**Documentazione del Progetto: Avventura Testuale**

**"Jason Bird e il Teschio di Cristallo"**

Roberto Rotunno mat.758448 Nicola Mastromarino mat.757709

**Immagine che contiene testo, emblema, logo, simbolo

Descrizione generata automaticamente**

Sommario

[**Breve Descrizione Generale del Caso di Studio** 3](#_Toc144310314)

[TRAMA 3](#_Toc144310315)

[REGOLE 4](#_Toc144310316)

[STRUTTURA 4](#_Toc144310317)

[TEST 4](#_Toc144310318)

[LISTA DEI COMANDI 5](#_Toc144310319)

[**Mappa** 7](#_Toc144310320)

[**Diagramma delle Classi e Descrizione** 8](#_Toc144310321)

[**Specifica Algebrica di una Struttura Dati Utilizzata** 9](#_Toc144310322)

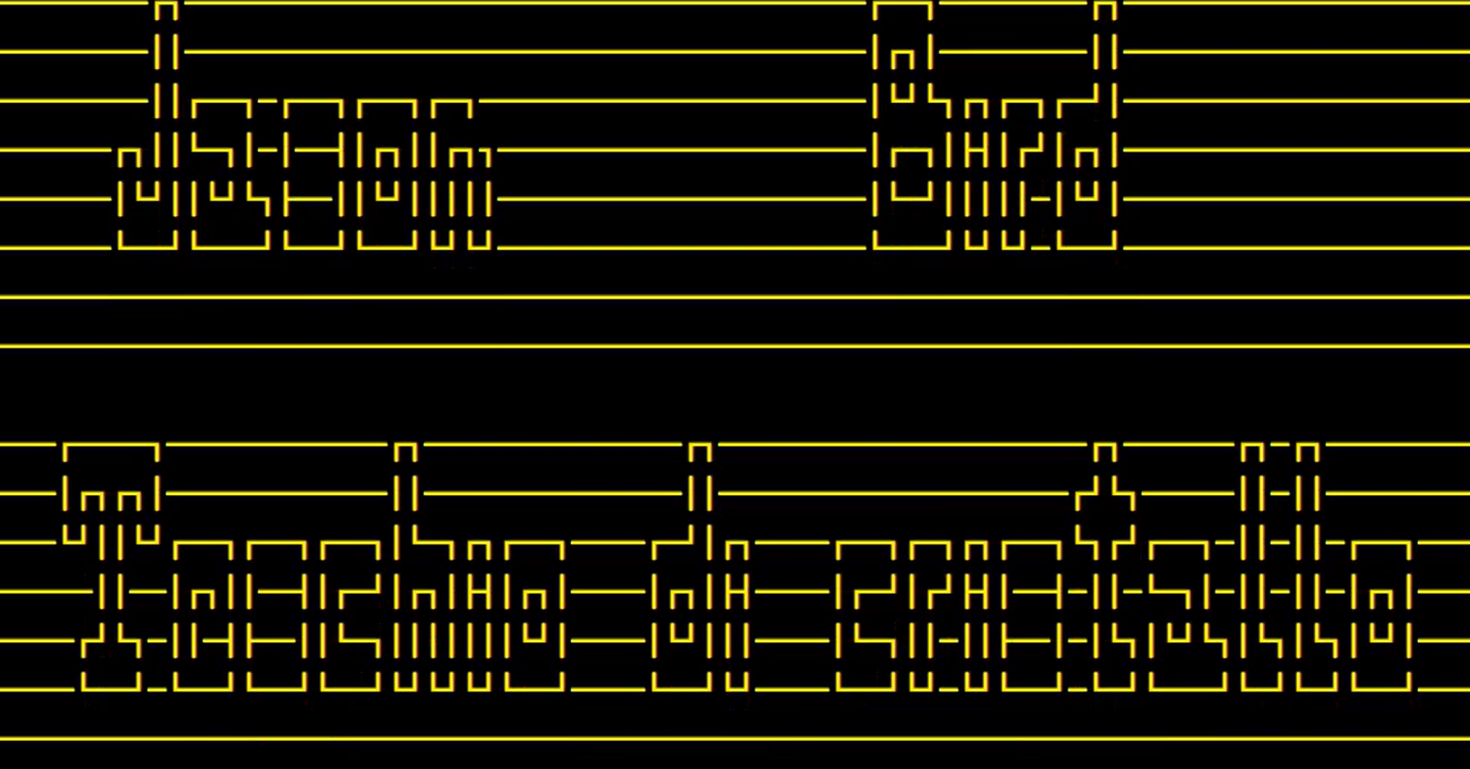
[SPECIFICA SINTATTICA 9](#_Toc144310323)

