Esame di Ingegneria del Software – 9 CFU – 15 Luglio 2022

- Scrivere immediatamente, su ogni foglio che vi è stato consegnato, cognome, nome, numero di matricola.
- Non è consentito consultare appunti, libri, colleghi, né qualunque dispositivo elettronico, pena l'immediato annullamento della prova.
- Tempo a disposizione: 3 ore.

Esercizio 1

Si vuole realizzare un sistema per semplificare la gestione di partite in un gioco di ruolo fantasy. Il sistema permette ad un game master di gestire una o più partite. Per ciascuna partita, identificata da un nome e da una data di creazione, il game master può aggiungere o rimuovere utenti. Per aggiungere un nuovo utente, il game master deve indicare un nome, una classe (correntemente sono supportate le classi *guerriero*, *mago*, *furfante*) e delle statistiche (numeri interi positivi) per i tre attributi chiave di forza (STR), destrezza (DEX) e intelligenza (INT). La somma delle statistiche in STR, DEX e INT non può superare i 10 punti. Infine, il game master può finalizzare l'aggiunta dell'utente scegliendo (opzionalmente) un oggetto bonus da inserire nell'inventario del personaggio, selezionandolo tra quelli presenti nel sistema. Gli oggetti sono identificati da un proprio nome univoco, e possono essere armi (caratterizzate da un danno e da un requisito minimo di STR), armature (caratterizzate da una statistica di difesa) oppure pozioni (caratterizzate da una descrizione testuale dell'effetto).

a) Dettagliare il caso d'uso relativo alla funzionalità di aggiunta di un personaggio a una partita per mezzo di mock-up e descrizioni testuali strutturate secondo il formalismo di Cockburn. Usare la propria conoscenza del dominio per derivare dettagli non definiti nei requisiti.
b) Definire un class diagram di analisi del sistema, inteso come modello di dominio, relativo al caso d'uso della aggiunta di un personaggio a una partita. È possibile rifarsi alle euristiche *EBC*, e di *Abbott*.
c) Fornire un sequence diagram di analisi per il caso d'uso relativo all'aggiunta di un personaggio a una partita.

Esercizio 2

```
public float averageStudentAge(List<Person> list) {
	int sum = 0;
	boolean isEmpty = list.isEmpty();
	if(isEmpty) {
		return 0;
	}
	else {
		int size = 0;
		for(Person p : list) {
			boolean isStudent = p.isStudent();
			if(isStudent) {
				sum += p.getAge();
				size++;
			}
		}
		if(size==0) {
			return 0;
		}
		else {
			return (float)sum/size;
		}
	}
}
```

Si modelli con un sequence diagram un'invocazione del metodo `averageStudentAge` della classe `Utility` riportato sopra.

Esercizio 3

Il metodo `computeDamage` della classe `Player` viene utilizzato per calcolare la quantità di danno subita da un giocatore in seguito ad un attacco da parte di un nemico. Il metodo prende in input tre parametri:

- `int attackStrength` è un intero strettamente positivo rappresentante la forza dell'attacco nemico
- `int levelDiff` è un intero rappresentante la differenza di livello tra il nemico e il giocatore. Un valore positivo indica che il nemico è di livello più alto, un valore negativo indica il contrario.
- `boolean isCritical` è un flag booleano che indica se l'attacco infliggerà danni extra.

Il metodo `computeDamage`, se i parametri in input sono validi, calcola il danno subito dal giocatore moltiplicando la forza dell'attacco per un coefficiente dipendente dalla differenza di livello tra nemico e giocatore, indicato nella tabella seguente.

	`levelDiff` <-10	-10<= `levelDiff` <-5	-5 <= `levelDiff` <= 5	5<`levelDiff` <=10	`levelDiff` > 10
Coefficiente	0.1	0.5	1	2	3

Se il colpo è critico, ovvero se il flag `isCritical` è attivo, il valore risultante dal prodotto precedente è a sua volta moltiplicato per due. Inoltre, se uno degli argomenti non è valido, il metodo solleva una `IllegalAttackException`.

a) Indicare le classi di equivalenza per il metodo `computeDamage`.
b) Scrivere quattro test JUnit con strategia Black Box per il metodo `computeDamage`, indicando per ciascuno di essi quali classi di equivalenza copre. Si richiede inoltre che un test corrisponda a scenari in cui i parametri non sono validi, e che i restanti tre corrispondano a scenari in cui i parametri sono validi.
c) Quanti test sono necessari per testare il metodo con strategia SECT?

Esercizio 4

Si descriva il design pattern observer, evidenziandone in particolare vantaggi e svantaggi.