

AVVENTURA TESTUALE  
THE GREAT JOURNEY

# STORIA

In un'isola del Pacifico vive Brian Timmins, un giovane ragazzo avventuroso e curioso. Brian è il figlio del governatore dell'isola. Dopo anni di prigione all'interno del palazzo reale i suoi genitori hanno concesso di farlo vivere da solo nella depandance a pochi passi dal palazzo. Da allora il suo obiettivo è diventato quello di esplorare il mondo, cercando (o meglio costruendo) un modo per andar via dall'isola senza che i suoi genitori ficchino il naso dappertutto!

Brian appena rientrato da una mini vacanza si mette subito all'opera per trovare il modo di andarsene via per sempre da quest'isola.

Imparerà a dialogare e raccogliere oggetti di ogni tipo pur di costruire qualcosa che possa permettergli di scoprire cosa c'è al di là del mare.

# METODOLOGIE DI SVILUPPO

# JOURNEY

“Journey” è un programma, interamente scritto in JAVA, che offre la possibilità di giocare una qualsiasi avventura testuale.

Sfrutta un’interfaccia grafica, realizzata con la libreria **SWING**, che permette l’inserimento dei comandi da parte del giocatore e mostra le opportune risposte.

La caratteristica di Journey è che non contiene alcun riferimento ad una avventura testuale bensì ne supporta potenzialmente infinite.

Nella fase di *start* del programma, la classe **Engine** esegue il setup dell'avventura.

# ENGINE

Classe che implementa l'interfaccia *Runnable*, è il motore del programma, poichè gestisce tutte le comunicazioni con le altre entità:

- Comunica con la UI attraverso un *PrintStream*
- Comunica con il *Parser*, gestendo eccezioni (*ParsingException*) e risposte a comandi predefiniti
- Ascolta l'input dell'utente sfruttando uno *Scanner*
- Mantiene in memoria un istanza della classe *Game*

# PARSER

Lo scopo di questa classe è trasformare l'input (*String*) fornito dal giocatore in un *ParserOutput*, scatenando eventuale un'eccezione di tipo *ParsingException*.

Il parser adopera un algoritmo di trasformazione dell'input andando ad effettuare diverse operazioni come: trimming, rimozione delle *stopwords*, analisi delle entità con parole multiple.

Inoltre lavorando su un set predefinito di comandi, effettua controlli puntuali in base al comando (primo token) fornito.

Stampa il messaggio completo in caso di input parzialmente valido.

# STORY HANDLER

## IL CORE DEL PROGRAMMA

La classe elabora l'output del parser sfruttando le "stories" fornite dallo scrittore e comunica con l'engine in due modi:

1. Restituisce il messaggio di risposta
2. Fornisce una domanda per la quale è necessaria una risposta (s/n)

L'elaborazione dell'output consiste nell'aggiornare la struttura del gioco (stanza corrente, oggetti, persone, inventario).

Le meccaniche alla base di una story si basano sul concetto di "eventi".

# STORY

*Story* è un oggetto in grado di gestire una parte della storia del gioco, caratterizzato da:

- *Input*: determina i requisiti per “eseguire” questa storia (comando, stanza, inventario, persona, oggetti)
- *Output*: può o meno fornire una domanda (s/n) e gestire i cambiamenti del gioco (modifiche di oggetti, persone e stanze)
- *Score*: il punteggio totalizzato a valle dell’esecuzione della storia

# GAME

Il gioco può essere visto come un automa a stati finiti non deterministico (NFA) dove la funzione di transizione per passare da uno stato all'altro è rappresentata dalle "story".

# CARATTERISTICHE

Il punto di forza di "Journey" è la possibilità di gestire la storia direttamente da un file JSON senza la necessità di modificare il codice sorgente.

Il codice sorgente, tuttavia, è aperto alle modifiche in quanto è possibile facilmente:

- Supportare nuovi comandi (inserendo la tipologia, eseguendo il parsing ed eventualmente gestendo le modifiche al gameplay)
- Aggiungere nuove direzioni
- Migliorare l'algoritmo di parsing

# CARATTERISTICHE

Oltre alle stories, i tipi supportati dal programma sono:

- stanze (*Room*)
- oggetti (*AdvObject*)
- persone (*Person*)
- comandi (*Command*)
- Eccezioni (*ParsingException*)
- direzioni (*Direction*)

Lo scrittore può popolare tali strutture dati (popolando dei JSON) a meno delle direzioni che sono un subset dei comandi.