[SPECIFICA SEMANTICA 10](#_Toc144310324)

[SPECIFICA DI RESTRINZIONE 10](#_Toc144310325)

[**Applicazione degli Argomenti del Corso** 11](#_Toc144310326)

[ATTENZIONE: Spoiler Avanti! 12](#_Toc144310327)

****

# **Breve Descrizione Generale del Caso di Studio**

Obiettivo del Gioco

## TRAMA

Il gioco è una rivisitazione di Indiana Jones, il famoso film di avventura e azione. Il protagonista del gioco è Jason Bird, un archeologo e avventuriero che si trova a dover recuperare un teschio di cristallo, un antico manufatto misterioso e potente. Il gioco è ambientato in un luogo esotico e pericoloso, la giungla. Il gioco ha lo scopo di far vivere al giocatore un’esperienza avvincente e divertente, mettendolo alla prova con enigmi, trappole e sorprese.

Immerso in un'avventura affascinante, dovrai guidare Jason attraverso intricati percorsi, risolvere enigmi e superare ostacoli per raggiungere l'ambita reliquia. Nel corso del tuo viaggio, dovrai esplorare stanze segrete e svelare indizi per avanzare nel gioco.

Il teschio di cristallo non è solo un oggetto di valore inestimabile, ma è anche legato a leggende antiche e poteri misteriosi. La tua missione è non solo recuperarlo, ma anche uscire dal tempio prima che la maledizione ti uccida.

Mentre guidi Jason attraverso questa epica avventura, dovrai prendere decisioni sagge, sfidare il tempo e dimostrare abilità di pensiero logico per trionfare nei confronti delle sfide che si presenteranno lungo il cammino. Il destino del teschio di cristallo e la tua reputazione come coraggioso esploratore saranno nelle tue mani.

Preparati a un'esperienza coinvolgente e mozzafiato mentre immergi te stesso in "Jason Bird e il Teschio di Cristallo" e affronti l'ignoto.

## REGOLE

Il gioco è un’avventura testuale, ovvero un tipo di gioco in cui il giocatore interagisce con il mondo del gioco attraverso dei comandi scritti in linguaggio naturale.

Il gioco risponde al giocatore descrivendo le situazioni, gli oggetti, i personaggi e le azioni che avvengono nel gioco. Il giocatore può esplorare le varie stanze del gioco, raccogliere e usare oggetti e risolvere enigmi.

Il gioco ha una trama lineare, ma il giocatore può scegliere diversi modi di affrontare le sfide che incontra. Il gioco termina quando il giocatore riesce a recuperare il teschio di cristallo e a uscire dal tempio, oppure quando il giocatore muore.

## STRUTTURA

Il gioco è composto da diverse componenti principali, tra cui:

Stanze: sono le unità di base del gioco, rappresentano i luoghi che il giocatore può visitare. Ogni stanza ha un nome, una descrizione, una lista di oggetti e una lista di uscite verso altre stanze. Alcune stanze hanno anche degli eventi speciali che si attivano quando il giocatore entra o fa determinate azioni.

Oggetti: sono gli elementi che il giocatore può raccogliere, usare e esaminare. Ogni oggetto ha un nome, una descrizione e una funzione. Alcuni oggetti sono indispensabili per la risoluzione degli enigmi, altri sono opzionali.

Eventi: sono le situazioni dinamiche che si verificano nel gioco in risposta alle azioni del giocatore. Ogni evento ha una condizione di attivazione, una descrizione e un effetto. Alcuni eventi sono parte della trama principale, altri sono secondari o casuali. Un esempio di evento è il giocatore che trova un passaggio segreto o una porta che si apre.

