**ESCAPE ROOM**

**Davide Porcelluzzi – Domenico Signorile**

1. INTRODUZIONE

Progetto realizzato per il completamento del corso di Metodi Avanzati di Programmazione, è stato realizzato un Engine per avventure grafiche/testuali attraverso l’utilizzo del linguaggio Java, delle Swing e del tool Maven.

L’avventura testuale “Escape Room” trae ispirazione dalla saga horror “SAW L’Enigmista”. La saga narra le vicende di un maniaco omicida che mette alla prova determinate persone per vedere se sono degne di continuare a vivere una vita che fino a quel momento hanno disprezzato.

Ecco “Escape Room” è una rivisitazione ovviamente meno cinica e meno spietata, vota al sottolineare l’importanza delle nostre scelte e le conseguenze che queste comportano, soprattutto se fatte per un proprio tornaconto e a svantaggio di altri.

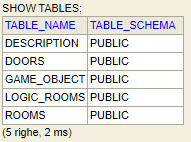
2. REQUISITI PROGETTUALI

Il sistema progettato deve soddisfare le seguenti specifiche:

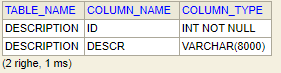
* Definire i componenti di un'avventura ed essere in grado di eseguirla correttamente
* Interpretare in maniera corretta i comandi forniti dall'utente
* Essere in grado di eseguire diverse avventure testuali senza dover necessariamente modificare la stragrande maggioranza del codice

3. DESCRIZIONE DATABASE (H2)

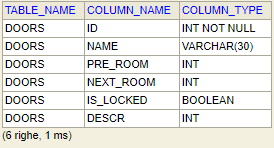
Il DataBase si basa sulle seguenti tabelle.



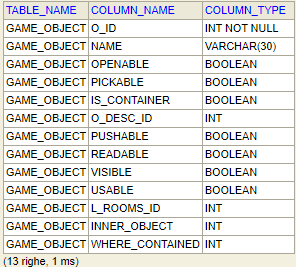
* Struttura delle singole tabelle:
* **DESCRIPTION**:



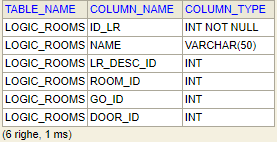
* **DOORS**



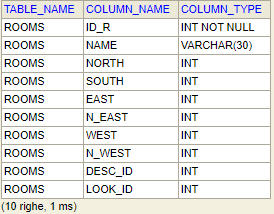
* **GAME\_OBJECT**



* **LOGIC\_ROOMS**



* **ROOMS**



4. CARATTERISTICHE TECNICHE DEL PROGETTO

L’avventura si presenta come gioco Escape Room in modalità avventura grafica/testuale, grazie anche all’utilizzo delle SWING. Il sistema sin interfaccia con un DB H2 contente:

* Descrizioni testuali di gioco
* Stanze di gioco (Rooms composte da):
  + Stanze logiche (Logic Rooms composte da):
    - Porte
    - Oggetti

In questo modo è possibile focalizzarsi sulla progettazione dell'avventura senza badare troppo alla modifica del codice sorgente.

5. SPECIFICHE ALGEBRICHE

Logic\_Rooms

Specifica sintattica

Tipi: Logic\_Room, short, String, String, List, List, Door, Object

Operatori: Logic\_Room()-> Logic\_Room

setIid(short) -> short

getId() -> short

setName(String) -> String

getName() -> String

setDescription(String) -> String

getDescription() -> String

addDoor(Door) -> List

getDoors() -> List

addObject(Object) -> List

getObjects() -> List

Siano:

l = lista di Logic\_Rooms

l2 = lista di GameObject (oggetti nella stanza)

l3 = lista di Door (porte nella stanza)

id = identificativo di Room (stanza corrente)

o\_id = identificativo di Object(Primo oggetto incontrato entrando nella Logic Room)

