Scuola universitaria professionale della Svizzera italiana
Dipartimento tecnologie innovative

SUPSI

Editor 2D

Gruppo 4

Studente/i	Relatore	Correlatore
Marco Lorusso Adriano Chiriacò	Amos Brocco Sandro Pedrazzini	

Corso di laurea	Modulo / Codice Progetto	Anno
Ingegneria Informatica	Software Engineering and Development II (M-I4040)	2024/2025

Committente	Data
Giancarlo Corti	16/12/2024

Indice

- Contesto
- Motivazioni
- Diagramma Use case
- Problema
- UML di attività
- Approccio
- Demo
- Conclusioni

Contesto

Contesto generale:

- Necessità di software per la manipolazione di immagini nei formati PNM (PBM, PGM, PPM)

Obbiettivi principali:

- Fornire un’applicazione per caricare, trasformare, salvare immagini nei formati PNM
- Garantire un’interfaccia utente intuitiva e localizzabile

Motivazione

Perché questo progetto è importante:

- Mancanza di strumenti user-friendly per gestire i formati PNM
- Opportunità di combinare concetti di programmazione avanzata: GUI con JavaFX, strutture MVC, e manipolazioni di array.
- Applicazione pratica di design patterns (Observer, Command, Strategy, Chain of Responsibility).