Queste componenti sono collegate tra loro da delle relazioni che definiscono la logica del gioco. Ad esempio, le stanze sono collegate da delle uscite che indicano le direzioni possibili per spostarsi, questi collegamenti possono variare in base alle azioni del giocatore; Gli oggetti sono collegate da delle funzioni che indicano come possono essere usati; In caso di eventi vengono innescate delle condizioni che provocano un effetto sulla trama del gioco.

## TEST

Il gioco è stato testato usando diversi strumenti e metodi, tra cui:

- Debugger: uno strumento per l'analisi e la correzione degli errori del codice Java, che permette di eseguire il gioco passo per passo, controllare i valori delle variabili e interrompere l'esecuzione in caso di eccezioni.

- Playtesting: un metodo per il testing manuale del gioco, che consiste nel far giocare il gioco a dei tester umani, che possono fornire feedback sul gameplay, sulla difficoltà, sulla trama e sull'interfaccia del gioco.

I test hanno permesso di individuare e risolvere diversi problemi del gioco, tra cui:

- Errori di sintassi o di logica nel codice Java, che causavano il blocco o il malfunzionamento del gioco.

- Errori di ortografia o di grammatica nei testi del gioco, che rendevano il gioco meno credibile o comprensibile.

- Errori di coerenza o di continuità nella trama o nelle descrizioni del gioco, che creavano incongruenze o contraddizioni nel mondo del gioco.

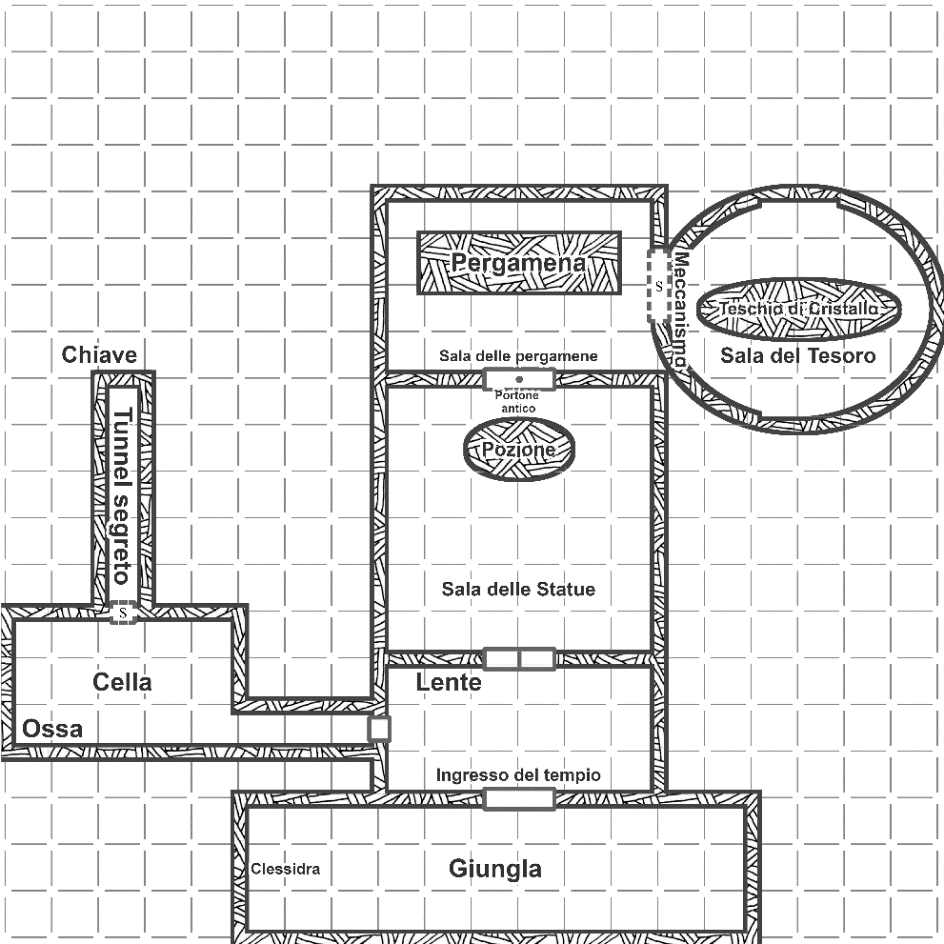
## LISTA DEI COMANDI

Per quanto riguarda la lista dei comandi nel gioco sono presenti sia i comandi di movimento che i comandi d’azione.