# RUNTIME

Il sistema mostra le informazioni presenti nel JSON della classe *Game*: introduzione, titolo, eventuale “help” e descrizione della prima stanza.

L’input inserito dall’utente viene elaborato dal *Parser* ed inviato alla classe *StoryHandler* che si occupa di effettuare le eventuali modifiche al gioco (classe *Game*).

La console mostra la risposta opportuna e l’*Engine* attende il nuovo input dell’utente.

Il sistema si interrompe solo quando il giocatore digita il comando *END* o esegue la “story” finale (completa il gioco).

# NOTE

L'avventura testuale “The Great Journey” è interamente scritta su file in formato JSON.

La piattaforma “Journey”, scritta in JAVA, permette il caricamento di una qualsiasi avventura testuale senza la necessità di modificare il codice.

L'unico requisito per scrivere un'avventura testuale è la conoscenza della notazione JSON e la capacità di descrivere l'andamento della storia attraverso eventi (*Story*).

Inoltre, grazie a questa caratteristica, è in grado di eseguire un'avventura testuale scritta in qualsiasi lingua.

# ANALISI DEI FILE JSON

# JSON - COMANDO

```
[  
  {  
    "name": "END",  
    "alias": ["end", "fine", "abbandona", "quit", "muori"]  
  },  
  {  
    "name": "RESTART",  
    "alias": ["restart"]  
  },  
  {  
    "name": "INVENTORY",  
    "alias": ["i", "inventario", "inv"]  
  },  
  {  
    "name": "SCORE",  
    "alias": ["punti", "punteggio", "mosse"]  
  },  
  {  
    "name": "NORTH",  
    "alias": ["nord", "n"]  
  },  
  {  
    "name": "SOUTH",  
    "alias": ["sud", "s"]  
  },  
  {  
    "name": "EAST",  
    "alias": ["est", "e"]  
  },  
  {  
    "name": "WEST",  
    "alias": ["ovest", "w"]  
  }]
```

Il JSON dei comandi è una lista di oggetti che hanno le seguenti chiavi:

- “name”: indica il nome del comando (indicato nell’enum *Command.Name*)
- “alias”: è una lista di stringhe che identificano quel comando

# JSON - COMANDI PREDEFINITI

```
[{"commands": ["merda", "dannazione", "porco"], "answer": "Modera i termini o ti rinchiedo nelle prigioni del palazzo!"}, {"commands": ["dormi", "sonnecchia"], "answer": "Non hai sonno!"}, {"commands": ["spiacente", "scusa"], "answer": "Oh, non scusarti."}]
```

Il JSON dei comandi predefiniti serve a fornire sempre la stessa risposta:

- “commands”: è una lista di stringhe che identificano il comando
- “answer”: è la risposta che viene fornita quando il giocatore digita uno dei comandi presenti

# JSON - STANZA

```
[  
  {  
    "id": 0,  
    "name": "Dependance privata",  
    "description": "Sei nella dependance, il posto in cui vivi, un  
    arredamento moderno ma con un vecchio frigo che ogni tanto smette di  
    funzionare. Dalla finestra vedi i giardini reali.\nL'uscita è a  
    ovest.\n\nSul letto, ovviamente disfatto, vedi qualcosa.",  
    "west": 1,  
    "wrongDirectionMessage": "L'uscita della dependance è a ovest"  
  },  
  {  
    "id": 1,  
    "name": "Giardini del palazzo",  
    "description": "Sei all'interno dei magnifici giardini, il biglietto  
    da visita del palazzo reale. Intorno a te ci sono maestosi alberi e  
    fontane, farfalle che danzano su piccoli e coloratissimi fiori.\nOggi  
    il palazzo è chiuso in quanto è in corso una riunione top secret. A  
    est c'è l'ingresso della dependance, a nord la lavanderia, mentre a  
    sud vai verso il mare.\n\nC'è una guardia all'ingresso.",  
    "north": 2,  
    "east": 0,  
    "south": 3  
  },  
  {  
    "id": 2,  
    "name": "Lavanderia",  
    "description": "Sei in una grande stanza con grandi vasche per la  
    lavanda. C'è un grande armadietto con diversi strumenti per la  
    pulizia. A sud c'è un'uscita.  
      
    messaggio: Lavanderia  
    messaggio errore: Non puoi uscire da qui",  
    "south": 1  
  }]
```

