Programmazione di applicazioni software
CdL in Ingegneria dei Sistemi Informativi
a.a. 2019/2020
Docente: Prof. Andrea Prati

Prova pratica del 30 luglio 2020

Ricordarsi di consegnare il codice sorgente di tutte le classi e della classe main salvando il progetto Code::Blocks in Z:

Simulazione gioco di ruolo

Si scriva un programma per simulare un semplice gioco di ruolo (stile Dungeons&Dragons™). Il gioco di ruolo è costituito da **Personaggio** che partecipano alla partita. Ogni Personaggio è caratterizzato da un nome e da alcune caratteristiche (semplificate, ovviamente, rispetto al gioco originale):

- forza (intero) che rappresenta la forza del Personaggio
- intelligenza (intero) che rappresenta la capacità del Personaggio di fare magie
- carisma (intero) che rappresenta la capacità di guarire
- HP (health point – punti ferita, intero) che rappresenta la salute attuale del Personaggio. *Quando gli HP sono 0 (o negativi) il Personaggio è morto e non può più attaccare*.

Oltre a questo, il Personaggio ha un equipaggiamento sotto forma di una lista di **Oggetti** (si veda dopo). Oltre a setter e getter, la classe Personaggio (e derivate) deve avere anche un metodo per visualizzare lo stato e le caratteristiche del Personaggio, e un metodo attacca a cui passare un altro Personaggio (quello da attaccare) e l'ID (rispetto alla lista di cui sopra) dell'Oggetto da usare per l'attacco.
Esistono tre tipologie di Personaggio: **Mago**, **Guerriero** e **Chierico**. La differenza tra i tre tipi sta solo nel tipo di Oggetti che utilizza (si veda dopo). I costruttori di queste classi devono prevedere di indicare solo il nome, mentre le caratteristiche (forza, intelligenza, carisma e HP) devono essere generati casualmente secondo queste regole:

- Un Mago deve avere:
  - Forza tra 1 e 12
  - Intelligenza tra 12 e 20 (20 è il valore massimo)
  - Carisma tra 1 e 16
  - HP tra 1 e 10
- Un Guerriero deve avere:
  - Forza tra 14 e 20
  - Intelligenza tra 1 e 12
  - Carisma tra 1 e 16
  - HP tra 10 e 30
- Un Chierico deve avere
  - Forza tra 1 e 16
  - Intelligenza tra 1 e 12
  - Carisma tra 12 e 20
  - HP tra 1 e 20

Tempo MASSIMO a disposizione: 180 minuti

Un **Oggetto** è caratterizzato da un nome e una probabilità di successo (successo nel suo utilizzo), codificato come un numero intero da 0 a 100 (ad indicare una percentuale di successo). Esistono tre tipologie di Oggetto: **Arma**, **Pergamena** e **Cura**. Ad un Arma è in aggiunta associato un danno (intero) che rappresenta quanti HP vengono tolti all'avversario se l'attacco ha successo. Ad una Pergamena è associato un booleano per indicare se è una magia di attacco o di cura e un intero da indicare il danno/cura come effetto dell'incantesimo. Ad una Cura è associato un valore di cura che rappresenta il numero di HP recuperato dall'obiettivo (Personaggio) nel caso la cura abbia successo. Ogni tipologia di Oggetto deve avere un metodo usa che ha come parametro un Personaggio ed usa (con le conseguenze – positive o negative – sui suoi HP *se l'utilizzo dell'oggetto ha successo*) l'Oggetto su quel Personaggio.
Ad ogni uso di un Oggetto deve essere generato un numero casuale da 0 a 100 e verificato se questo numero è minore della probabilità di successo dell'Oggetto. Se lo è, l'utilizzo ha successo, altrimenti no. Ad esempio, un Arma "Spada" con probabilità di successo 70 e danno 20, se usata contro il Personaggio P e se il numero casuale fosse 47, avrebbe successo e diminuirebbe di 20 gli HP di P (che se ne ha in quel momento meno di 20, morirebbe).

Nel suo equipaggiamento un Mago ha una lista di Pergamena, un Guerriero una lista di Arma e un Chierico una lista di Cura.

Il main deve inizializzare una serie (non ne servono molti) di Personaggio di ogni tipologia, assegnando ad essi un certo equipaggiamento.
Successivamente il main devi invocare i vari metodi (anche non tutti) di visualizzazione dello stato e di attacco/cura, visualizzando i risultati in output.
Per il raggiungimento della lode, realizzare le operazioni di attacco/cura mediante l'uso dei thread, con opportuni ritardi per verificarne il funzionamento.

Sviluppare le classi che permettono una soluzione basata sulla programmazione Object Oriented al problema.

Tempo MASSIMO a disposizione: 180 minuti