I comandi sono i seguenti:

| **Comando** | **Alias** | **Descrizione** |
| --- | --- | --- |
| Leggi | leggi, leggere | Leggi un oggetto o un testo. |
| Nord | n, N, nord, NORD | Sposta il personaggio a nord. |
| Inventario | inv, i, I, inventario | Mostra l'inventario del personaggio. |
| Sud | s, S, Sud, SUD, sud | Sposta il personaggio a sud. |
| Est | e, E, Est, EST, est | Sposta il personaggio a est. |
| Ovest | o, O, Ovest, OVEST, ovest | Sposta il personaggio a ovest. |
| Esci | end, fine, esci, ... | Termina il gioco e esce. |
| Osserva | guarda, vedi, osserva, ... | Osserva un oggetto o l'ambiente circostante. |
| Raccogli | prendi, raccogli | Raccogli un oggetto e aggiungilo all'inventario. |
| Apri | apri, aprire, aperto | Apri un oggetto o una porta. |
| Premi | spingi, attiva, premi, ... | Premi un oggetto o un pulsante. |
| Bevi | ingurgita, sorseggia, ... | Bevi da una bottiglia o un contenitore. |
| Usa | utilizza, impiega, usa, ... | Usa un oggetto o interagisci con esso. |

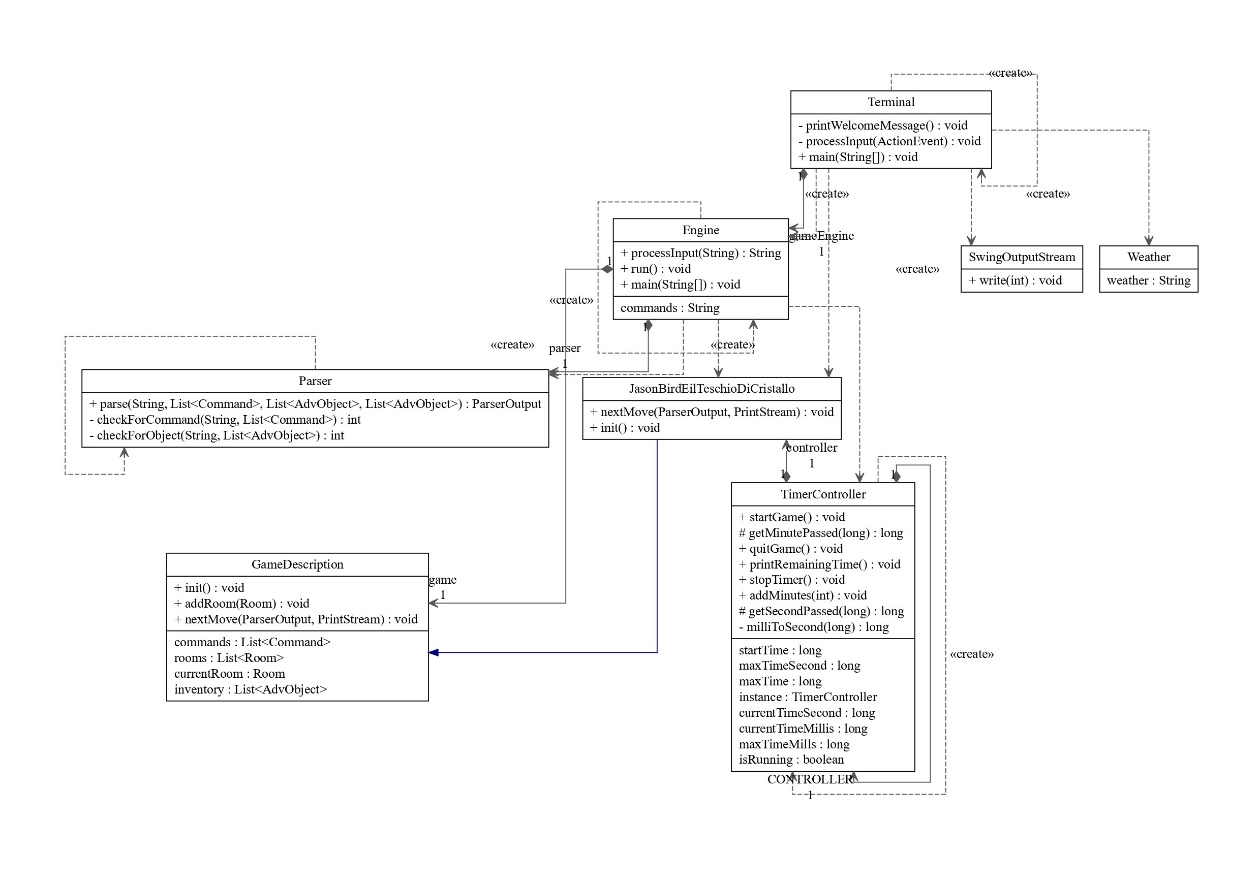
# **Mappa**



Come possiamo notare dalla mappa all’interno di ogni stanza sono presenti degli oggetti che il protagonista può utilizzare o aggiungere all’inventario. Le porte sono contrassegnate o da una S se è una porta segreta, o dal bianco che indica che il passaggio è libero o da un punto che indica che la porta richiede la chiave. Inoltre, sono presenti oggetti con il quale il giocatore può interagire ed aggiungere nell’inventario.

# **Diagramma delle Classi e Descrizione**

In questa sezione, presentiamo il diagramma delle classi che illustra le relazioni e la struttura delle principali classi coinvolte nel nostro progetto "Jason Bird e il Teschio di Cristallo". Il diagramma offre una visione visuale delle interazioni tra le classi chiave all'interno del sistema.