Il JSON della stanza identifica le caratteristiche delle stanze del gioco:

- “**id**”: univoco all’interno delle stanze
- “**name**”: nome della stanza
- “**description**”: descrizione della stanza
- “**south**”, “**north**”, “**west**”, “**east**”: indicano gli ID delle stanze collegate
- “**wrongDirectionMessage**”: messaggio opzionale che viene mostrato quando il giocatore sbaglia uscita.

# JSON - ECCEZIONE

```
[{  
    "name": "EMPTY_INPUT",  
    "message": "Sei in silenzio stampa?"  
},  
{  
    "name": "UNKNOWN_COMMAND",  
    "message": "Non conosco questo verbo."  
},  
{  
    "name": "LONG_INPUT",  
    "message": "Ho capito la frase solo fino a: %s"  
},  
{  
    "name": "CANT_OPEN",  
    "message": "Non puoi aprire %s"  
},
```

Il JSON delle eccezioni serve a fornire una descrizione dell'input errato/parziale del giocatore:

- “name”: indica il nome del comando (indicato nell’enum *ParsingException.Name*)
- “message”: è la risposta che viene fornita quando l’input determina un’eccezione

# JSON - OGGETTO

```
{  
    "id": 9,  
    "room": 1,  
    "name": "libro",  
    "alias": ["tomo", "giornale", "opera", "romanzo", "tomo", "testo",  
    "manuale"],  
    "description": "Sulla copertina del libro c'è una grossa nave, se  
    solo riuscissi a prenderlo magari potrei imparare a realizzare una  
    barca per fuggire dall'isola.",  
    "customCommandMessages": [  
        {"command": "INVENTORY",  
         "description": "Un libro di carpenteria navale."  
        },  
        {  
            "command": "PICK_UP",  
            "description": "\"Ehi, ma che fai, è il mio libro!\" Sembra che  
            la guardia non voglia proprio disfarsene."  
        },  
        {  
            "command": "READ",  
            "description": "\"Ehi, ma che fai, è il mio libro!\" Sembra che  
            la guardia non voglia proprio disfarsene."  
        }  
    ]  
},
```

Il JSON di un oggetto descrive le sue caratteristiche, il suo stato (gestito da booleani, false di default) e eventuali risposte personalizzate:

- “**Id**”: numero univoco all’interno degli oggetti
- “**room**”: id della stanza in cui si trova (-1 se non è visibile)
- “**name**”: nome dell’oggetto
- “**alias**”: altri nomi dell’oggetto
- “**descrizione**”: descrizione dell’oggetto
- “**customCommandMessages**”: risposte personalizzati a comandi diretti all’oggetto

# JSON - PERSONA

```
{  
  "id": 0,  
  "room": 1,  
  "name": "Guardia del palazzo",  
  "description": "Vestita di tutto punto e con dei baffi enormi, la  
  guardia reale ti incute paura. Porta con sè un voluminoso libro,  
  nessuno ha mai osato prenderlo.",  
  "alias": ["guardia", "guardiano", "poliziotto", "gendarme", "guardia  
  reale", "guardia palazzo"],  
  "customCommandMessages": [  
    {"command": "SPEAK",  
     "description": "\"Ciao Brian, vedo che sei tornato dal tuo safari  
     avventuroso. Oggi il palazzo è chiuso per una importante  
     riunione.\nFinisco il turno tra un paio d'ore, magari ci becchiamo  
     più tardi in spiaggia.\"\\n\\nMi tocca aspettare la fine del turno  
     allora per entrare nel palazzo, potrei approfittarne e andare a  
     salutare il mio amico barista."}  
  ]  
},  
{
```

Il JSON di un persona descrive le sue caratteristiche ed eventuali risposte personalizzate:

