**Escape Room: Alcatraz**

Versione documento 1.0

Del 28/03/2020

Autore Rodolfo Pio Sassone

Matricola 719017

**Indice**

1. **Gioco………………………………………………………………………………..3**
   1. **Trama**
   2. **Regole**
   3. **Soluzione**
2. **Progettazione…………………………………………………………………..6**
   1. **Idea progettuale**
   2. **Sviluppo**
   3. **Diagramma delle classi**
3. **Specifica Algebrica………………………………………………………….10**
4. **Gioco**
   1. **Trama**

Tu e i tuoi tre amici siete stati rinchiusi nella prigione di massima sicurezza di Alcatraz nella baia di San Francisco, bisogna scappare prima che il vostro arresto sia confermato e veniate inviati alle celle nei sotterranei… o forse è solo un escape room a Padova e tu e i tuoi amici siete ubriachi. Ad ogni modo scappa!!

* 1. **Regole**

Niente che sia sopra la tua testa e niente che necessiti di essere forzato serve ad uscire dalla stanza, quindi non spaccare nulla. Hai 60 minuti ed un inventario comune a tutti i prigionieri per trovare le chiavi ed aprire la porta con 4 serrature che ti permetterà di scappare e vincere il gioco. Aprendo porte o inserendo chiavi all’interno della porta principale incrementerai il tuo score.

* 1. **Soluzione**

**Aprire cella 1 (20%):**

Da atrio o da cella 1

Digitare: sblocca porta est

Digitare: 2618

**Aprire cella 2 (5%):**

Da cella 2 o da atrio

Digitare svita sbarre

**Aprire cella 3 (5%):**

Da atrio

Digitare svuota giubbotto

Digitare prendi chiavi

Digitare apri porta est

**Sbloccare chiave 1 (18%):**

Da atrio

Digitare sblocca chiave1

Digitare 378

Digitare prendi chiave1

Digitare sblocca porta nord

**Sbloccare chiave 2 (24%):**

Da atrio

Digitare sblocca scrigno

Digitare apri scrigno

Digitare prendi chiave2

Digitare sblocca porta nord

**Sbloccare chiave 3 (2%):**

Da atrio

Digitare svuota giubbotto

Digitare prendi taglierino

Digitare infila taglierino cassetta

Digitare apri busta

Digitare prendi chiave3

Digitare sblocca porta nord

**Sbloccare chiave4 (26%):**

Da atrio

Digitare sblocca antaDestra

Digitare 303

Digitare apri antaDestra

Digitare prendi calamita

Digitare est

Digitare sfila laccio scarpone

Digitare ovest

Digitare lega laccio calamita

Digitare lancia calamita tubo

Digitare tira laccio

Digitare prendi chiave4

Digitare sblocca porta nord

**2.Progettazione**

1. **Idea Progettuale**

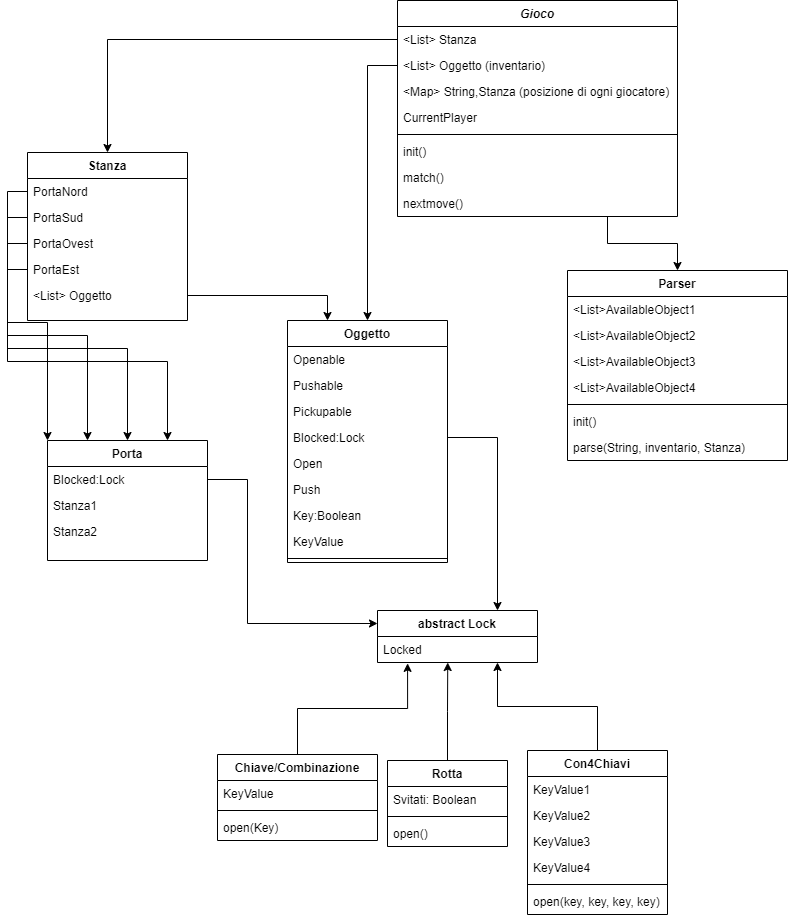
L’idea iniziale è quella di realizzare un’avventura testuale classica, sulla base di un’escape room svolta nella realtà, prendendo come spunto l’architettura del progetto “adventure”.