Spiegazione del Diagramma:

Engine: Questa classe rappresenta il motore del gioco e contiene metodi per processare l'input del giocatore, gestire il flusso del gioco e coordinare le altre classi coinvolte.

GameDescription: La classe GameDescription gestisce la descrizione del gioco, inizializzando le stanze e gestendo il flusso del gioco. Ha dipendenze da altre classi, come Room.

JasonBirdEilTeschioDiCristallo: Questa classe centrale gestisce il progresso del gioco e le azioni successive del giocatore.

Parser: La classe Parser analizza l'input del giocatore per interpretare i comandi e gli oggetti inseriti. Ha dipendenze da classi come Command e AdvObject.

SwingOutputStream: Questa classe personalizzata gestisce la stampa dei messaggi grafici nel gioco, utilizzando Swing per l'output visuale.

Terminal: La classe Terminal gestisce l'input e l'output per il giocatore sulla finestra Swing, consentendo l'interazione con il gioco tramite la console.

TimerController: Questa classe gestisce la logica del tempo nel gioco, consentendo il monitoraggio e il controllo del timer di gioco.

Weather: Rappresenta la gestione delle condizioni ambientali nel gioco.

# **Specifica Algebrica di una Struttura Dati Utilizzata**

## SPECIFICA SINTATTICA

sorts: inventory, AdvObject, boolean

operations:

newinventory() -> inventory

getInventory(inventory) -> List<AdvObject>

setInventory(inventory, list: List<AdvObject>) -> inventory

add(inventory, AdvObject) -> inventory

remove(inventory, AdvObject) -> inventory

isEmpty(inventory) -> boolean

inventory: Il tipo di dato che rappresenta l'inventario.

AdvObject: Il tipo di dato che rappresenta un oggetto nell'inventario.

boolean: Il tipo di dato che rappresenta un valore booleano (true o false).