- “Id”: numero univoco all’interno delle persone
- “room”: id della stanza in cui si trova (-1 se non è visibile)
- “name”: nome della persona
- “alias”: altri nomi dell’oggetto
- “descrizione”: descrizione della persona
- “customCommandMessages”: risposte personalizzati a comandi diretti alla persona

# JSON - PROTAGONISTA

```
{  
    "id": -1,  
    "name": "Brian Timmins",  
    "description": "Hai sempre lo stesso bell'aspetto.",  
    "alias": ["me stesso", "brian", "brian timmins", "me"],  
    "customCommandMessages": [  
        {  
            "command": "PICK_UP",  
            "description": "Sei sempre il possessore di te stesso."  
        },  
        {  
            "command": "PUSH",  
            "description": "Questo sarebbe meno che cortese."  
        },  
        {  
            "command": "PULL",  
            "description": "Questo sarebbe meno che cortese."  
        },  
        {  
            "command": "GIVE",  
            "description": "Sei sempre il possessore di te stesso."  
        },  
        {  
            "command": "SPEAK",  
            "description": "Stai parlando a te stesso."  
        }  
    ]  
}
```

Il JSON del protagonista descrive le sue caratteristiche ed eventuali risposte personalizzate:

- “**Id**”: numero identificativo
- “**name**”: nome della persona
- “**alias**”: altri nomi dell’oggetto
- “**descrizione**”: descrizione della persona
- “**customCommandMessages**”: risposte personalizzati a comandi diretti alla persona

# JSON - GIOCO

Oltre ai contenuti del gioco indica:

```
{  
    "introduction": "",  
    "title": "The Great Journey",  
    "description": "Un'avventura entusiasmante di Alberto Saltarelli",  
    "helpQuestion": "Vuoi visualizzare le istruzioni? (s/n)",  
    "help": "",  
    "restartQuestion": "Sei sicuro di voler ricominciare?",  
    "endQuestion": "Sei sicuro di voler uscire?",  
    "uselessCombineCommand": "Cosa dovrebbe uscirne fuori?",  
    "endGameMessage": "\t*** Hai vinto ***",  
    "increaseScoreMessage": "[Il tuo punteggio è appena aumentato di %d]",  
    "scoreMessage": "Finora hai totalizzato %d punti su %d possibili, in %d turni.",  
    "inventoryEmpty": "Non stai portando niente.",  
    "inventoryFull": "Stai portando:",  
    "unknownOutput": "Non ha alcuna importanza per lo svolgimento della tua avventura.",  
    "yesAlias": ["si", "s"],  
    "noAlias": ["no", "n"],  
    "maxScore": 16,  
    "initialRoom": 0  
}
```

- “**helpQuestion**”: domanda d’aiuto iniziale
- “**Help**”: risposta al comando **help**
- “**unknownOutput**”: messaggio di errore. Non dovrebbe essere mai mostrato
- “**initialRoom**”: id della prima stanza

# JSON - STORIA

Il JSON che caratterizza l'avanzamento della storia dell'avventura.

```
{  
    "input": {...},  
    "output": {...},  
    "score": int,  
    "isLast": bool,  
    "delete": bool  
}
```

- “input”: indica il requisito per eseguire la storia
- “output”: indica la risposta al giocatore
- “score”: il punteggio effettuato (opzionale)
- “isLast”: indica la fine del gioco (opzionale)
- “delete”: se false, la storia può essere eseguita più volte (opzionale)

# JSON - STORIA (INPUT)

```
{  
    "command": "PICK_UP",  
    "room": int,  
    "inventoryRequirements": [int],  
    "person": int,  
    "objects": [int]  
}
```

Azione che deve compiere il giocatore

- “command”: indica il nome simbolico del comando
- “room”: id della stanza corrente (opzionale)
- “inventoryRequirements”: oggetti necessari nell’inventario (opzionale)
- “person”: id della persona a cui è rivolto il comando (opzionale)
- “objects”: lista di oggetti inerenti al comando (opzionale)