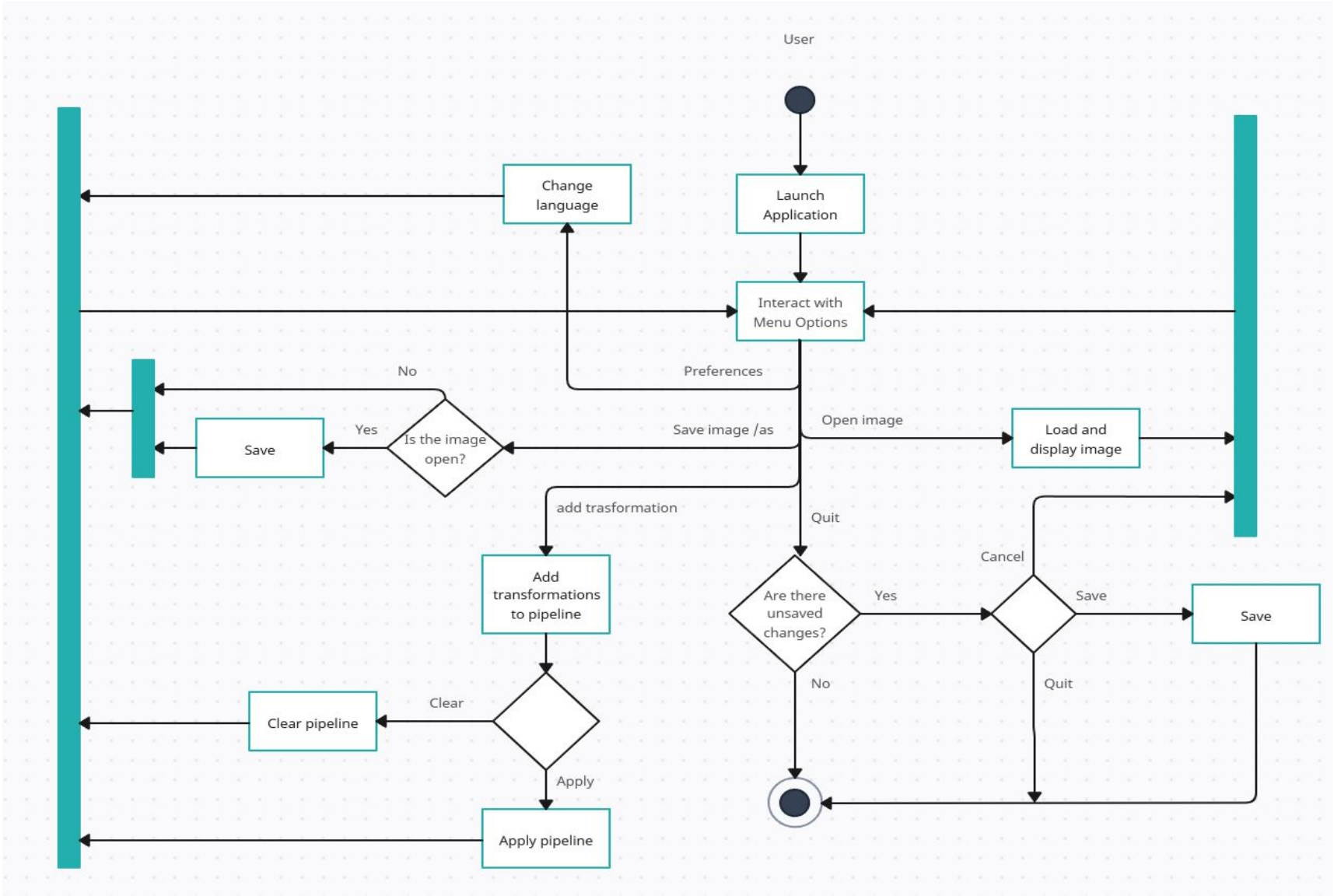
UML use case diagram

Problema

Problemi da risolvere:

- Supportare correttamente i formati PNM
- Implementare trasformazioni grafiche come flip, rotate, negativo
- Integrare un sistema multilingua per l'utente

UML di attività



Approccio

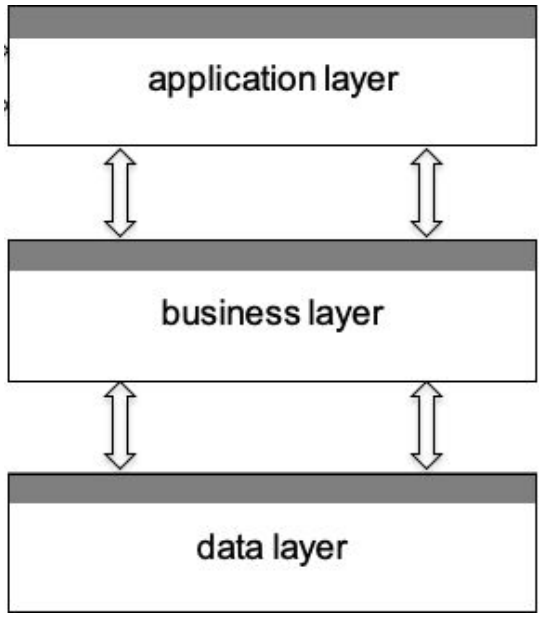
Architettura del sistema:

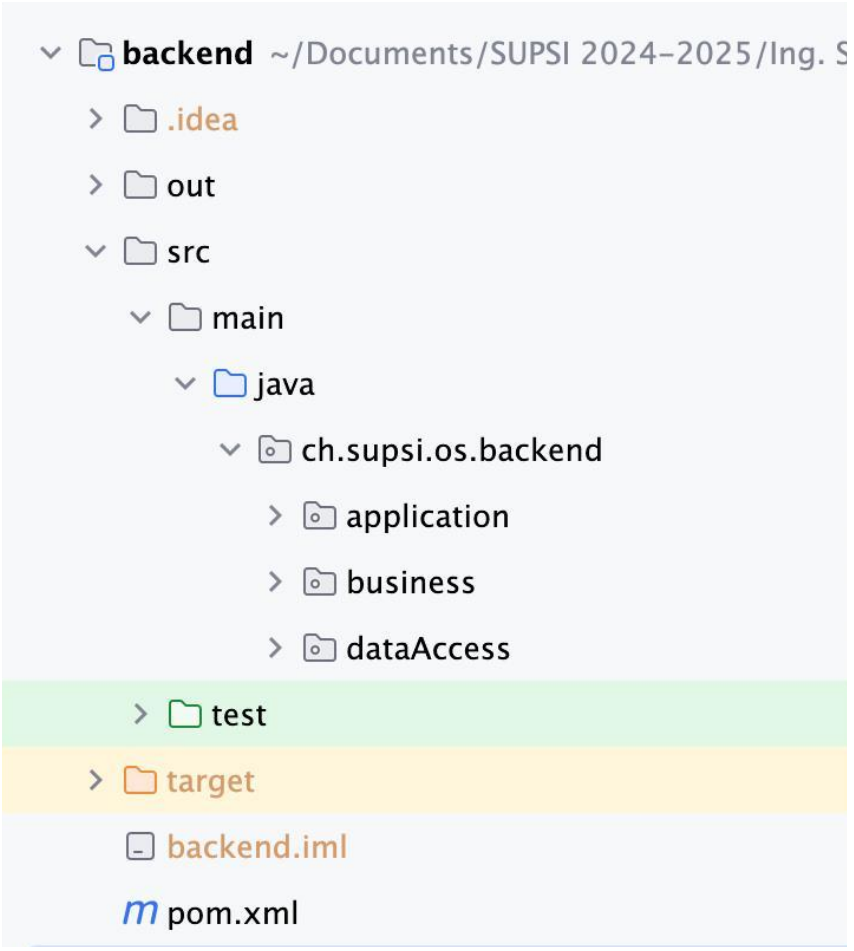
- Separazione di backend (business logic) e frontend (MVC)
- Uso di JavaFX per la GUI
- Testing automatizzato con JUnit e TestFX

Approccio

Software Design (Backend)

