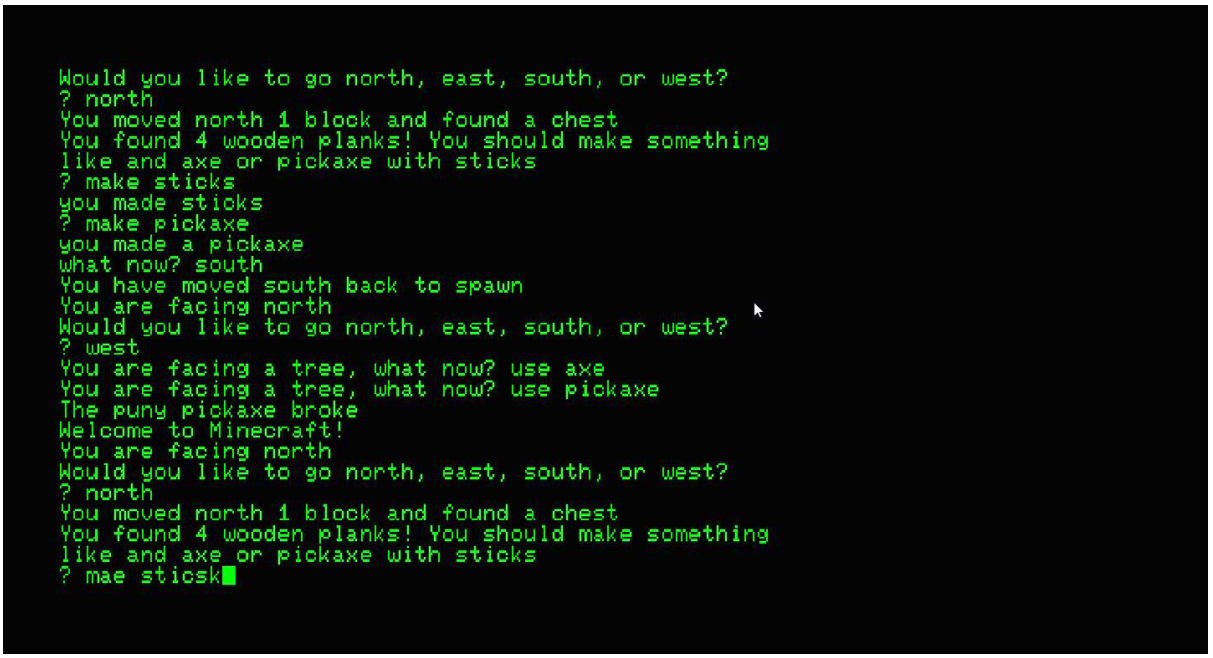
Specifiche del Progetto del Corso di Metodologie di Programmazione 2020/2021 (M-Z)
Laurea Triennale in Informatica - Sapienza Università di Roma
Prof. Roberto Navigli

I COMMENTI SONO BENVENUTI

Java Text Adventure - Un'avventura testuale

Un'avventura testuale è un videogioco che permette a un giocatore di esplorare un mondo virtuale mediante comandi da tastiera:



Gli elementi minimi del gioco da codificare sono: il mondo, un ambiente del mondo (che chiameremo stanza, anche se si tratta di un ambiente aperto), i personaggi e il giocatore.

La stanza

La stanza è l'elemento base del mondo. Ogni stanza ha un nome, una descrizione testuale (e/o grafica, vedi progetto per due studenti) e mantiene informazioni su:

- gli oggetti contenuti nella stanza (un oggetto può trovarsi in una sola stanza in un dato momento);
- i personaggi e/o giocatore presenti nella stanza (ogni giocatore o personaggio può trovarsi in una sola stanza in un dato momento);
- i punti di accesso ad altre stanze (chiamati `link` nel seguito).

Il mondo

Il mondo è un insieme di stanze ed è dotato di un nome e una descrizione testuale (e/o grafica, vedi progetto per due studenti).

Personaggio

Ogni personaggio ha un nome. Un personaggio che dispone di un inventario di oggetti. Il personaggio dispone di una serie di metodi che gli permettono di interagire con la stanza in cui si trova e con gli altri personaggi presenti nella stanza.

Giocatore

Il giocatore, come gli altri personaggi, dispone di un inventario di oggetti, che però è inizialmente vuoto. E' previsto un solo giocatore nel gioco, quindi è sempre possibile ottenere un riferimento al giocatore.