# JSON - STORIA (OUTPUT)

```
{  
  "question": "...",  
  "message": string,  
  "editing": {  
    "objects": [  
      {  
        "id": int,  
        "moveToRoomID": int,  
        "moveToInventory": bool,  
        "description": string,  
        "customCommandMessages": [  
          {  
            "command": "PICK_UP",  
            "description": ""  
          }  
        ]  
      }  
    ],  
    "people": [  
      {  
        "id": int,  
        "moveToRoomID": int,  
        "description": string,  
        "customCommandMessages": [  
          {  
            "command": "PICK_UP",  
            "description": ""  
          }  
        ]  
      }  
    ],  
    "rooms": [  
      {  
        "id": 0,  
        "description": ""  
      }  
    ]  
  }  
}
```

L'output fornisce informazioni circa le modifiche al gioco e la risposta da fornire al giocatore

- “question”: oggetto che prevede una domanda e risposta
- “message”: messaggio da mostrare al giocatore
- “editing”: oggetto responsabile delle modifiche
- “objects”: lista di oggetti da modificare caratterizzati da tutte le proprietà dell'oggetto (canTake, canPull, isPush ecc...)
- “people”: lista di persone da modificare
- “rooms”: lista di stanze per cui necessita la modifica della descrizione

Tutti i campi, tranne “message”, sono opzionali. Inoltre “moveToRoomID” permette di spostare oggetti/persone in altre stanze (se -1 rimuove dal gioco).

# JSON - STORIA (ESEMPI)

“prendi i vestiti”

```
{  
  "input": {  
    "command": "PICK_UP",  
    "objects": [  
      6  
    ],  
    "output": {  
      "message": "Presi.",  
      "editing": {  
        "objects": [  
          {  
            "id": 5,  
            "description": "Il letto è disfatto..."  
          }  
        ],  
        "rooms": [  
          {  
            "id": 0,  
            "description": "Sei nella depandance..."  
          }  
        ]  
      },  
      "score": 1  
    },  
  },  
  "output": {  
    "message": "Tutto pronto per la marcia!"  
  },  
  "inventoryRequirements": [  
    25,  
    29  
  ]  
}
```

“usa attrezzi con palo”

```
{  
  "input": {  
    "command": "USE",  
    "objects": [  
      25,  
      29  
    ],  
    "inventoryRequirements": [  
      29  
    ],  
    "output": {  
      "message": "Clang! Il palo scivola per terra..."  
    },  
    "delete": true  
  },  
  "output": {  
    "message": "Tutto pronto per la marcia!"  
  },  
  "inventoryRequirements": [  
    25,  
    29  
  ]  
}
```