Multi Layer Architecture

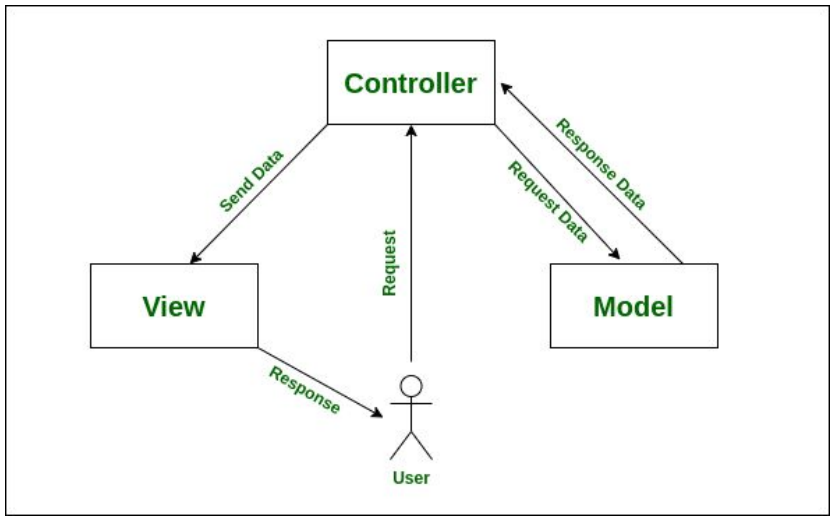


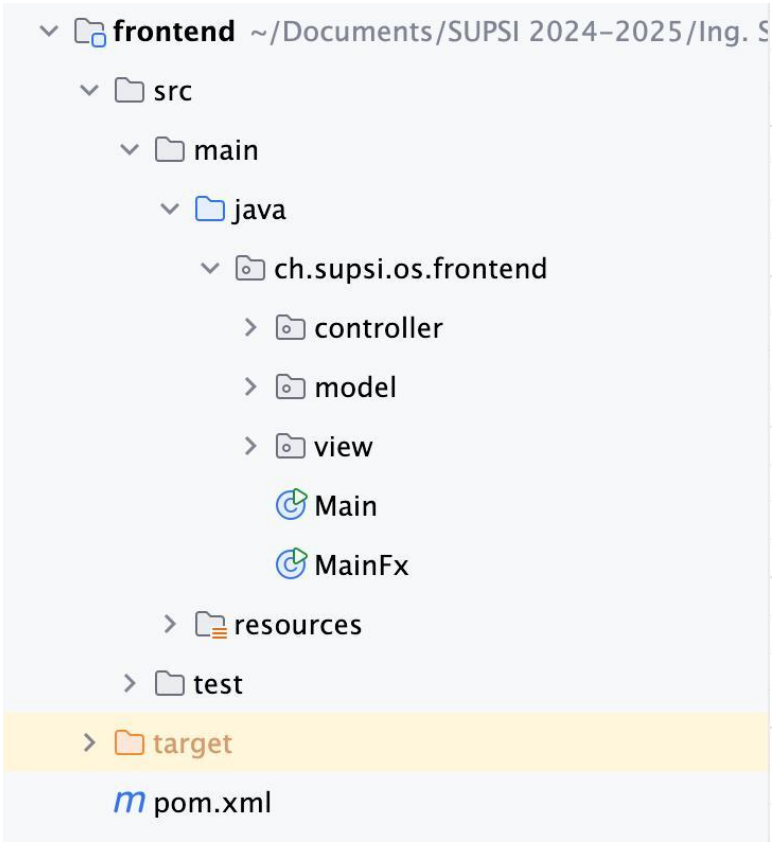


Approccio

Software Design (Frontend)

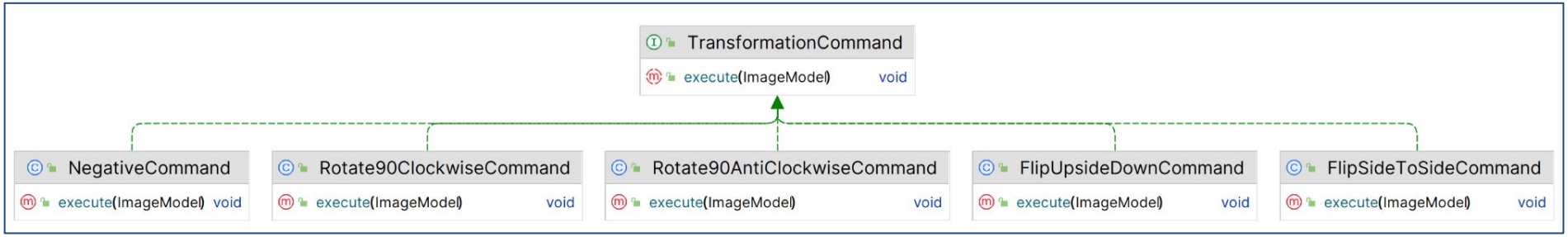
MVC Pattern





Approccio, command pattern(frontend)

Software Design




```
@Override
