Classi
Attributi
Responsabilità
Associazioni

Esame di Object Orientation (Gr. 2) – 20 marzo 2023

- *Scrivere immediatamente su ogni foglio che vi è stato consegnato Cognome, Nome, N° Matricola.*
- *Non è consentito consultare appunti, libri, o colleghi, né qualunque dispositivo elettronico, PENA IMMEDIATO ANNULLAMENTO DELLA PROVA*
- *Tempo a disposizione: 3 ore*

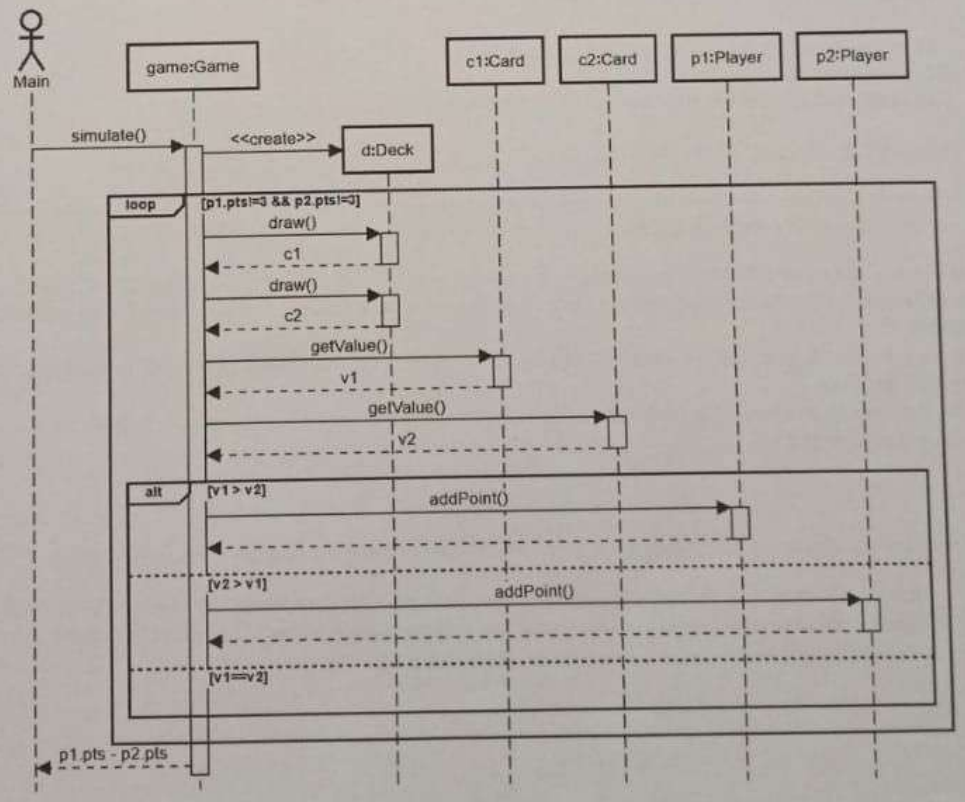
Esercizio 1

Vi viene commissionata la realizzazione di un software per la gestione di Scenari di test di sistemi elettronici (e.g.: decoder satellitari, registratori di cassa, etc.). Uno Scenario di test è caratterizzato da un nome univoco e da una sequenza di Passi (o *Step*) da eseguire nell'ordine dato. Ciascun passo può essere di due tipi: (1) *azione* oppure (2) *asserzione*. Le azioni sono caratterizzate da una descrizione testuale dell'azione da effettuare (e.g.: "*premi il pulsante X*"), mentre le asserzioni sono caratterizzate da una descrizione in testo libero della condizione da verificare (e.g.: "*il led Y deve essere accesso*") e da un livello di severità che può assumere valori in {BASSO, MEDIO, ALTO}. Il sistema permette a un *Test Engineer* di creare (e successivamente modificare) Scenari di test. Per creare uno Scenario di test, il *Test Engineer* seleziona innanzitutto il dispositivo elettronico cui il test è riferito tra quelli presenti nel sistema. Successivamente, l'*Engineer* può specificare una sequenza arbitrariamente lunga di Step. Si sottolinea che è necessario che il sistema tenga traccia dell'Engineer che ha creato un particolare scenario di test.

