– Esercitazione – 03/05/2022

Questa esercitazione è costituita da due esercizi. Potranno essere consegnati al docente solo gli esercizi che avranno superato i nostri JUnit Test. I test svolti dagli studenti non saranno presi in considerazione per la valutazione. La consegna di almeno un esercizio è condizione necessaria, ma non sufficiente, per passare la prova.

Per la valutazione saranno presi in considerazione aspetti di "buona programmazione", "pulizia del codice" ed efficienza. Ad es.: formattazione corretta del codice, rendere il codice modulare aggiungendo ove necessario altri metodi rispetto a quelli richiesti dall'esercizio, soprattutto se questi rendono il codice più pulito e leggibile, o se evitano duplicazione di codice.

IMPORTANTE: seguire attentamente le specifiche per quanto riguarda i nomi dei metodi e la firma dei metodi, altrimenti i test automatici falliranno rendendo il compito insufficiente.

CONSEGNA ESERCIZI:

Entro il termine ultimo previsto per la consegna dello scritto, gli studenti che intendono consegnare provvedono a caricare i sorgenti **.java** attraverso l'apposita attività "compito", disponibile nella pagina MOODLE del corso al seguente link:

https://e-l.unifi.it/mod/assign/view.php?id=866403

Fino allo scadere del tempo, lo studente potrà apportare modifiche al proprio lavoro. Non saranno accettate consegne effettuate in ritardo, o con modalità diverse da quelle definite dal docente. All'orario stabilito ad inizio compito il docente dichiara finita la prova e chiude la sessione.

Esercizio Java n. 1: Gioco Fantasy

Sviluppare le classi relative ai personaggi di un gioco di ruolo fantasy.

Creare nel pacchetto *fantasy* una classe *Personaggio*, che rappresenta le caratteristiche di base di un personaggio del gioco. Ogni personaggio è caratterizzato dal suo nome e dalla sua forza (intero da 1 a 10) e dall'attuale energia (intero da 0 a 100). Ogni personaggio ha inoltre una borsa che può contenere al massimo 20 cose (istanze della classe *Cosa*).

Un personaggio viene creato specificando nome e forza. L'energia iniziale avrà sempre il valore massimo e la borsa sarà inizialmente vuota (cioè il personaggio quando viene creato non possiede cose). Se la forza del personaggio non viene correttamente specificata, il personaggio non verrà creato e sarà generata un'eccezione di tipo *IllegalArgumentException*.

Non sarà possibile creare istanze della classe *Personaggio*, ma solo istanze delle sue classi derivate descritte sotto.

Ogni istanza della classe *Cosa* (anch'essa nel pacchetto *fantasy*) è caratterizzata da una stringa che ne definisce la tipologia (pietra preziosa, pozione, spada, ecc.) e un numero intero positivo che ne definisce il valore (il tentativo di definire una cosa con valore negativo comporterà la creazione di una cosa con valore pari a 0).

Implementare i seguenti metodi della classe *Personaggio*:

- *inserisciCosaNellaBorsa(Cosa cosa)*: inserisce *cosa* nella borsa del personaggio; se la borsa è piena allora il metodo non inserisce la cosa e semplicemente termina senza fare nulla (cioè NON genera un'eccezione).
- *consumaCosa(String tipologia)*: rimuove dalla borsa del personaggio una occorrenza di una cosa con tipologia *tipologia*; se la borsa contiene tale cosa allora questa viene rimossa dalla borsa e restituita dal metodo, altrimenti viene sollevata un'eccezione di tipo *CosaNonPresenteException* (che non include alcun messaggio o valore). **[opzionale]:** Utilizzare un iterator per l'implementazione del metodo *consumaCosa.*
- *calcolaValoreTotaleBorsa()*: calcola e restituisce la somma dei valori di tutte le cose presenti nella borsa.

Estendere la classe *Personaggio* con la classe *Contadino* (anch'essa nel pacchetto *fantasy*), il cui metodo costruttore crea ogni contadino con forza pari a 5.

Fornire un metodo *toString()* che restituisca tutti i dati del contadino.

Esercizio Java n. 2: Gioco Fantasy: il Mago

Arricchire l'applicazione Java dell'Esercizio 1 come segue.

Aggiungere alla classe *Personaggio* i metodi seguenti:

- *scambia(String tipologia, Personaggio altro)*: se il personaggio su cui viene eseguito il metodo e il personaggio *altro* possiedono entrambi una cosa di tipo *tipologia*, allora i due se la scambiano, altrimenti viene sollevata un'eccezione di tipo *CosaNonPresenteException*.
- *applicaDanno(int danno)*: riduce l'energia del personaggio del valore pari a *danno* se il danno è inferiore all'energia attuale del personaggio, altrimenti imposta la sua energia a 0.

Estendere la classe *Personaggio* con la classe *Mago* (anch'essa nel pacchetto *fantasy*) che deve implementare le interfacce *Combattente* e *Magico* (già fornite nel pacchetto *fantasy*). La forza di un mago non deve essere superiore a 6. Un mago è caratterizzato, oltre che dal suo nome, dalla sua energia e dalla sua forza, anche dal suo potere magico (intero da 1 a 10), che non deve essere inferiore a 5. E' possibile creare un mago in due modi:

1. specificando nome, forza e potere magico; se forza e/o potere magico del personaggio non vengono correttamente specificati, il personaggio non verrà creato e sarà generata un'eccezione di tipo *IllegalArgumentException*;
2. specificando solo il nome e generando numeri random per forza e potere magico che rispettino i vincoli sopra menzionati.

L'implementazione del metodo *attacca(Personaggio altro)* è la seguente: quando un mago esegue un attacco produce un danno pari al doppio della sua forza; il danno così calcolato viene quindi applicato ad *altro* e restituito dal metodo.

L'implementazione del metodo *eseguiIncantesimo(Personaggio altro)* è la seguente: quando un mago esegue un incantesimo produce un danno pari alla somma del suo potere magico e del risultato della divisione intera fra la sua energia e il valore 10; il danno così calcolato viene quindi applicato ad *altro* e restituito dal metodo.

Sovrascrivere il metodo toString() restituendo tutti i dati del mago.

Creare infine la classe *Main* (nel pacchetto di default) con un metodo statico *avviaGioco()* che crea un mago specificando esplicitamente nome, forza e potere

magico, gestendo l'eventuale eccezione di tipo *IllegalArgumentException* come segue: l'eccezione viene catturata e per gestirla viene creato un mago con valori random che rispettano sempre i vincoli relativi ai valori di forza e potere magico. Un volta che il mago è stato creato, indipendentemente dal metodo costruttore utilizzato, il metodo *avviaGioco()* stampa a video i dati del mago.