**Elaborato**

Progetto: Seconda parte

Gruppo: Simona Ramazzotti (mat), Reda Kassame (mat)

Richieste da fare: in rosso

Richieste fatte: in verde

**Punto 1)**

Diagramma delle classi

Diagramma dei package

Discussione applicazione del principio di separazione modello-vista.

Per definizione gli oggetti del modello, ovvero gli oggetti del dominio, non devono avere una conoscenza diretta degli oggetti della vista, ovvero dell’interfaccia utente. Perciò gli oggetti della Vista non dovranno comunicare direttamente con gli oggetti del modello. Questo anche e soprattutto per minimizzare l’impatto sullo strato del dominio (oggetti del modello) dei cambiamenti dei requisiti relativi all’interfaccia, quindi con la possibilità di sostituire completamente l’interfaccia senza agire sul modello, poiché entrambi indipendenti.

Perciò i ruoli sono ben definiti, infatti abbiamo che lo strato Modello – che rappresenta la logica applicativa – fornitore dei metodi per accedere ai dati di interesse e lo strato Vista che si occupa dell’interazione con l’utente e permetta la visualizzazione dei dati processati dal modello.

Dunque abbiamo cercati di utilizzare questo principio nel progetto, separando tutta la logica applicativa dallo strato della Vista, facendo in modo che l’utente inserisse/scegliesse le richieste senza comunicare direttamente con i metodi presenti nel Modello.

(aggiungere qualcosa visti i diagrammi)

**Punto 2)**

Casi d’uso uno o più.

Svolgere:

* Diagramma di sequenza
* Determinare le operazioni di sistema (che si evidenziano dal diagramma di sequenza)

Per tutte o una parte delle operazioni di sistema, definire il contratto ( post e pre condizioni) e discutere l’assegnazione delle responsabilità mediante l’uso di pattern GRASP mostrando e commentando i diagrammi di sequenza o comunicazione in modo da mostrare l’uso di al più 3 diversi pattern.

Prendiamo in esempio il caso d’uso

* ricerca risorsa ?
* creazione risorsa ?
* prestito risorsa ?
* login
* registrazione utente
* Caso d’uso della registrazione di un utente

|  |  |
| --- | --- |
| **Nome** | Registrazione nuovo utente |
| **Attore** | Utente |
| **Scenario Principale** | 1. L’utente inserisce 1 e preme invio per avviare la registrazione 2. Il programma chiede di inserire i vari dati 3. L’utente li inserisce e conferma ad ogni passaggio 4. A fine inserimento il programma notifica dell’avvenuta creazione dell’utente 5. Il programma stampa di nuovo il menu iniziale   Fine |
| **Scenario alternativo** | 1a. L’utente inserisce numeri diversi da quelli richiesti dal menu  Il programma avverte dell’input errato e visualizza il menu |
| **Scenario alternativo** | 3a. L’utente inserisce valori errati durante la registrazione  Il programma avverte dell’input errato e invita a inserire il dato nuovamente |

-Diagramma di sequenza

-Operazioni di sistema (descrizione dei metodi dei diagramma di sequenza)

Applicazione di al più 3 pattern grasp

**Precondizioni**: nome utente univoco, età utente > 18 anni.

**Postcondizioni**: utente registrato nella lista utenti nel Database.

Il Controller prende la richiesta dalla Vista, quindi dall’utente, e una volta passati tutti i dati il Controller Operators richiede al Database di fare dei controlli. Se questi controlli (condizione: l’utente deve avere la maggiore età) vanno a buon fine allora l’oggetto utente viene creato e registrato.

**Creator:** prevede di assegnare alla classe A la responsabilità di creare oggetti di B quando A contiene o aggrega oggetti di B, A registra B, A possiede i dati per iniziallizzare B (nostro caso).

Da fare

**Low Copuling**: assegnare responsabilità per tenere le dipendenze fra le classi più basse possibili nel caso in cui A crea oggetto di B.

Da fare

* Caso d’uso dell’accesso di un utente

|  |  |
| --- | --- |
| **Nome** | Accesso utente |
| **Attore** | Utente |
| **Scenario Principale** | 1. L’utente inserisce 1 per accedere al menu utenti 2. L’utente inserisce 2 e preme invio per avviare il login 3. Il programma chiede di inserire l’username e la password 4. L’utente li inserisce e conferma ad ogni passaggio 5. A fine inserimento il programma conferma la notifica dell’avvenuto accesso. 6. Il programma stampa di nuovo il menu iniziale   Fine |
| **Scenario alternativo** | 1a. L’utente inserisce numeri diversi da quelli richiesti dal menu  Il programma avverte dell’input errato e visualizza il menu |
| **Scenario alternativo** | 3a. L’utente inserisce username o password non presenti  Il programma avverte dell’input errato e invita a rieseguire l’accesso |

-Diagramma di sequenza

-Operazioni di sistema (descrizione dei metodi dei diagramma di sequenza)

Applicazione di al più 3 pattern grasp

**Precondizioni**: nome utente deve esistere, password deve essere associata al nome utente.

**Postcondizioni**: utente viene autenticato.

Il Controller prende la richiesta dalla Vista, quindi dall’utente, e una volta passati tutti i dati il Controller Operators richiede al Database di controllare che le informazioni fornite da tastiera siano corrette.

**Information Expert**: tale pattern afferma che l’assegnazione delle responsabilità deve essere affidata alla classe che possiede le informazioni necessarie per soddisfarle.

Per tale gestione del login dell’utente, quindi l’assegnazione di tale responsabilità, possiamo notare come oltre al Controller ( e Controller Operators), l’altra classe che ha le informazioni dirette necessarie per soddisfare tale responsabilità è il Database. Infatti tale classe registra l’elenco di tutti gli utenti e conosce il nome utente e la password univoci per l’identificazione del fruitore.

* Controller applicato su tutto il progetto + Low Copuling (Controller che comunica con Controller Operator o Resource in tanti casi d’uso)

Da fare

**Punto 3)**

Si discuta l’applicazione di al più 3 differenti pattern SOLID sulle classi del progetto, riportando e commentando una porzione di codice JAVA precedente al refactoring che viola il principio (se presente) e una porzione del codice JAVA successiva al refactoring che rispetta il principio.

Richieste da fare: in rosso

**Punto 4)**

Si discuta l’applicazione di al più 3 pattern GoF sulle classi del progetto, mostrando e commentando il diagramma delle classi coinvolte, un esempio di diagramma di sequenza o comunicazione o riportando il codice JAVA che mostra l’applicazione del pattern.

**Punto 5)**

Si discuta l’applicazione di al più 5 pattern di refactoring mostrando e commentando una porzione di codice JAVA prima e dopo l’attività di refactoring.