public void execute(ImageModel imageModel) {
    transformation.applyTransformation(imageModel);
}
```

Obiettivo:

- Incapsulare ogni operazione (trasformazione) come un comando separato.

Ruolo:

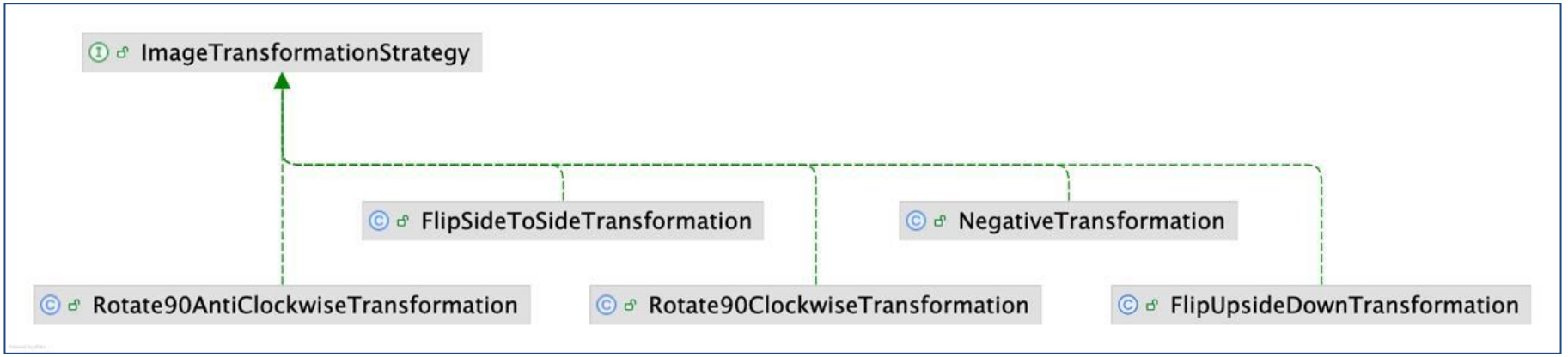
- Ogni comando può essere aggiunto alla pipeline e successivamente eseguito.