Caricamento del mondo

Il mondo viene caricato con il metodo `Mondo.fromFile(Path)` che restituisce un mondo caricato e istanziato dal file specificato. Il formato del file in input è il seguente:

```
[world:nome del mondo]
description <tab> descrizione testuale del mondo
start <tab> nome stanza 1

[room:nome stanza 1]
description <tab> descrizione testuale
objects <tab> oggetto1,oggetto2,...,oggetton
characters <tab> personaggio1,...,personaggiom
links <tab> N:porta 1,S:stanza 3,O:stanza 2
```

```
[room:nome stanza 2]
description <tab> …
…
links <tab> E:stanza 1

[room:nome stanza 3]
…
ecc.

[links]
porta1 <tab> Porta <tab> nome stanza 1 <tab> nome stanza 2
finestra1 <tab> Finestra <tab> nome stanza 4 <tab> giardino
teletrasporto1 <tab> Teletrasporto <tab> stanza magica <tab> pianeta X

[objects]
sushi nella boccia <tab> Pesce
chiave <tab> Chiave <tab> porta1
chiave 2 <tab> Chiave <tab> porta2
ecc.

[characters]
Harry <tab> Personaggio
Mary <tab> Personaggio
bobby <tab> Cane
ecc.

[player]
Zak <tab> Giocatore
```

dove “ **<tab>** “ rappresenta il carattere di tabulazione (inclusi gli spazi, inseriti qui solo per separare meglio i campi). In particolare, ci sono sei tipi di sezioni:

- la sezione relativa al mondo che contiene il nome subito dopo la parola chiave `world:`, la descrizione nel formato `description` **<tab>** descrizione del mondo e la stanza da cui iniziare il mondo nel formato `start` **<tab>** `nome stanza 1`
- stanza (`room:nome univoco della stanza` tra parentesi quadre) che contiene i seguenti campi:
    - `description` **<tab>** descrizione della stanza
    - `objects` **<tab>** elenco dei nomi degli oggetti separati da virgola
    - `characters` **<tab>** elenco dei nomi dei personaggi separati da virgola
    - `links` **<tab>** elenco separato da virgola di coppie `carattere:nome di un’altra stanza o di un oggetto di tipo collegamento` (ad es. una porta) dove il carattere può essere `N, S, O, W, E` (`O` e `W` hanno il medesimo significato), a indicare la direzione in cui bisogna andare per entrare in quella stanza.

- elenco dei link (porte, finestre, ecc.) che permettono un collegamento tra due stanze nella sezione `[links]`. Le righe in questa sezione contengono quattro campi: nome del link, classe del tipo di link (`Porta`, `Finestra`, ecc.), nomi delle due stanze collegate dal link.
- elenco degli oggetti con la loro associazione alla classe Java di cui sono istanza nella sezione [`objects`] nel formato: `nome oggetto` **`<tab>`** `classe Java` (in alcuni casi è possibile che venga fornito un terzo parametro che l'oggetto con cui può interagire l'oggetto definito).
- elenco dei personaggi [`characters`] nel formato: `nome personaggio` **`<tab>`** `classe Java`.
- sezione `[player]` del giocatore: contiene il nome e la classe del giocatore. Si può dare per scontato un solo giocatore (e dare errore altrimenti).

L'ordine di queste sezioni non è predeterminato (ad esempio, `character` e `objects` non appaiono necessariamente in fondo al file, come mostrato sopra).

Il metodo per il caricamento deve controllare (ed emettere il conseguente errore) se: 1) lo stesso oggetto o personaggio viene inserito in più stanze, 2) se gli oggetti specificati in ogni stanza sono sempre associati a una classe Java corrispondente (altrimenti non potranno essere istanziati); 3) altri eventuali errori che possono rendere impossibile la creazione del mondo (ad es. due stanze od oggetti con lo stesso nome).

L'estensione del file per la parte grafica è lasciata al gruppo di studenti che la implementeranno (vedi sotto).

Motore di interazione testuale

E' il cuore del gioco. Si tratta di un motore che trasforma un comando testuale in una chiamata a un corrispondente metodo del giocatore. Ad esempio, se scrivo:

```
ovest
```

questo si tradurrà nella chiamata al metodo `Personaggio.ovest()`. E' da notare che non necessariamente un comando testuale si tradurrà direttamente in una chiamata a un metodo con lo stesso nome. Ad esempio, potresti andare ad ovest anche con comandi come:

```
O
vai O
vai a ovest
```

(l'importante è che i comandi presenti negli script forniti funzionino). Quest'ultimo comando, ad esempio, potrebbe tradursi in `Personaggio.vai(Direzione.OVEST)`. Quando un comando non viene riconosciuto, il motore solleva un'eccezione. Come appena mostrato, i comandi possono fornire anche parametri. Ad esempio:

```
prendi sushi nella boccia
```

dovrà far corrispondere il parametro dopo `prendi` con tutti i nomi di oggetti, personaggi, stanze ecc. e, ad esempio, invocare il metodo `Personaggio.prendi(oggetto)`. Analogamente:

```
apri porta con chiave1
```

L'eleganza dell'implementazione del motore è oggetto di valutazione (una cascata di if non è ideale). E' possibile ma non necessario introdurre un concetto di stopword che verranno scartate nel caso in cui fallisca il riconoscimento del comando. Ad esempio:

```
apri la porta con il chiave1
```

L'esecuzione di ogni comando tramite il motore di interazione testuale deve restituire un riscontro testuale del successo o meno dell'operazione richiesta.

