |
| Osservazioni | Costruttori di S’ | |
| new() | setObject(S,e) |
| dimension(S’) | 0 | dimension(S)+1 |
| isEmpty(S’) | vero | falso |
| removeObject(S’,o) | errore | if isEmpty(S) then (if (e=o) then new() else errore) else (if(e=o) then S else removeObject(S,o)) |
| contain(S’,o) | falso | if (o=e) then vero else contain(S,o); |
| clear(S’) | new() | new() |

OSSERVAZIONE BINARIA

|  |  |  |
| --- | --- | --- |
| Costruttore di S1 | Costruttore di S | |
| new() | setObject(S1,e) |
| new() | vero | falso |
| setObject(S,o) | falso | If(e=o) then equal(S,S1) else falso |

SPECIFICHE DI RESTRIZIONE

Restrizioni:

*removeObject*(*new*(),o) = errore;

*removeObject*(*setObject*(S,e),o) = if (*isEmpty*(S) and e != i) then errore;

**6. Comandi per completare l’avventura**

Per il corretto uso dei comandi, l’input deve essere composto da <azione> <oggetto/personaggio> oppure, in alcune situazioni, da <azione>. L’azione deve essere scritta in forma imperativa, come ad esempio: osserva, cammina, usa spada, prendi zaino.

Nel momento in cui tra un dialogo e l’altro compare il simbolo “#”, per proseguire, è necessario premere invio.

Per uscire dal gioco, bisogna immettere “#esci”, mentre per richiamare la guida ai comandi “help”.

Attraverso il comando “inventario”, è possibile consultare l’inventario.

**Capitolo 0**

1. Accettare la spunta alla finestra che compare
2. Inserire il nome
3. Attesa del caricamento del capitolo successivo

**Capitolo 1**

1. Tocca pavimento;
2. collega cavi;
3. osserva muro;
4. spingi muro;
5. guarda stanza;
6. prendi zaino;
7. apri baule;
8. raccogli legnetto;
9. osserva quadro;
10. sposta quadro;
11. raccogli chiave;
12. apri porta;
13. usa chiave;
14. percorri corridoio.

**Capitolo 2**

1. Osserva;
2. analizza masso;
3. raccogli pietra;
4. guarda spazzatura;
5. raccogli elastico;
6. prendi ghianda;
7. costruisci fionda;
8. percorri ponte;
9. apri porta;
10. osserva albero;
11. sposta foglie;
12. esamina cosa;
13. parla con lo scoiattolo;
14. dai ghianda;
15. ripara sequenza;
16. percorri corridoio.

**Capitolo 3**

1. Prosegui;
2. soluzioni indovinelli: 1. Uovo; 2. Tempo; 3. Impronte;
3. avvicinati;
4. corri;
5. osserva ceneri;
6. leggi giornale;
7. osserva stanza;
8. apri baule;
9. raccogli elmo;

**Capitolo 4**

1. Attraversa porta;
2. si;
3. controlla computer;
4. osserva;
5. raccogli moneta;
6. continua;
7. ripara computer;
8. spegni&&riaccendi computer // EASTER EGG
9. continua;
10. guarda;
11. esamina mani;
12. raccogli monete;
13. prosegui;
14. apri porta.

**Capitolo5**

1. Osserva;
2. tocca porta;
3. guarda stanza;
4. scrivere qualsiasi comando;
5. controlla slot machines;
6. usa slot machines;
7. osserva tavoli;
8. cerca zaino;
9. guarda bar;
10. suona campanello;
11. accetta;
12. si;
13. bevi drink;
14. osserva bicchiere;
15. esamina fiches;
16. parla con beatester;
17. si;
18. scrivere i numeri letti sulle fiches;

**Capitolo 6**

1. Osserva baule;
2. raccogli armatura;
3. esamina stanza;
4. prendi torcia;
5. prosegui;
6. usa torcia;
7. controlla muro;
8. osserva;
9. vai avanti;
10. osserva pilastro destro;
11. esamina donna;
12. analizza sfera;
13. raccogli sfera;
14. osserva pilastro centrale;
15. osserva pilastro sinistro;
16. osserva grata;
17. apri lucchetto;
18. combinazione: 16 18 13;

**Capitolo 7**

1. Chiama ascensore;
2. soluzione impiccato: anche i draghi piangono.

**Capitolo** **8**

1. Scappa;
2. raccogli spada;
3. parla con alberto;
4. cedi sogno;
5. attacca drago.