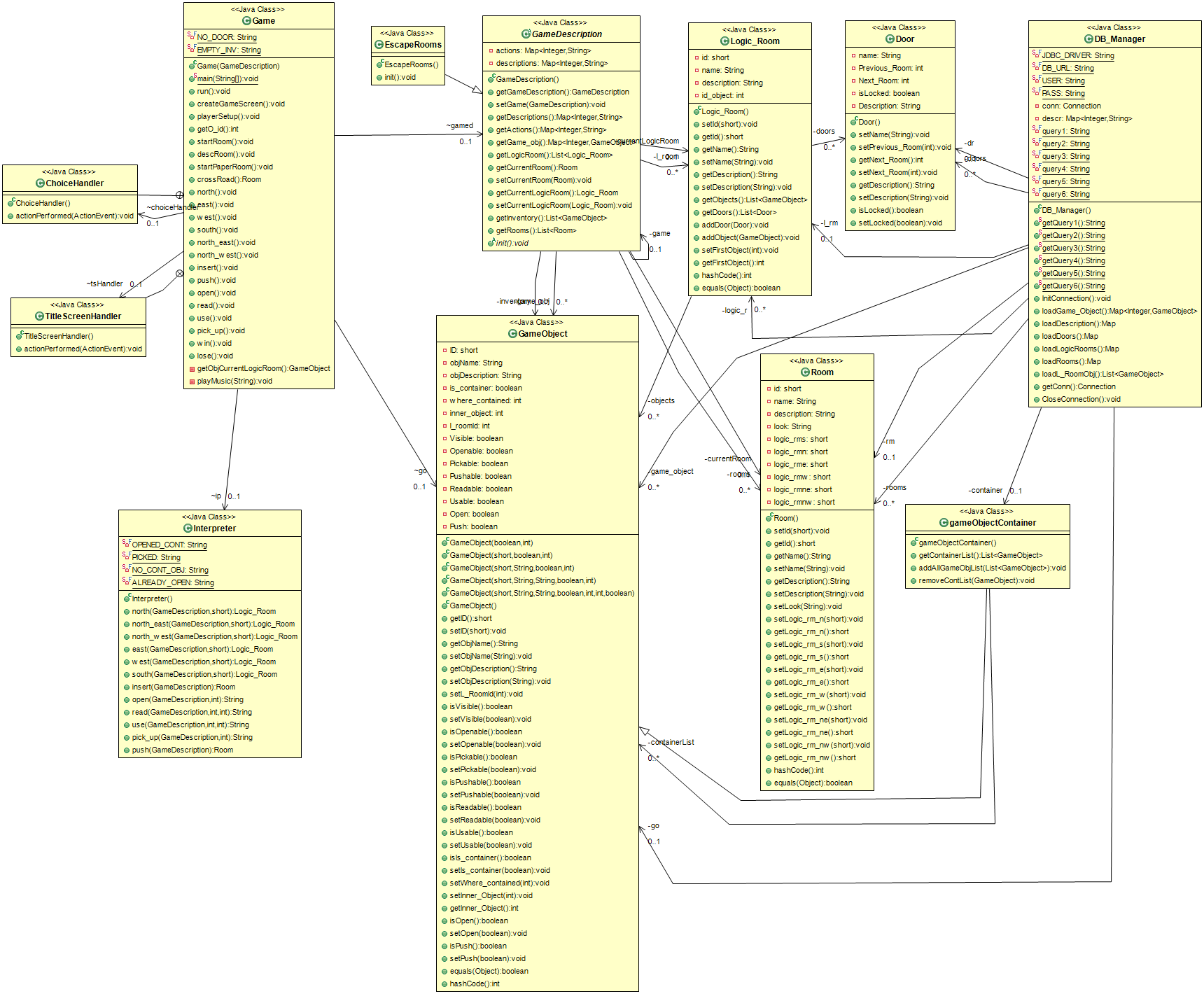
|  |  |
| --- | --- |
| **Osservazioni** | **Costruttori di Logic\_Rooms** |
|  | Logic\_Room() |
| setId(l,id) | Logic\_Room(l) |
| getId(id1) | if l==null then error else l |
| setName(l,name) | Logic\_Room(l) |
| getName(name2) | if l==null then error else l |
| setDescription(l,descr) | Logic\_Room(l) |
| getDescription(l,descr1) | if l==null then error else l |
| getObjects(l,l2) | if l==null then error else  if l2==null then error else l,l2 |
| addObject(l,l2,obj) | if l==null then error else if l2==null then error  else Logic\_Room(l,l2+obj) |
| getDoors(l,l3) | if l==null then error else if l3==null then error  else l,l2 |
| addDoor(l,l3,door) | if l==null then error else if l3==null then error  else Logic\_Room(l,l3+door) |
| setFirstObject(o\_id) | Logic\_Room(l,l2+firstobject) |
| getFirstObject() | If l2 == null then error else l2(o\_id) |

**Specifica di restrizione**

Restriction

addDoor(Logic\_Room()) Error

6. OO CLASS DIAGRAM



7. TEORIA APPLICATA:

* JDBC

E’ l'acronimo di Java Data Base Connectivity. Standard che permette l'indipendenza delle piattaforme anche per le applicazioni che operano su basi di dati. Offre un driver manager che permette l'interrogazione delle basi di dati tramite le query. Nell'ambito del progetto, il database è stato creato ed utilizzato al fine di poter permettere l'esecuzione potenziale di più avventure sfruttando il medesimo motore di gioco. Per questo motivo al lancio del sistema vengono scaricati tutti i dati dal DB mediante una serie di query e una volta scaricati i dati e chiusa la connessione, il gioco prosegue sfruttando i dati scaricati in locale.

* RTTI

Sta per Run-Time Type Identification e permette l'identificazione a runtime. Abbiamo utilizzato l’ RTTI attraverso la parola chiave **instanceof** che ci è stata utile quando data una stanza con oggetti di tipo GameObject e gameObjectContainer, era necessario applicare comportamenti e operazioni diverse sulla base del fatto che l'istanza corrente fosse un oggetto di gioco o un contenitore di oggetti di gioco.

* I/O from file

Abbiamo utilizzato le librerie Java File necessaria per la lettura di file per la visualizzazione delle immagini e per l’audio di sottofondo.

8. STRUTTURA BASE DEL SISTEMA

Il nostro Sistema permettere di poter riutilizzare l’Engine principale cambiando semplicemente DB con una nuova storia potendo aggiungere e/o modificare: Storia, stanze e oggetti.

Ciò permette di rendere più generico possibile il sistema che carica, istanzia e crea tutti gli elementi dell'avventura in maniera automatica dal DB.

Per fare ciò abbiamo rispettato il Riuso (dividendo tutte le componenti del sistema), la Modificabilità (sfruttando il caricamento dell'avventura tramite DB).

9.DESIGN

Il Design dell’avventura testuale: “Escape Room” è composto:

1. Da bottoni che indicano le varie azioni che l’utente può svolgere.

Cambiano testo funzione/azione e quantità in base alla scelta dell’utente.

1. Da una Text Area Dinamica che mostra la descrizione in base alle azioni svolte dell’utente.