**Documentazione**

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

Obiettivo del Gioco

## TRAMA

L’avventura testuale sviluppata trae ispirazione dalla prima stagione della serie TV *Prison Break*, una delle produzioni televisive più avvincenti e cariche di tensione degli anni 2000. La serie segue la storia di Michael Scofield, un brillante ingegnere che mette in atto un piano estremamente complesso e meticoloso per far evadere il fratello Lincoln Burrows dal carcere di massima sicurezza di Fox River. Lincoln è stato condannato ingiustamente alla pena di morte per un crimine che non ha commesso, vittima di un complotto orchestrato da potenti forze occulte. Convinto della sua innocenza, Michael decide di sacrificare la propria libertà per salvarlo, facendosi arrestare e incarcerare intenzionalmente all’interno della stessa prigione.

L’avventura testuale offre al giocatore la possibilità di vivere in prima persona le dinamiche della fuga e l'interazione con il mondo carcerario. Nei panni di Michael, il giocatore dovrà esplorare l’ambiente della prigione e aggirare le rigide regole imposte dalle guardie e dall’amministrazione carceraria. Ogni scelta avrà un impatto diretto sullo sviluppo della storia, rendendo fondamentale l’uso della logica, dell’ingegno e della strategia per superare gli ostacoli.

Uno degli aspetti chiave dell’avventura è la gestione del tempo e delle risorse: proprio come nella serie TV, Michael ha un piano dettagliato, basato sui tatuaggi impressi sul suo corpo, che nascondono la mappa del penitenziario e altri indizi essenziali per la fuga. Tuttavia, imprevisti e minacce metteranno continuamente alla prova il giocatore, che dovrà adattarsi e trovare soluzioni alternative per portare avanti il piano senza destare sospetti.

L’obiettivo finale è riuscire a evadere prima che il tempo scada, bilanciando rischio e opportunità in un contesto ostile dove ogni minimo errore può compromettere l’intero piano.

## REGOLE

## Il gioco è un’avventura testuale, un genere in cui il giocatore interagisce con l’ambiente di gioco tramite comandi scritti in linguaggio naturale. Il sistema risponde descrivendo luoghi, oggetti, personaggi ed eventi, guidando così il giocatore attraverso la narrazione.

## Durante la partita, il giocatore può esplorare diverse aree della prigione, raccogliere e utilizzare oggetti, oltre a risolvere enigmi per avanzare nella storia. Sebbene la trama segua uno sviluppo lineare, esistono diverse strategie per affrontare gli ostacoli e superare le sfide proposte.

## L’obiettivo finale è riuscire a evadere prima che il tempo scada; in caso contrario, la fuga fallisce e il gioco termina.

## 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.

## LISTA DEI COMANDI

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

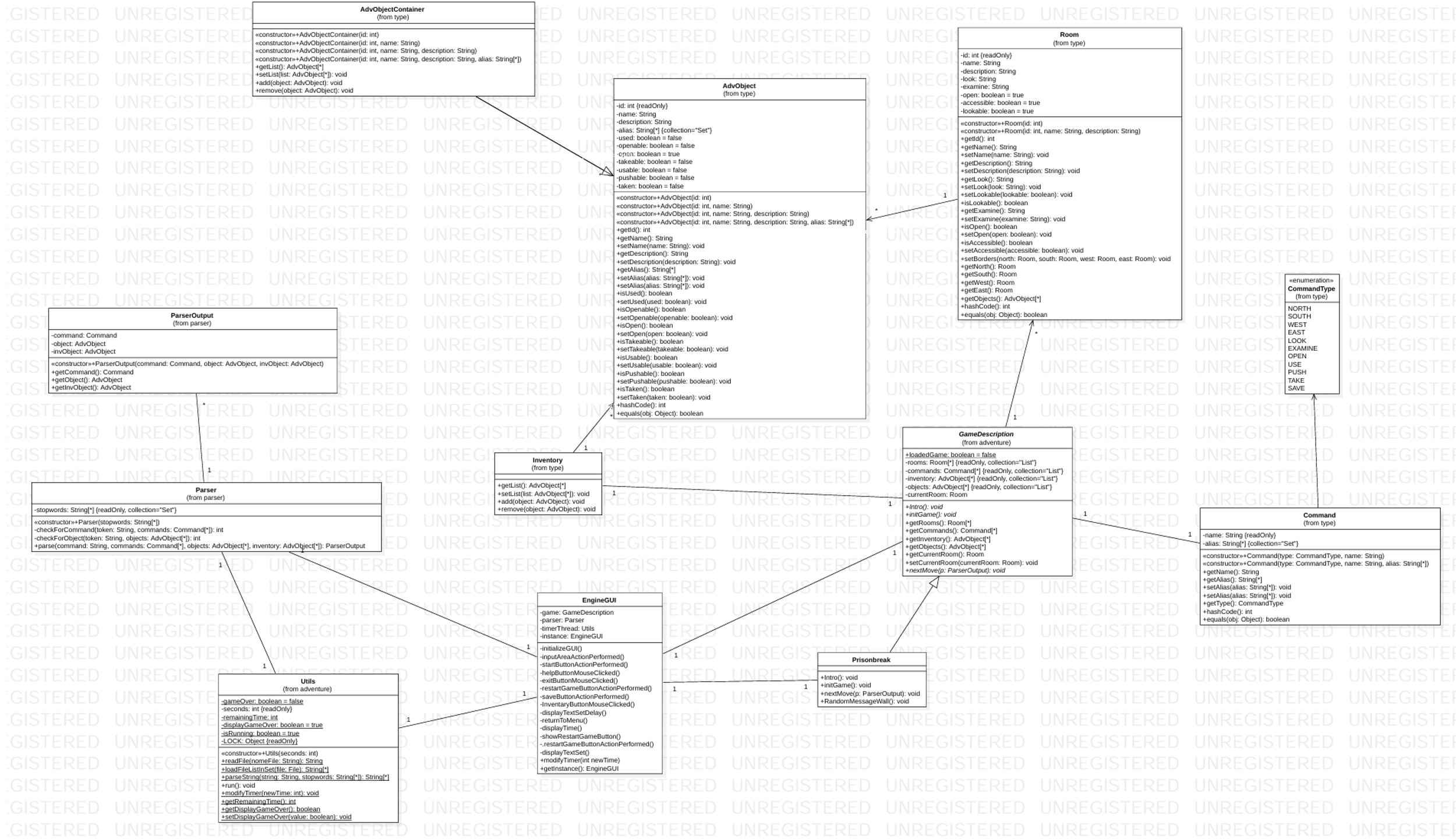
I comandi sono i seguenti:

Immagine che contiene testo, elettronica, schermata, software

Descrizione generata automaticamente

## DIAGRAMMA DELLE CLASSI

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



## Specifica Algebrica

## **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**

**File**: sono stati utilizzati per caricare diversi testi, tra cui l'introduzione, l'elenco dei comandi e i messaggi in caso di vittoria o Game Over. Questa scelta è stata fatta per mantenere il codice del gioco ordinato, evitando di avere più linee di codice che contengono direttamente il testo. Inoltre, è stato impiegato un file di stopwords, che include tutte le parole da ignorare dal parser.

Per rendere l'attività più coinvolgente, sono stati utilizzati anche file audio, aggiungendo un elemento sonoro che arricchisce l'esperienza di gioco.

**JDBC**: è stato adottato per garantire il salvataggio sicuro dello stato di una partita in corso, includendo tutti i progressi raggiunti fino a quel momento. È possibile archiviare in modo efficiente i dati di gioco, permettendo al giocatore di riprendere l'esperienza da dove l'aveva interrotta. Il caricamento della partita salvata può essere effettuato comodamente dal menu.

**Lambda Expression**: è stata utilizzata per gestire la stampa ritardata del testo, un'operazione che deve essere eseguita più volte con un intervallo di tempo specifico. Quando il timer raggiunge l'intervallo stabilito, la lambda expression viene attivata. Essa verifica se l'indice corrente è inferiore alla lunghezza del testo; se sì, estrae la lettera successiva, la stampa e incrementa l'indice di 1. Questo processo continua finché l'indice non raggiunge la fine del testo, momento in cui il timer viene fermato.

**Thread**: è stato utilizzato per implementare un timer che gestisce un countdown, stabilendo il tempo massimo di durata di una partita. Se il gioco termina prima che il tempo scada, il timer si ferma. In caso di salvataggio, il timer continua a contare, riprendendo dai minuti rimanenti. Durante il corso della partita, il timer può essere modificato in base alle scelte del giocatore: determinate azioni o eventi possono influire sul tempo disponibile, rendendo la gestione del countdown dinamica e adattiva all'evolversi del gioco.

**Swing**: è stato utilizzato per creare un menu iniziale che permette di avviare una nuova partita, caricare una partita salvata o uscire dal gioco. Inoltre, è stata sviluppata una GUI che simula una console, comprendente un display che mostra tutto il contenuto letto dai file e le risposte del gioco dopo l'input. L'interfaccia include un'area di input dove puoi digitare i comandi.

Nella GUI, è presente un menu con diverse opzioni: premendo il tasto File, puoi salvare la partita in corso, mentre il tasto Comandi ti fornisce una guida su come interagire con il gioco. Il tasto Inventario ti permette di consultare gli oggetti raccolti, e infine, il tasto Esci ti consente di terminare la partita e uscire dal gioco. Inoltre, nella stessa interfaccia, vengono visualizzati il tempo rimanente e la stanza in cui ti trovi.

**SOLUZIONE GIOCO**

**>prendi chiodo**

**>nord**

**>nord**

**>usa chiodo (opzionale se vuoi guadagnare tempo)**

**>prendi chiave**

**>apri porta**

**>ovest**

**>premi bottone**

**>sud**

**>prendi cacciavite**

**>usa cacciavite**

**>sud**

**>prendi bastone(opzionale se vuoi guadagnare tempo)**

**>usa bastone (opzionale se vuoi guadagnare tempo)**

**>prendi badge**

**>usa badge**

**>ovest**

**>prendi corda**

**>usa corda**

**>ovest**