Avvio del gioco

L'avvio del gioco avviene mediante le seguenti istruzioni:

```
Gioco g = new Gioco();
Mondo m = Mondo.fromFile(fileName);
g.play(m);
```

Il seguente file deve compilare correttamente (si noti che il package è lo stesso da utilizzare per le classi `Gioco` e `Mondo`; gli oggetti da implementare possono invece utilizzare un sottopackage ad hoc):

```
package it.uniroma1.textadv;

public class Test
{
	public static void main(String[] args) throws Exception
	{
		Gioco g = new Gioco();
		Mondo m = Mondo.fromFile("minizak.game");
		g.play(m);
	}
}
```

Giocata fast forward

Il gioco permette di effettuare una giocata in modalità fast forward prendendo in ingresso uno script che contiene la sequenza dei comandi testuali, tramite la chiamata al metodo d'istanza `play` con due parametri, `Mondo` e `Path`. Ad esempio:

```
Path scriptDiGioco = Paths.get("minizak.ff");
g.play(m, scriptDiGioco);
```

`Gioco Minizak`

Il progetto deve funzionare correttamente sui file .game e .ff di minizak:
https://drive.google.com/file/d/18myOcrSx7NC5wNPiLckxWR0jfy__XZvC/view?usp=sharing
(v1.0.5, include anche mappa del gioco, grazie ad Alain Niccolò Manieri!)
(per legacy, nel caso qualcuno avesse già implementato il progetto, sarà considerata accettabile anche la versione precedente:
https://drive.google.com/file/d/177OA8X8RIoNQuIgA-lTrMrDJoPLRhHpf/view?usp=sharing)

Localizzazione dei comandi (3 punti extra)

E' possibile ottenere 3 punti extra implementando la localizzazione dei comandi (prendi, vai a, ecc.). In questo caso, il mondo dispone di un metodo `Gioco.localizza(lingua)` (dove lingua appartiene a un'enumerazione `Language` con valori `EN` e `IT`) che può passare alla lingua richiesta (con conseguente modifica della gestione dei comandi da parte del motore testuale, che dovrà utilizzare una qualche forma di mappatura dalla lingua di default a quella richiesta, se diversa da quella di default).

Creazione di una propria storia (4 punti extra)

In aggiunta a rendere funzionante minizak, è possibile implementare una propria storia (.game) e il corrispondente file fastforward (.ff), con delle classi ad hoc per oggetti specifici del gioco (queste classi dovrebbero comunque essere inserite all'interno dello stesso package utilizzato per gli oggetti di minizak).

Giochi nascosti, stanze bonus, Easter egg (4 punti extra)

E' possibile sviluppare un easter egg, un gioco nascosto o una stanza bonus, modificando il mondo di Zak (o inserendolo all'interno della proprio storia).

Valutazione

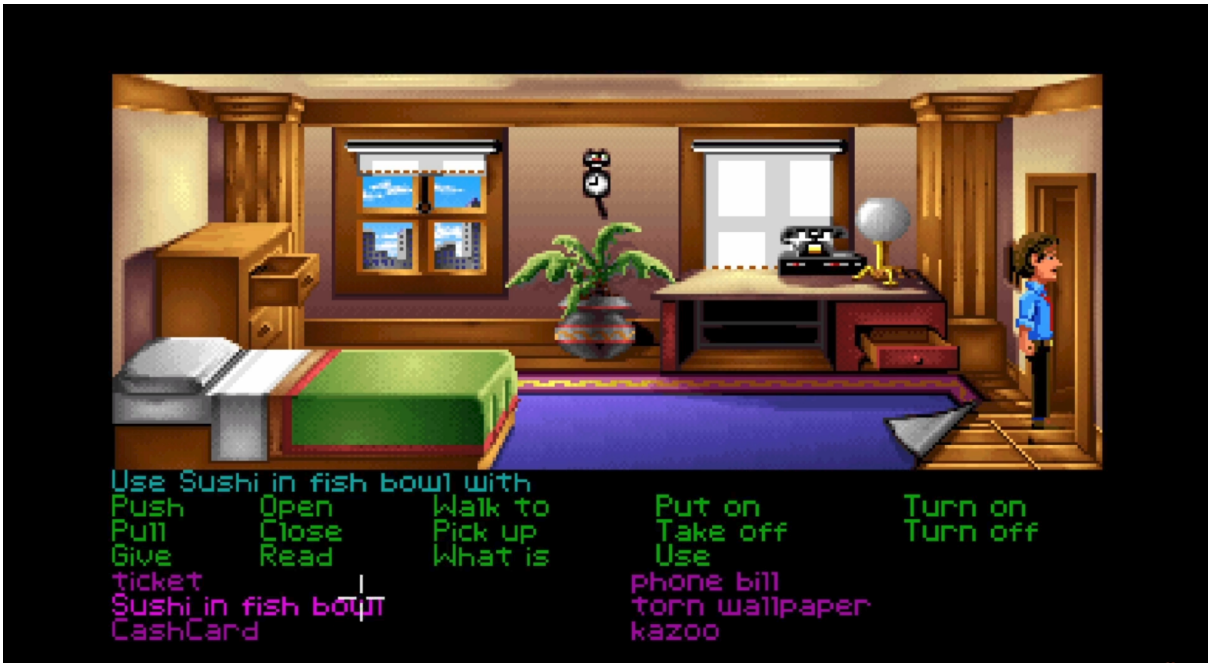
Per ottenere la sufficienza il progetto deve girare correttamente. Verranno valutati i seguenti elementi:

