Scuola universitaria professionale della Svizzera italiana
Dipartimento tecnologie innovative

SUPSI

Interprete di comandi

per la simulazione di un File System

Studente/i
Tony Kolarek
Samuele Saporito
Lorenzo Lo Brutto

Docente
Giancarlo Corti

Corso di laurea
Ingegneria Informatica, I3A

Modulo
Software Engineering and Development II

Anno
2023/2024

Data
18 dicembre 2024

STUDENTSUPSI

Indice

- Contesto e motivazioni
- Problema
- Requisiti
- Tecnologie utilizzate
- Stato dell’arte
- Approccio utilizzato
- Struttura
- Risultati ottenuti
- Conclusioni e miglioramenti

Contesto e motivazione

- Progetto nell'ambito dell'ingegneria del software
  - Requisito per certificare il corso Software Engineering and Development II

- Collaborare su un progetto attraverso Gitlab:
  - Milestones, issues

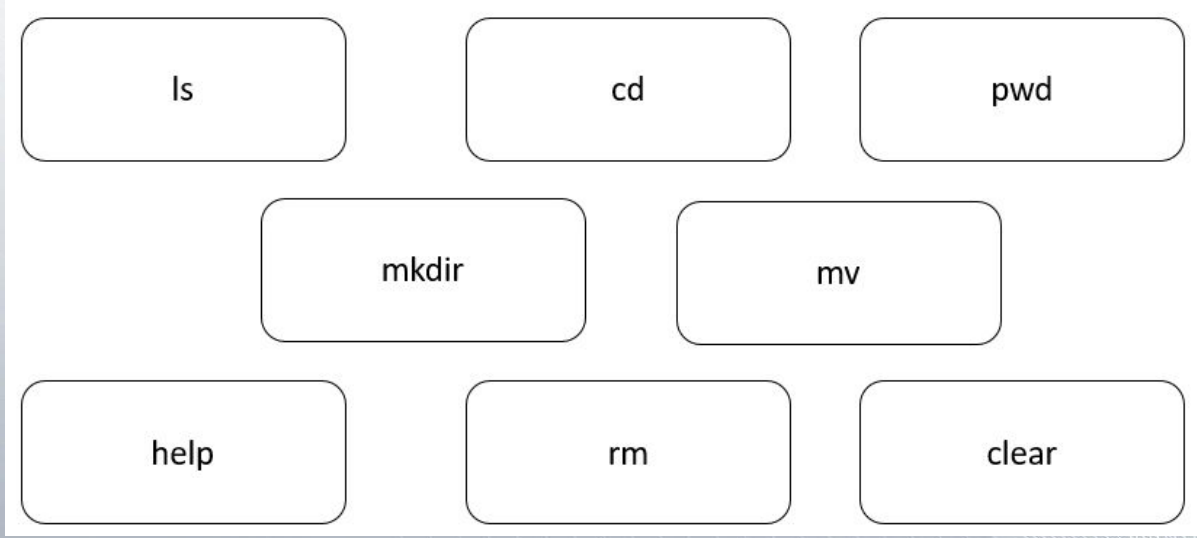
- Messa in pratica dei concetti teorici:
  - Pattern
  - Scrum (iterazioni settimanali)
  - Unit test

Problema

- Implementare una soluzione che permetta di interpretare dei comandi di terminale

Requisiti

- Personalizzazione della lingua e della grandezza dell'applicazione
- Possibilità di eseguire i seguenti comandi: pwd, mkdir, ls, cd, mv, rm, help, clear



Tecnologie utilizzate

- Linguaggio di applicazione: Java
- Graphic User Interface: JavaFX
- Versioning: Git
- Codebase management: GitLab
- Dependencies e build management: Maven
- Tests: JUnit e Mockito

Stato dell’arte

- Esistono soluzioni simili, più o meno avanzate
- La nostra applicazione funziona su più sistemi operativi
- Quest’ultima non dipende dal sistema operativo

Approccio utilizzato

- Pattern MVC (Model-View-Controller)
  - Separazione dei diversi layer (presentazione, interazione con l'utente e dati)
  - Controller → Model → Elaborazione dati → Controller → View

- Command pattern
  - Dispatching dei comandi
  - Controllo sintattico dei comandi

- Singleton pattern
  - Localizzazione

Struttura

- Suddivisione tra backend (engine) e frontend (client)