Figura Sketch Iniziale

1. **Sviluppo**

Durante lo sviluppo lo sketch iniziale è stato più volte modificato fino all’attuale schema delle classi. Dopo la realizzazione dell’avventura testuale è stato aggiunto un server che gestisce il database h2 per tenere traccia dei punteggi dei giocatori così che si possano sfidare tra loro (anche se per adesso il server viene eseguito in locale, il database risiede in locale e quindi è utile solo a salvare i propri progressi o sfidare i punteggi di prova), ed infine un launcher realizzato con l’utilizzo della libreria swing di java.

**Funzionalità**

Il programma offre all’utente la possibilità di giocare una partita dall’inizio, di salvare un punteggio raggiunto e di visualizzare i punteggi salvati anche di altri utenti. È stato deciso di non offrire la funzionalità di messa in pausa poiché l’utente avrebbe potuto usarla come metodo per allungare il tempo a disposizione; è stato deciso, inoltre, di non offrire la funzionalità di salvataggio della partita poiché l’utente avrebbe potuto usarla per mettere in pausa il gioco. Agli utenti Linux il programma offre la possibilità di scegliere un terminale tra gnome-terminal, xterm, e konsole per giocare. In caso non si possegga di nessuno di essi bisogna installarne uno per poter giocare.

**Architettura**

Quando il gioco viene avviato compare a schermo una finestra che permette all’utente di scegliere se giocare, visualizzare i punteggi degli altri giocatori oppure leggere la guida. Se si seleziona la voce “gioca” si aprirà un terminale e verrà avviato il gioco (in caso il sistema operativo sia Linux viene lasciata all’utente la scelta di un terminale tra gnome-terminal, xterm o konsole). Nel caso invece si clicca su “Risultati” verrà avviata una connessione al DB per mostrare in una dialog una tabella formattata in html dei punteggi degli altri giocatori. Infine, se si clicca su “Aiuto” verrà visualizzata una dialog con un file html in cui c’è una guida completa del gioco.

Il Gioco vero e proprio carica un’istanza della classe Game da file, successivamente prende in input una stringa che viene passata al parser. Il parser restituisce un oggetto di tipo ParserOutput il quale viene passato al metodo nextMove() di Game che si occupa di cambiare lo stato dell’oggetto Game.

Attualmente il progetto è composto dai seguenti package:

* escaperoom: contiene il Main e la logica di gioco;
* type: contiene tutti i tipi utilizzati all’interno del gioco;
* parser: contiene tutte le classi utili al parser;
* dev: contiene la main class che si occupa di scrivere un’oggetto di tipo Game in un file;
* utils: contiene alcuni metodi utili;
* server: contiene la classe server e il protocollo con cui client e server comunicano;
* launcher: contiene il form della libreria swing che viene mostrato all’avvio del gioco.