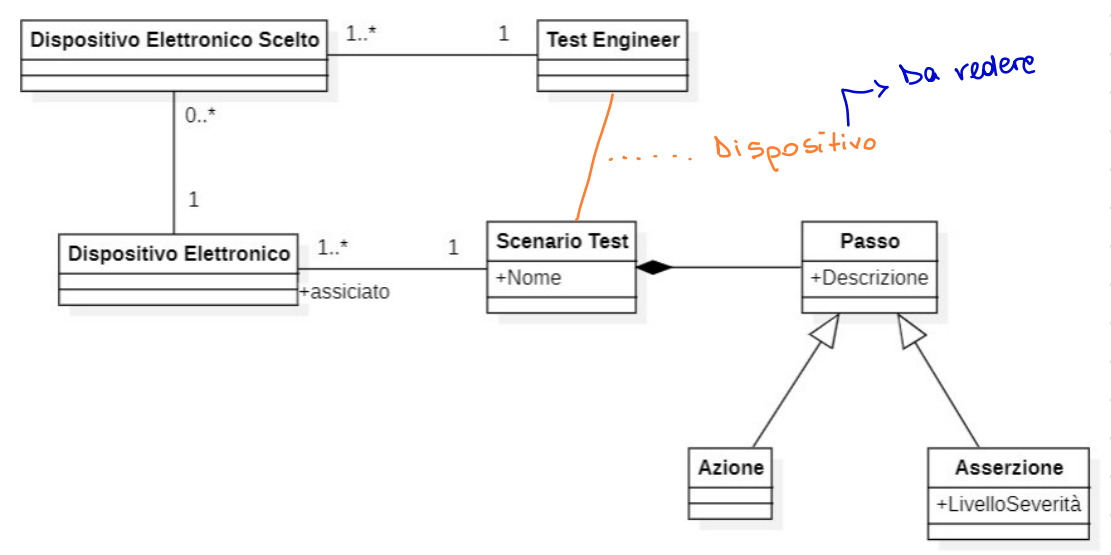
Si richiede di definire (a) Class Diagram e (b) Sequence Diagram per modellare la funzionalità di **inserimento di uno scenario di test**, eventualmente aiutandosi con delle rappresentazioni grafiche dell'interfaccia utente (mock-up). Il Class Diagram e il Sequence Diagram, intesi come modello di dominio, devono rifarsi all'euristica Entity-Boundary-Control (EBC).

Esercizio 2

Si scriva tutto il codice Java che è possibile desumere dal seguente sequence diagram.



ESERCIZIO 1



ESERCIZIO 2

Esercizio 3

```
public class Evaluator {
    List<String> answers;
    public Evaluator(List<String> a ){
        this.answers=a;
    }

    public String evaluate() {
        double vote = 0;
        for(String e : answers){
            vote+= e.length()*6;
        }
        vote=vote/4;
        if(vote<18){
            System.out.println("VOTO MINORE DI 18!");
            throw new Exception("Esame non superato, voto:"+vote);
        }
        return "Esame superato con valutazione: "+vote;
    }
}

public static void main(String args[]) {
        List<String> a = new List<>(); → Sbagliato
        a.add("");
        a.add("abcd");
        a.add("abcde");
        a.add("ab");
        Evaluator evaluator = new Evaluator(a);
        System.out.println("VALUTAZIONE IN CORSO...");
        try {
            System.out.println(evaluator.evaluate());
        } catch (Exception e) {
            System.out.println("Bocciato!"+e.getMessage());
        }
        System.out.println("FINE VALUTAZIONE");
}
```

Si considerino le classi java riportate sopra (si assuma che ogni classe sia definita in un file apposito).

a) Il codice compila correttamente? Motivare la risposta e fornire possibili correzioni in caso di risposta negativa.
b) Al netto delle eventuali correzioni apportate al punto precedente, qual è l'output di una esecuzione del metodo *main* della classe *Main*?
c) Nel codice è presente un esempio di *overriding*? Se sì indicare dove. Se no, indicare come è possibile modificare il codice per introdurne un esempio.
d) Nel codice è presente un esempio di *overloading*? Se sì indicare dove. Se no, indicare come è possibile modificare il codice per introdurne un esempio.

Esercizio 4

Spiegare la differenza tra associazioni, aggregazioni e composizioni nel contesto di class diagram di dettaglio UML.

Scrivere almeno un esempio per ognuna di tali relazioni, sia in termini di disegno del diagramma che con un opportuno spezzone di codice di esempio. Gli esempi devono riguardare concetti relativi ai componenti di un personal computer (ad esempio processore, scheda madre, memoria, mouse wireless e corrispondente ricevitore bluetooth).

ESERCIZIO 3

a) Non compila è sbagliato `List<String> a = new List<>()`. Deve essere `List<String> a = new ArrayList<>();`

b) L'output sarà:

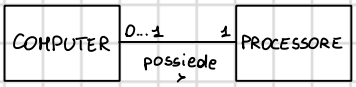
```
VALUTAZIONE IN CORSO...
VOTO MINORE DI 18!
Bocciato! Esame non superato, voto: 16,5
FINE VALUTAZIONE
```

c) Non è presente un overriding. Per ottenerlo, la classe Evaluator dovrebbe avere una classe che la estende contenente il metodo evaluate() con lo stesso prototipo ma diversa implementazione.

d) Non è presente overloading. Per ottenerlo potremmo aggiungere un altro costruttore alla classe Evaluator con un diverso numero o tipo di parametri.

ESERCIZIO 4

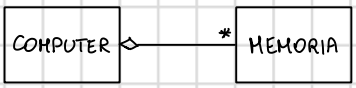
L'associazione rappresenta una relazione tra due classi in cui un oggetto di una classe è collegato a uno o più oggetti di un'altra classe. Un esempio può essere:




```
public class Computer {
    Processore processore;
    public Computer(Processore p) {
        this.processore = p;
    }
}
```

```
public class Processore
    Computer computer;
}
```

L'aggregazione è una relazione più forte dell'associazione, in cui un oggetto di una classe possiede o contiene uno o più oggetti di un'altra classe. Tuttavia, gli oggetti possono esistere indipendentemente dalla classe che li contiene.




```
public class Computer {
    ArrayList<Memoria> listaMemorie;
}
```

```
public class Memoria {
    Computer computer = null;
}
```

La composizione è una relazione ancora più forte dell'aggregazione, in cui l'oggetto di una classe è composto da uno o più oggetti di un'altra classe e la vita degli oggetti contenuti dipende dalla vita dell'oggetto contenitore.