Le operazioni definite sono:

newinventory() -> inventory: Crea un nuovo inventario restituendo un oggetto di tipo inventory.

getInventory(inventory) -> List<AdvObject>: Restituisce la lista di oggetti nell'inventario.

setInventory(inventory, list: List<AdvObject>) -> inventory: Imposta la lista di oggetti nell'inventario con la lista specificata.

add(inventory, o: AdvObject) -> inventory: Aggiunge un oggetto o all'inventario e restituisce l'inventario aggiornato.

remove(inventory, o: AdvObject) -> inventory: Rimuove un oggetto o dall'inventario e restituisce l'inventario aggiornato.

isEmpty(inventory) -> boolean: Restituisce true se l'inventario è vuoto, altrimenti false.

## SPECIFICA SEMANTICA

declare

Inv1, inv2: inventory;

Obj1, obj2: AdvObject;

List: list<AdvObject>

getInventory(newinventory) = list

getInventory(add(inv1, obj1), list)) = list + [obj1]

remove(add(inv1, obj1), obj2) = list + [obj1] - [obj2]

isEmpty(newinventory()) = true

isEmpty(add(inv1, obj1)) = false

setInventory(newinventory(), list) = inv1

setInventory(add(inv1, obj1), list) = inv2

getInventory(newinventory) = []: L'inventario appena creato è vuoto, quindi getInventory restituisce una lista vuota.

getInventory(setInventory(inv, list)) = list: Se imposti l'inventario inv con una lista list, otterrai la stessa lista tramite getInventory.

getInventory(add(inv, obj)) = list + [obj]: Se aggiungi un oggetto obj all'inventario inv, otterrai la lista precedente più l'oggetto aggiunto.

getInventory(remove(inv, obj)) = list - [obj]: Se rimuovi un oggetto obj dall'inventario inv, otterrai la lista precedente meno l'oggetto rimosso.

isEmpty(newinventory) = true: L'inventario appena creato è vuoto, quindi isEmpty restituisce true.

isEmpty(add(inv, obj)) = false: Se aggiungi un oggetto all'inventario inv, l'inventario non è più vuoto, quindi isEmpty restituisce false.

isEmpty(remove(inv, obj)) = true Se obj non in list, altrimenti false: Se rimuovi un oggetto obj dall'inventario inv e l'oggetto non è nella lista, allora l'inventario sarà vuoto e isEmpty restituirà true. In caso contrario, l'inventario non sarà vuoto e isEmpty restituirà false.

## SPECIFICA DI RESTRINZIONE

restrictions

remove(newinventory, obj) = error

# **Applicazione degli Argomenti del Corso**

Di seguito è spiegato come abbiamo applicato i vari argomenti del corso di Metodi Avanzati di Programmazione al nostro caso di studio:

File: Abbiamo sfruttato le operazioni di lettura su file per creare una pergamena nel gioco. Questa pergamena è un elemento chiave della trama, contenente indizi critici per il recupero del teschio di cristallo. Abbiamo adottato il cifrario di Cesare come metodo di crittografia per rendere la pergamena più interessante e coinvolgente. Durante il processo di creazione della pergamena, ogni carattere del testo è stato traslato di un passo secondo l'algoritmo del cifrario di Cesare. Questo ha aggiunto un elemento di sfida per i giocatori, poiché devono decifrare la pergamena per ottenere gli indizi necessari. Recuperando la lente dell’antico sole. L'implementazione del cifrario di Cesare ha coinvolto l'applicazione di concetti di manipolazione delle stringhe e dell'aritmetica dei caratteri. Questo approccio ha unito con successo la gestione dei file con l'elaborazione dei dati in modo creativo e coinvolgente.

REST: Abbiamo abbracciato l'architettura REST per arricchire l'esperienza di gioco con dati ambientali realistici. Abbiamo utilizzato chiamate REST per acquisire informazioni sull'altezza e sul ciclo giorno/notte di un monte in Asia, rendendo l'ambientazione del gioco più dinamica e coinvolgente. Abbiamo implementato una connessione alle API REST di un servizio meteorologico per ottenere dati di altezza che sono stati integrati nel gioco. In base a questi dati, abbiamo reso possibile descrivere in modo accurato la topografia dei luoghi visitati all'interno del gioco, offrendo ai giocatori una rappresentazione realistica dell'ambiente. Inoltre, abbiamo integrato dati sul ciclo giorno/notte tramite chiamate REST a un servizio che fornisce informazioni sulle fasi del giorno. L'implementazione di queste chiamate REST ha richiesto la comprensione dei concetti di comunicazione HTTP e gestione degli errori delle chiamate. In sintesi, l'integrazione di chiamate REST per acquisire dati ambientali ha arricchito l'esperienza di gioco, creando un ambiente dinamico e realistico.