Vantaggi:

- Separazione tra logica UI e implementazione delle trasformazioni.
- Estensibilità: aggiungere nuovi comandi senza modificare codice esistente

Approccio, strategy pattern(backend)

Software Design



Obiettivo:

- **Definire un'interfaccia comune per tutte le trasformazioni delle immagini.**

Ruolo:

- **Ogni classe concreta fornisce una strategia specifica per modificare i pixel dell'immagine.**

Vantaggi:

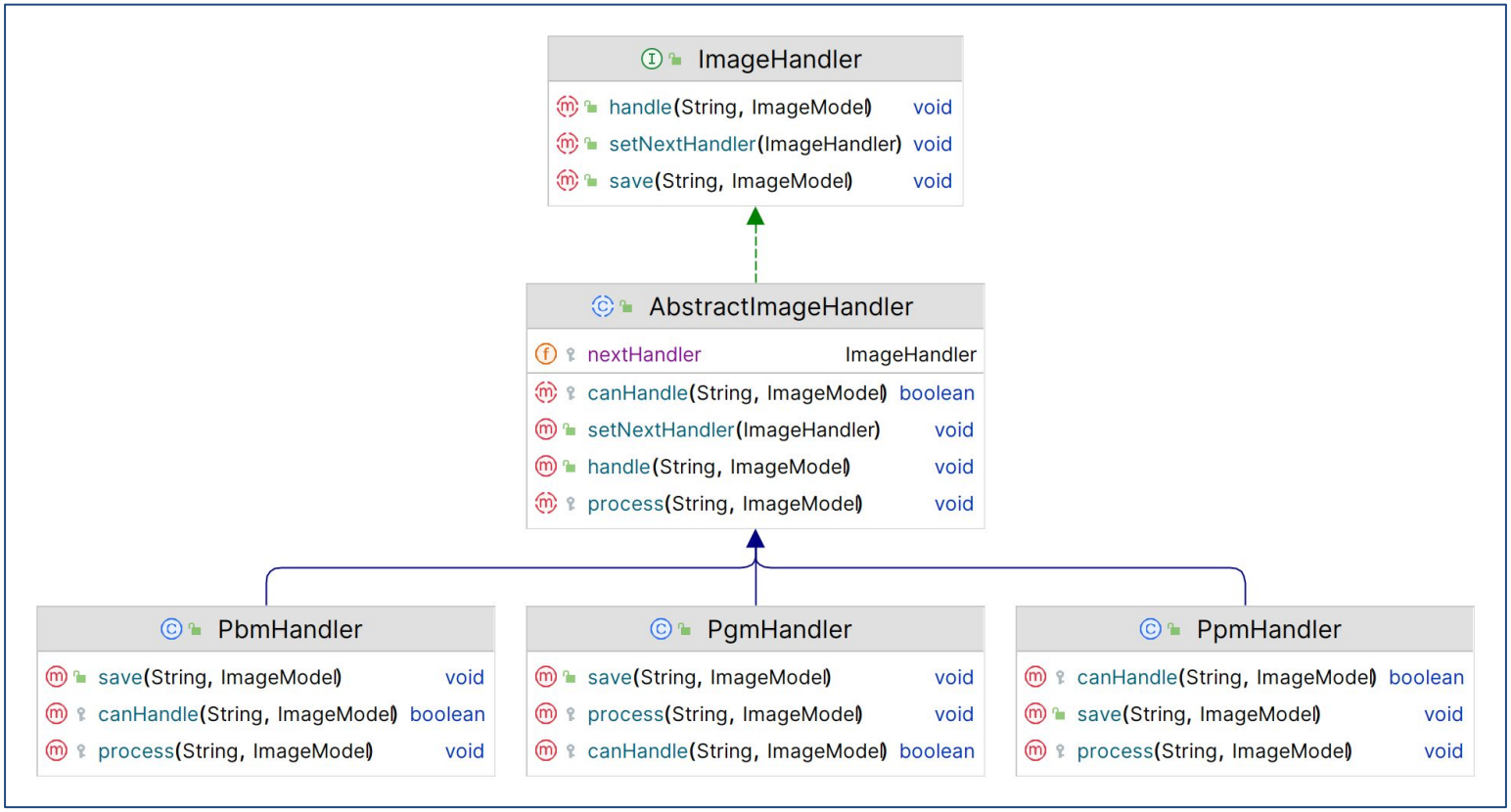
- **Separazione dell'algoritmo di trasformazione.**
- **Open/Closed Principle: aggiungere nuove trasformazioni senza toccare il codice esistente.**

```java
@Override
public void applyTransformation(ImageModel imageModel) {
    int[][] pixels = imageModel.getPixels();
    int width = imageModel.getWidth();
    int height = imageModel.getHeight();
    int channels = imageModel.getChannels();

    for (int i = 0; i < height; i++) {
        for (int j = 0; j < width / 2; j++) {
            if (channels == 3) {
                int leftIndex = j * 3;
                int rightIndex = (width - 1 - j) * 3;
                for (int k = 0; k < 3; k++) {
                    int temp = pixels[i][leftIndex + k];
                    pixels[i][leftIndex + k] = pixels[i][rightIndex + k];
                    pixels[i][rightIndex + k] = temp;
                }
            } else {
                int temp = pixels[i][j];
                pixels[i][j] = pixels[i][width - 1 - j];
                pixels[i][width - 1 - j] = temp;
            }
        }
    }
    imageModel.setPixels(pixels);
}
```