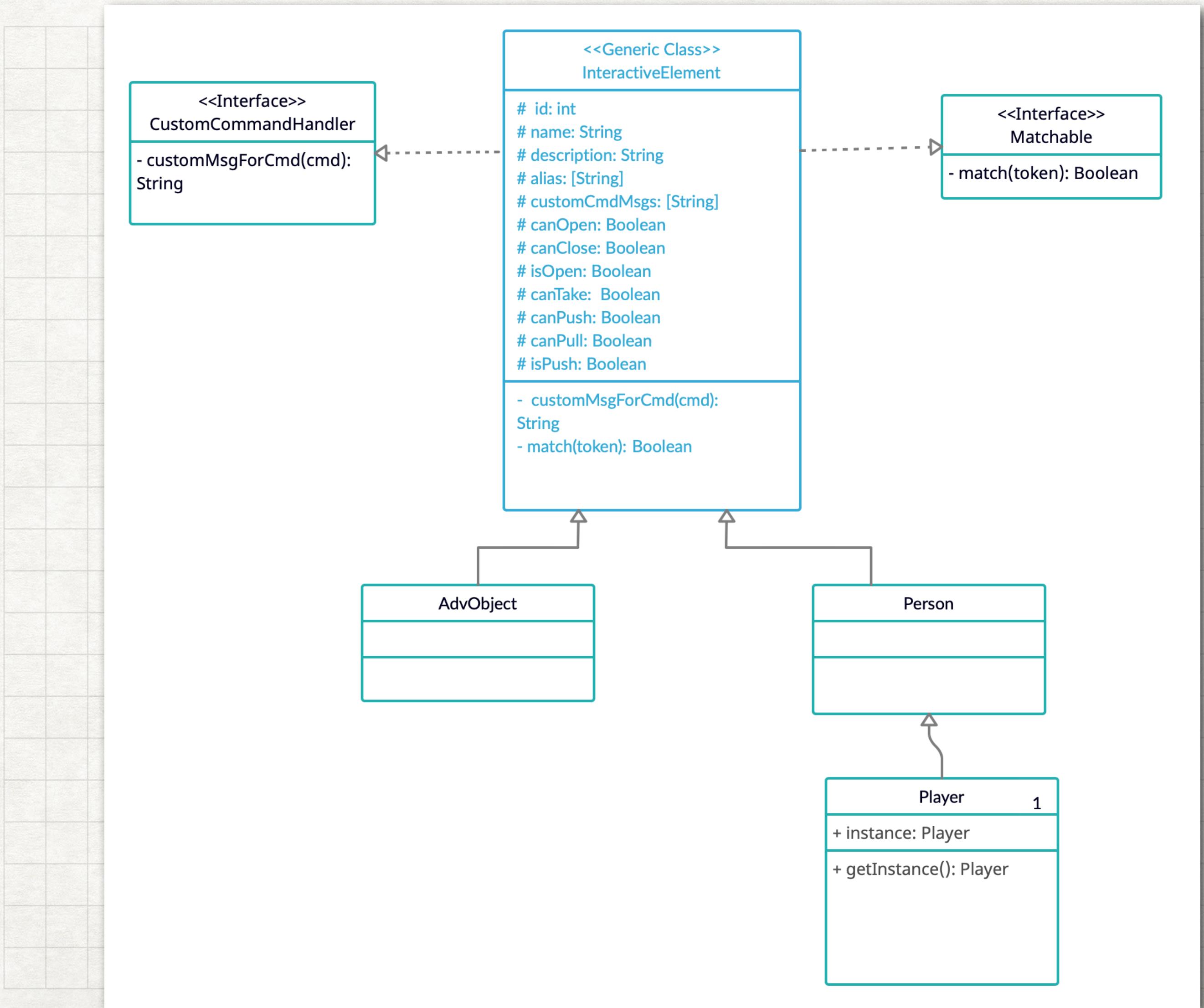
“canta”

```
{  
  "input": {  
    "command": "SING"  
  },  
  "output": {  
    "message": "Risparmia il fiato per cose più importanti."  
  },  
  "inventoryRequirements": [  
    25,  
    29  
  ]  
}
```