Lambda Expression: Abbiamo sfruttato le lambda expressions di Java per semplificare il processo di ottenere una lista di comandi disponibili nel gioco. In particolare, abbiamo applicato una lambda expression in combinazione con gli stream di Java per ottenere una rappresentazione dei nomi dei comandi.

Nella seguente porzione di codice del nostro progetto:

public String getCommands() {

return game.getCommands().stream() .map(Command::getName).reduce("", (acc, command) -> acc +

command + "\n");

}

Abbiamo utilizzato una lambda expression nell'operazione `map` per trasformare ciascun oggetto `Command` nella sua proprietà `name`. Inoltre, abbiamo applicato un'altra lambda expression nell'operazione `reduce` per concatenare i nomi dei comandi insieme, separati da un carattere di nuova linea.L'utilizzo delle lambda expressions ci ha permesso di definire logiche di trasformazione e riduzione in modo conciso e leggibile, evitando la necessità di scrivere metodi aggiuntivi o cicli espliciti. Ciò ha contribuito a mantenere il codice pulito e focalizzato sulla sua funzionalità principale.Le lambda expressions in questo contesto hanno dimostrato la loro utilità nell'elaborazione dei dati e nell'applicazione di operazioni su collezioni, semplificando la scrittura del codice e migliorando la leggibilità complessiva.

Thread: Abbiamo sfruttato i concetti di multithreading per implementare un timer di gioco accurato e reattivo. Abbiamo utilizzato i thread per creare un timer che tenesse traccia del tempo trascorso durante il gioco, e che possa variare nel caso in cui si verifichino determinati eventi, fornendo una componente di sfida e suscitando un senso di urgenza tra i giocatori. In definitiva, l'utilizzo dei thread per gestire il timer di gioco è stato cruciale per la dinamicità e l'emozione dell'esperienza di gioco.

Regex: Abbiamo sfruttato le espressioni regolari (regex) come strumento per validare e filtrare i dati all'interno del gioco. Le espressioni regolari sono state applicate sia all'input dell'utente che all'output delle chiamate REST API, migliorando l'accuratezza e la sicurezza dell'interazione con il gioco.

Per quanto riguarda l'input dell'utente, abbiamo usato espressioni regolari per controllare che gli input rispettino determinati formati.

Nel caso delle chiamate REST API, abbiamo applicato espressioni regolari all'output ricevuto dalle API per filtrare e isolare le informazioni rilevanti. Questo processo di filtraggio ha consentito di estrarre solo i dati necessari dalla risposta JSON delle API, semplificando l'elaborazione e migliorando le prestazioni del gioco. L'implementazione delle espressioni regolari ha richiesto una comprensione dettagliata delle sintassi e delle funzioni delle espressioni stesse. Abbiamo garantito che le espressioni regolari fossero ottimizzate per mantenere un equilibrio tra la complessità e l'efficienza. Complessivamente, l'utilizzo delle espressioni regolari ha aggiunto un livello di controllo e precisione al nostro progetto. Le espressioni regolari hanno migliorato l'esperienza del giocatore, contribuendo a prevenire errori di input e garantendo che solo i dati pertinenti fossero utilizzati nell'interazione con le chiamate REST API.

# ATTENZIONE: Spoiler Avanti!

Sei pronto a scoprire la soluzione? Bene, ecco i comandi da eseguire:

"Nord"

"Prendi lente

"Ovest"

"Prendi ossa" -

"Premi pulsante con ossa"

"Nord"

"Prendi chiave"

"Sud"

"Est"

"Nord"

"Apri portone con chiave"

"Nord"

"Premi meccanismo con lente"

"Est"

"Prendi teschio"

"Ovest"

"Sud"

"Sud"

"Sud"