- l'**architettura**, la **modellazione a oggetti** e l'**organizzazione e la pulizia del codice** in termini di **correttezza ed estensibilità**, soprattutto in relazione all'uso appropriato di **naming** (es. **camelCase**), **incapsulamento** e **visibilità di campi e metodi**, **ereditarietà** e **polimorfismo**, il corretto uso di **hashCode** e **equals**.
- la documentazione del codice mediante **javadoc** (su tutte le intestazioni di classe e sui metodi richiesti da queste specifiche, inclusa la documentazione dei parametri), esportata da Eclipse/IntelliJ.

- l'uso di Java 8 (**espressioni lambda** e/o **riferimento a metodi** e **stream**).
- l'utilizzo di **pattern di progettazione**.

Punteggi (da 0 a 34)

- Per ottenere un punteggio minimo (18) il progetto deve poter essere eseguito, **senza percorsi assoluti cablati** (ad esempio, percorsi di file o altre informazioni inserite all'interno delle classi) e deve poter eseguire correttamente la visita del mondo (sulla base di uno script base indicativo che esplora il mondo, senza porte, finestre, ecc.). E' possibile utilizzare questa versione semplificata per verificare l'ottenimento della sufficienza (modulo la validità di quanto sviluppato anche al livello di codice):
- https://drive.google.com/file/d/1t_iX0hPtIMsTGuDl9NPkPO1V0-V5a1YZ/view?usp=sharing aggiornato v1.0.2 (grazie Alain Niccolò Maniero!)
- E' possibile ottenere un voto > 30 solo se di fa un uso appropriato della reflection e dei design pattern.
- **Realizzazione in gruppi di due studenti:** la realizzazione in team è possibile se uno dei due studenti realizza l'interfaccia grafica per il motore testuale sottostante. Si lascia ampia libertà per quanto riguarda 1) la tecnologia della libreria grafica (purché sia una libreria Java, ad esempio JavaFX, LibGDX o altre), 2) la modalità di interazione grafica. Gli unici vincoli importanti sono che 1) l'interfaccia grafica utilizzi il motore testuale realizzato dall'altro studente e il corrispondente feedback, 2) l'interfaccia grafica permetta comunque di interagire con il mondo **ANCHE** mediante la digitazione dei comandi testuali.



Plagio

Indicazioni significative di plagio porteranno gli studenti coinvolti all'annullamento di tutti gli esoneri svolti (anche in precedenza) e all'annullamento di eventuali esami sostenuti in

precedenza, **senza distinguere tra chi ha fatto copiare e chi ha copiato**. Si consiglia **caldamente** **di non condividere** **il codice con altri studenti (neanche per 5 minuti)**.

Invio del progetto

L'invio deve avvenire utilizzando esclusivamente il seguente collegamento. E' obbligatorio l'invio di un file .zip (**non .rar**) contenente l'intero progetto utilizzando il modulo it.uniroma1.textadv. Lo zip deve contenere una sola cartella, chiamata <COGNOME>.<MATRICOLA> (es. Rossi.12345678), al cui interno si troverà il progetto (cartelle src, bin, doc, ecc.). Scompattando il file zip si dovrà ottenere la cartella:

<COGNOME>.<MATRICOLA>

che, al suo interno, conterrà src, bin e doc.

ATTENZIONE: qualunque altro nome di file, formato o modalità di invio porteranno alla mancata correzione del progetto.

Power up!

Tutti gli studenti con voto > 30 saranno premiati con una maglietta di BabelNet!