Approccio, chain of responsibility

Software Design

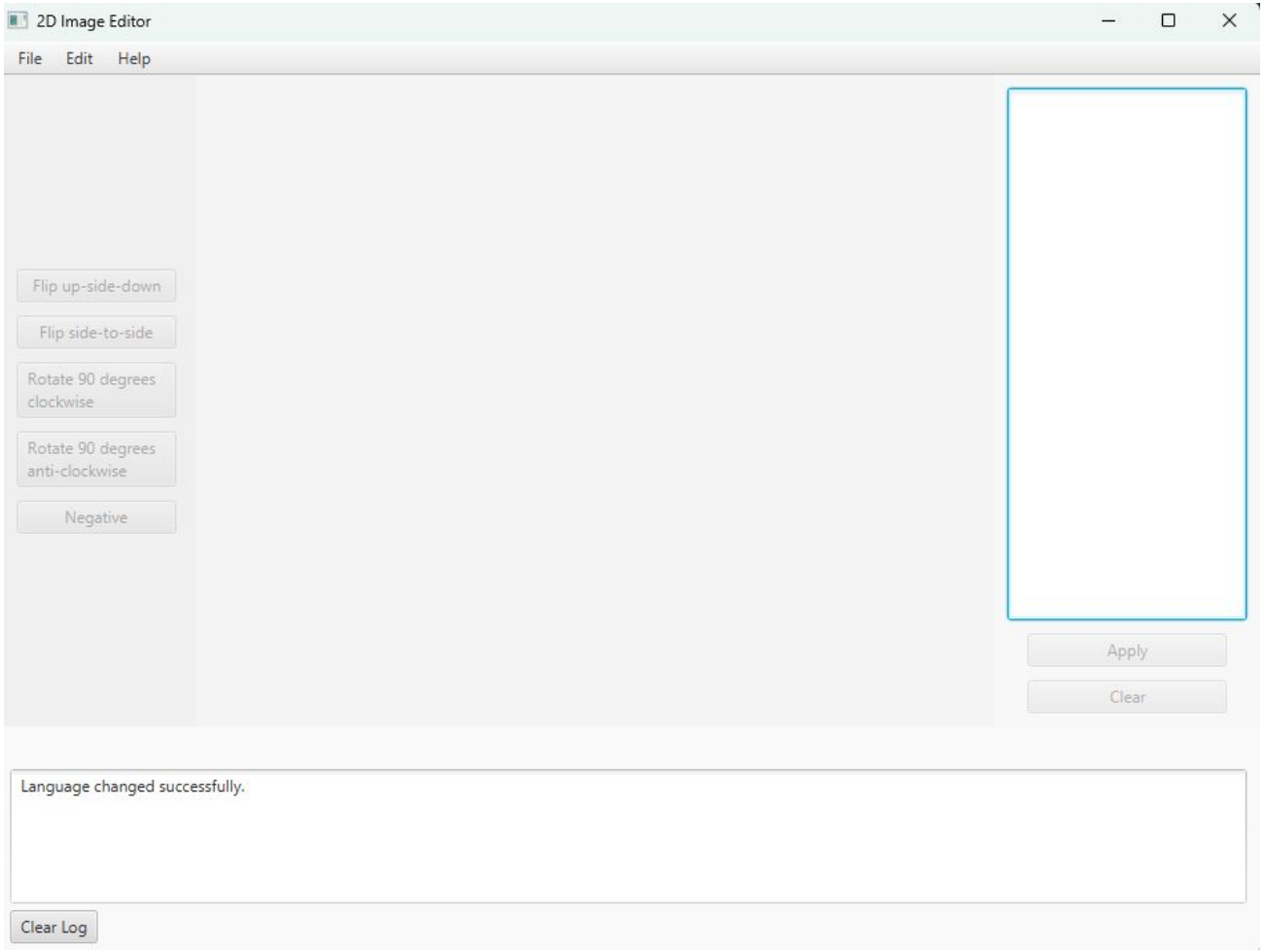


- **Obiettivo:** Gestire file in diversi formati PNM senza duplicare la logica.
- **Vantaggi:**
  - **Estensibilità:** Nuovi formati possono essere aggiunti senza modificare il flusso.
  - **Flessibilità:** Ogni handler è responsabile di un formato specifico.

```java
@Override
public void handle(String filePath, ImageModel imageModel) throws IOException {
    if (canHandle(filePath, imageModel)) {
        process(filePath, imageModel);
    } else if (nextHandler != null) {
        nextHandler.handle(filePath, imageModel);
    } else {
        throw new IllegalArgumentException("Unsupported image format");
    }
}
```

```java
@Override
protected boolean canHandle(String filePath, ImageModel imageModel) throws IOException {
    try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
        String magicNumber = reader.readLine().trim();
        return "P1".equals(magicNumber);
    }
}
```

Demo



Conclusioni

Cosa è stato raggiunto:

- Applicazione funzionale con GUI responsive e multilingua.
- Architettura ben strutturata e testabile.
- Supporto completo per formati PNM e trasformazioni.

Lezioni apprese:

- Importanza della modularità e dei design patterns.
- Come gestire testing automatizzato in un progetto complesso.

Futuri sviluppi:

- Supporto a ulteriori formati di immagini.
- Aggiunta di nuove trasformazioni grafiche.

Grazie per l’attenzione!