HOMEWORK 4

Classi Astratte
Riflessione
Eccezioni e I/O

Esercizio 0 (prerequisito)

- Chi non avesse concluso la scrittura dei test, lo faccia in questo homework, prima di fare le modifiche al codice

Esercizio 1

- Scrivere una classe astratta **`AbstractComando`** per eliminare le implementazioni "vuote" dei metodi `setParametro()` dalle classi concrete che estendono l'interfaccia **`Comando`** ma modellano comandi privi di parametri (ad es. **`ComandoGuarda`**)
- Scrivere dei test per la nuova classe e rifattorizzare i test della gerarchia **`Comando`** prodotti con i precedenti homework

Esercizio 2

- Scrivere la classe astratta **`AbstractPersonaggio`** e le classi concrete che la estendono **`Strega`**, **`Mago`**, **`Cane`** come descritto nelle trasparenze POO-15-classi-astratte
- Scrivere le classi **`ComandoSaluta`** e **`ComandoInteragisci`** che modellano i comandi attraverso i quali il giocatore può rispettivamente salutare e interagire con un personaggio come descritto nelle trasparenze POO-15-classi-astratte

Esercizio 3

- Modificare la classe astratta **`AbstractPersonaggio`** introducendo il metodo astratto:

```
public String riceviRegalo(Attrezzo attrezzo, Partita partita)
```

- Scrivere la classe **`ComandoRegala`**, attraverso la quale il giocatore può regalare un attrezzo al personaggio presente nella stanza (NB: in una stanza ci può essere solo un personaggio; il parametro del comando regala è il nome di uno degli attrezzi presenti nella borsa)

Esercizio 3

- Implementare il metodo astratto:
  `public String riceviRegalo(Attrezzo attrezzo)`
  nelle classi **`Cane`**, **`Strega`**, **`Mago`**:
  - Quando un cane riceve un regalo, se questo è il suo cibo preferito lo accetta, e butta a terra un attrezzo, altrimenti non lo accetta e lascia cadere il regalo nella stanza
  - Quando una strega riceve un regalo, se lo tiene e scoppia a ridere
  - Quando un mago riceve un regalo, gli dimezza il peso e lo lascia cadere nella stanza
- La stringa restituita dal metodo rappresenta il messaggio che deve essere restituito dal comando quando eseguito (analogamente al comando interagisci)

Esercizio 4

- Disaccoppiamo i comandi da `System.io`
- Togliamo ai comandi la responsabilità di stampare messaggi
- Anziché stampare a video, i comandi ritornano un messaggio in una stringa: modifichiamo il tipo di ritorno del metodo esegui() da `void` a `String`
- Dopo l'esecuzione di ogni messaggio la classe `DiaDia` (metodo `processaIstruzione()`) stampa il valore ritornato

Esercizio 5

- Disaccoppiamo tutto l'I/O dal gioco. Operiamo nel seguente modo
- Introdurre la seguente interface:

```
public interface InterfacciaUtente {
    public void mostraMessaggio(String messaggio);
    public String prendiIstruzione();
}
```

- Scrivere la classe **`InterfacciaUtenteConsole`** che implementa **`InterfacciaUtente`** usando lo stadard input (**`System.in`**) e lo standard output (**`System.out`**) per interagire con l'utente
- Modificare il codice della classe **`DiaDia`** affinchè deleghi alla classe **`InterfacciaUtenteConsole`** la gestione dell'I/O (la classe **`DiaDia`** avrà una variabile di istanza di tipo **`InterfacciaUtente`** opportunamente inizializzata nel costruttore)

Esercizio 6

- Introdurre ed utilizzare la fabbrica di comandi basata sulla riflessione (vedi trasparenze POO-17-riflessione) per la creazione gli oggetti `Comando`

- Controllando tutto il codice, assicurarsi che l'elenco dei comandi disponibili finalmente *non* sia ridondante

Esercizio 7

- Modificare la classe **`Labirinto`** affinché il labirinto venga caricato da file utilizzando la classe **`CaricatoreLabirinto`** (fornita dal docente)
- Scrivere dei test sulla classe per evidenziare e correggere gli errori di **`CaricatoreLabirinto`**
- *Suggerimenti:*
  - per favorire la la leggibilità dei test ed il loro autocontenimento, e per non vincolare i test all'effettiva presenza di file sul disco, fare uso di fixture specificate tramite stringhe direttamente nei test ed usare **`StringReader`** per farle leggere dal caricatore
  - Scrivere diversi test-case su fixture di complessità crescenti: labirinto monolocale, bilocale, ecc.

Esercizio 8 (facoltativo)

- Modificare la classe **`CaricatoreLabirinto`** affinché sia possibile caricare anche personaggi, stanze chiuse, stanze buie ecc. ecc.
  - *Suggerimento:* serve la riflessione

TERMINI DI CONSEGNA

- La soluzione deve essere inviata al docente entro le 20:00 del 8 giugno 2014 come segue:
  - Svolgere in gruppi di max 2 persone
  - Esportare (con la funzione File->Export di Eclipse) il progetto realizzato nel file **homework4.zip**
  - Inviare il file **homework4.zip** all'indirizzo di posta elettronica poo.roma3@gmail.com
  - Nel corpo del messaggio riportare: eventuali malfunzionamenti noti, ma non risolti, osservazioni, critiche, suggerimenti in merito all'esperienza
  - L'oggetto del messaggio deve iniziare con la stringa **[2014-HOMEWORK4]** seguita dalla matricola/e mittente/i
    - es. [2014-HOMEWORK4] 511276 421314