MATERIALE  
ACCESSORIO

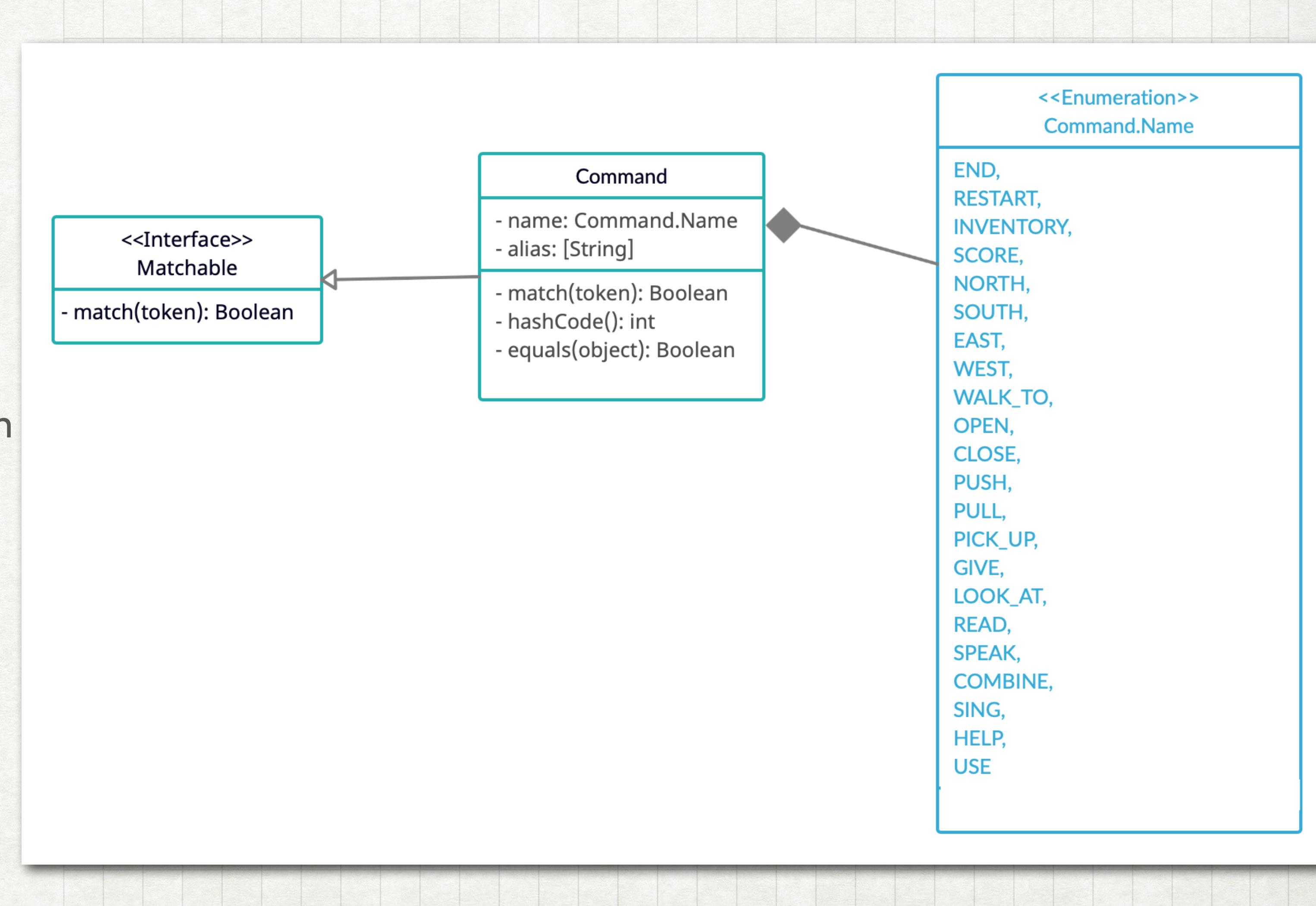
# DIAGRAMMA DELLE CLASSI

- La classe astratta `InteractiveElement` definisce tutte le proprietà di un elemento interattivo (con cui il giocatore può interagire). Sono definite `protected` per essere visibili nei costruttori delle sottoclassi `AdvObject` e `Person`.
- Inoltre la classe astratta `InteractiveElement` implementa due interfacce funzionali: `CustomCommandHandler` (ha un metodo che restituisce un messaggio prendendo in input un comando) e `Matchable` (restituisce true se l'istanza matcha il token in input)
- `AdvObject` e `Person` sono classi concrete ed ereditano per estensione dalla superclasse
- `Player` eredita per estensione dalla superclasse `Person` ed inoltre è un singleton. (Modella il protagonista dell'avventura)



# DIAGRAMMA DELLE CLASSI - 2

- La classe *Command* rappresenta un comando che l'utente può inserire.
- Implementa l'interfaccia funzionale *Matchable* implementando il metodo *Boolean match(String token)*.
- Ha una relazione di composizione con l'enumerazione *Command.Name*.



# SPECIFICA ALGEBRICA - BOOLEAN

## Specifica sintattica

sorts: bool

- true: () -> bool
- false: () -> bool
- not(bool) -> bool
- and(bool, bool) -> bool
- or(bool, bool) -> bool

## Specifica semantica

Declare a: bool, b: bool

- not(true()) = false()
- not(false()) = true()
- and(true(), true()) = true();
- and(a, false()) = false()
- and(false(), a) = false()
- or(a,b) = not(and(not(a),not(b)))

# SVILUPPI FUTURI

1. Sviluppo di un app per MacOS (Xcode, Swift) con interfaccia grafica per generare i file JSON. Chiunque potrà scrivere un'avventura testuale
2. Creazione di un server con DB e 2 API REST (una che scarica i titoli delle avventure ed una, o più, per ottenere i json necessari)
3. Possibilità di salvare e caricare la partita in un DB locale
4. Permettere ad una *Story* di fornire una domanda aperta
5. Aggiungere comandi (rispondi, aspetta, suona, dormi, indossa, togli, rompi, versa, mangia, bevi)
6. Migliorare il parser (inserimento grammatica, parser generico basato sull'input dello scrittore)