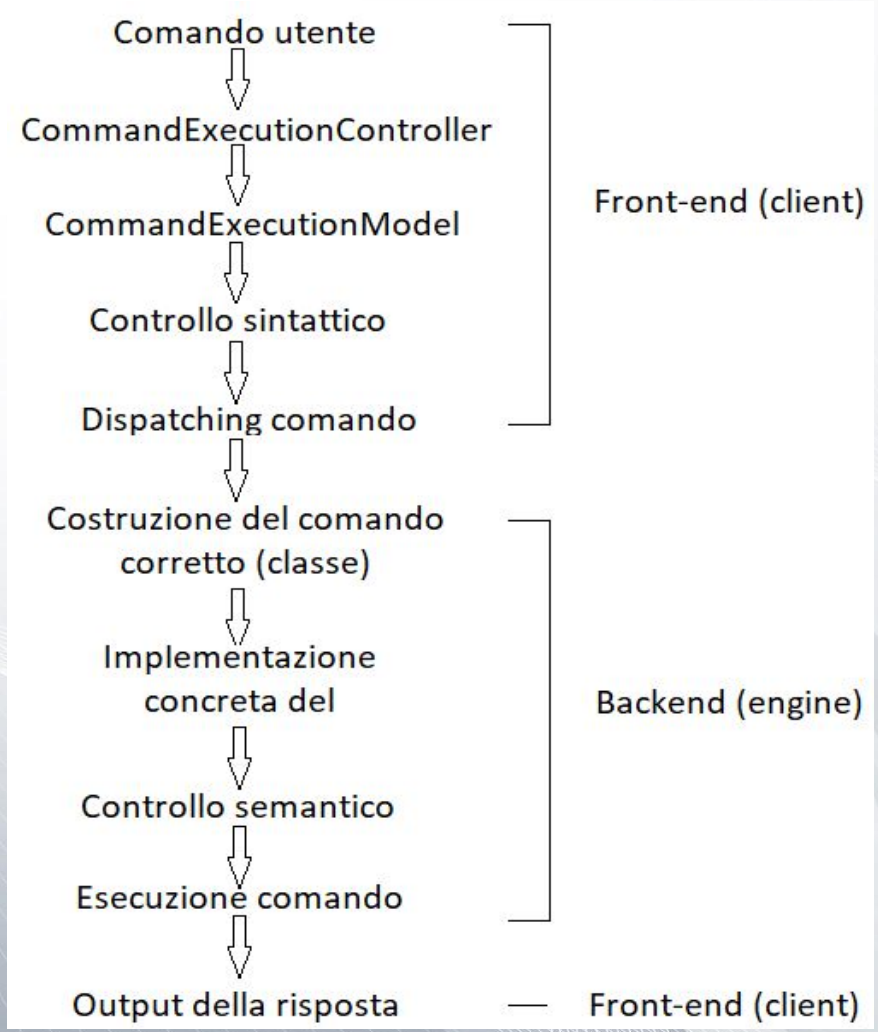
- Back-end: Logica dell'applicazione, elaborazione dei risultati e dei dati, localizzazione, implementazione dei comandi
  - Esempio: Implementazione del command pattern, logica del File System, logica delle preferenze dell'utente

- Front-end: Basato sul pattern MVC, controllo della GUI
  - Esempio: Creazione della GUI, dispatching dei comandi dell'utente

- In generale: Utente (Input) → Front-end → Back-end → Front-end → Utente (Output)

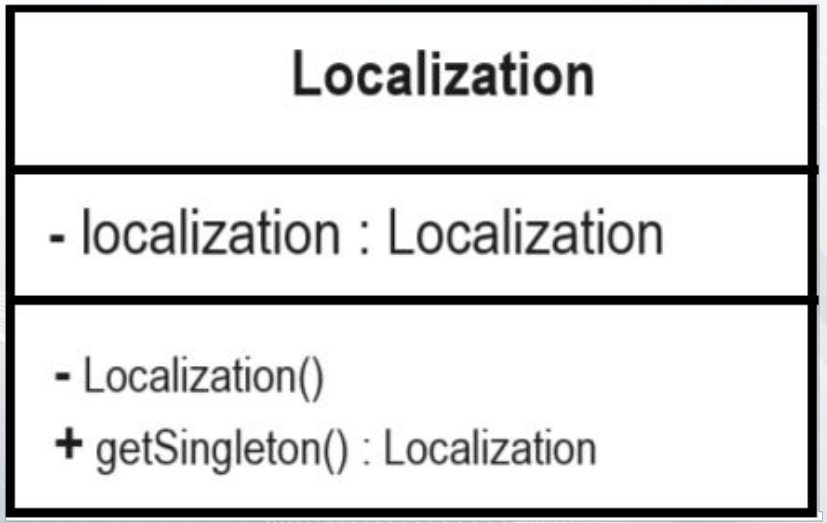
Esempio pattern: Command

- Approccio leggermente diverso da quello comune → uso di reflection e annotations



Esempio pattern: Singleton

- La localization
- Utilizzo nel classe Response
- Unica istanza



Esempio: gestione delle preferenze

- Preferences Model si occupa della logica delle preferenze
- Preferences Data si occupa della gestione del file di preferenze
- Approccio su delega: Preferences Controller delega a Preference Model, che delega a Preferences Data

Risultati ottenuti

- L’applicazione soddisfa tutti i requisiti prestabiliti
- I test implementati ci portano a concludere che l’implementazione funzioni correttamente

Conclusioni

- Ci riteniamo soddisfatti:
  - Conoscenza pratica dell'implementazione di alcuni pattern
  - Più comprensione dell'approccio agile
  - Applicazione pratica dei concetti visti nella teoria

- In futuro…
  - Migliorare ulteriormente il codice

STUDENTSUPSI

Possibili miglioramenti

- Implementazione dell'interpreter pattern
  - Per adesso il command pattern si occupa della grammatica dei comandi
  - L'interpreter pattern è più ragionevole → per definizione risolve il problema della grammatica
  - Il Command Pattern corrente funziona, ma:
    - il suo mantenimento è complesso: contesto generico, annotations, reflections…
    - l'interpreter pattern è più flessibile: estendere la grammatica è molto più semplice