Il server è stato realizzato per servire più client contemporaneamente. Effettua infatti “l’handshake” e crea un thread che si occupa di offrire il servizio al client (ovvero leggere o scrivere sul db). Ovviamente essendo attualmente eseguito in locale sulla stessa macchina del client non c’è nessun caso d’uso reale per cui venga sfruttata questa capacità del server.

1. **Diagramma delle classi**

Immagine che contiene testo

Descrizione generata automaticamenteQui è riportato il diagramma delle classi con le classi fondamentali.

1. **Specifica Algebrica**

**Lista**

La lista è una struttura dati lineare, a dimensione variabile, omogenea.

**Specifica Sintattica**

**Tipi**: posizione, tipoelem, boolean, lista

**Operatori**:

creaLista() -> lista

scriviLista(tipoelem, posizione, lista) -> lista

leggiLista(posizione, lista) -> tipoelem

insLista(tipoelem, posizione, lista) -> lista

cancLista(posizione, lista) -> lista

listaVuota(lista) -> boolean

Siano *t* di tipo tipoelem, *p* di tipo posizione e *l* una lista.

|  |  |  |
| --- | --- | --- |
| **Osservazioni** | **Costruttori di l’** | |
| creaLista() | insLista(t, p, l) |
| listaVuota(l’) | true | false |
| leggiLista(p’, l’) | error | **If**( p = p’ ) **then** t  **else** **if**(p’ > p) **then** leggiLista(p’- 1, l)  **else** leggiLista(p’, l) |
| scriviLista(t’, p’, l’) | error | **If**(p = p’) **then** insLista(t’, p, cancLista(p,l’))  **else** **If**(p’ > p) **then** scriviLista(t’, p’-1, l)  **else** scriviLista(t’, p’, l) |
| cancLista(p’, l’) | error | **If**(p = p’) **then** l  **else if** (p’ > p) **then** cancLista(p’-1, l)  **else** cancLista(p’, l) |

**Specifica Semantica**

l: lista; p, p’: posizione; t, t’: tipoelem; con p != p’ e t != t’

Il tipo posizione è un enumerativo e quindi riconducibile ad un sottoinsieme di N; per questo ci avvaliamo delle operazioni aritmetiche di somma e differenza oltre ai confronti qualora ne necessitiamo.

Il tipo tipoelem rappresenta un tipo qualsiasi.

listaVuota(creaLista()) = true

listaVuota(insLista(t, p, l)) = false

leggiLista(p, insLista(t, p, l)) = t

leggiLista(p’, insLista(t, p, l) = **If**(p’ > p) **then** leggiLista(p’- 1, l) **else** **if**(p’ < p) **then** leggiLista(p’, l)

scriviLista(t’, p, insLista(t, p, l)) = insLista(t’, p, cancLista(p, insLista(t, p, l)))

scriviLista(t’,p’, insLista(t, p, l) = **If**(p’ > p) **then** scriviLista(t’,p’-1,l) **else** **if**(p’ < p) **then** scriviLista(t’, p’, l)

cancLista(p, insLista(t, p, l)) = l

cancLista(p’, insLista(t, p, l)) = **If** (p’ > p) **then** cancLista(p’-1, l) **else if**(p’ < p) **then** cancLista(p’, l)

scriviLista(t, p, cancLista(p, l)) = scriviLista(t, p+1, l)

scriviLista(t, p’, cancLista(p, l)) = **If**(p’ > p) **then** scriviLista(t, p’+1, l) **else** **if**(p’<p) **then** scriviLista(t, p’, l)

leggiLista(p, cancLista(p, l)) = leggiLista(p+1, l)

leggiLista(p’, cancLista(p, l)) = **If**(p’ > p) **then** leggiLista(p’+1, l) **else** **if**(p’ < p) **then** leggiLista(p’, l)

**Restrizioni**

Indichiamo con 1 la prima posizione e con *n* la posizione dell’ultimo elemento nella lista, allora *1 <= p <= n* e *1 <= p’ <= n* in caso di scriviLista() e cancLista() e *1 <= p <= (n+1)* e *1 <= p’ <= (n+1)* in caso di insLista(). Nei casi in cui questo non avviene si ha un errore.

leggiLista(p, creaLista()) = error

scriviLista(t, p, creaLista()) = error

cancLista(p, creaLista()) = error