Design

Gruppo10

December 14, 2024

Contents

1 Design

Il documento racchiude l'insieme dei diagrammi che, secondo noi, rendono più comprensibile la struttura del software. Sono stati utilizzati diagrammi delle classi, di sequenza e dei package.

1.1 Diagramma delle classi

Il diagramma delle classi rappresenta la descrizione grafica di tutte le classi utilizzate, con realtivi metodi e attributi. Sono state prese in considerazione quattro classi.

- Persona
- Contatto
- Rubrica
- ImportExport

La classe Persona è dichiarata astratta; essa gestisce unicamente due attributi nome e cognome, con relativi get e set. È pensata per essere estesa da altre classi (come Contatto).
La classe Contatto invece rappresenta un contatto nella rubrica; essa estende la classe astratta Persona, alla quale aggiunge attributi specifici come una lista di numeri di telefono e una lista di indirizzi email. Fornisce dei metodi di gestione di questi, come l'aggiunta e la rimozione.
La classe Rubrica rappresenta la collezione principale di contatti. Come vediamo essa possiede un ArrayList di tipo Contatto per il salvataggio dei contatti. Include metodi per aggiungere, rimuovere e cercare contatti.
La classe ImportExport gestisce metodi di salvataggio su/da file esterni di tipo csv. I metodi statici, importRubrica ed exportRubrica, sono progettati per essere utilizzati senza dover creare un'istanza della classe.

Grazie alla suddetta suddivisione delle classi, riusciamo a far svolgere ad ognuna di esse solo i compiti strettamente legati alla classe stessa (SRP). Ciò ci permette di generare un'alta coesione tra le classi. Per quanto riguarda l'accoppiamento abbiamo diverse realazioni in base alle classi che prendiamo in considerazione.

- Accoppiamento tra Rubrica e Contatto: Per dati.
  La loro relazione è basata esclusivamente sullo scambio di dati, senza che Rubrica acceda direttamente alla struttura interna di Contatto.

- Accoppiamento tra Rubrica e ImportExport: Per controllo
  Rubrica passa informazioni di controllo che possono alterare il flusso del metodo, ad esempio, importRubrica che riceve il filename da cui importare i contatti.

- Accoppiamente tra Persona e Contatto: Per contenuti.
  Contatto estendendo Persona, accede direttamente agli attributi definiti nella classe Persona.

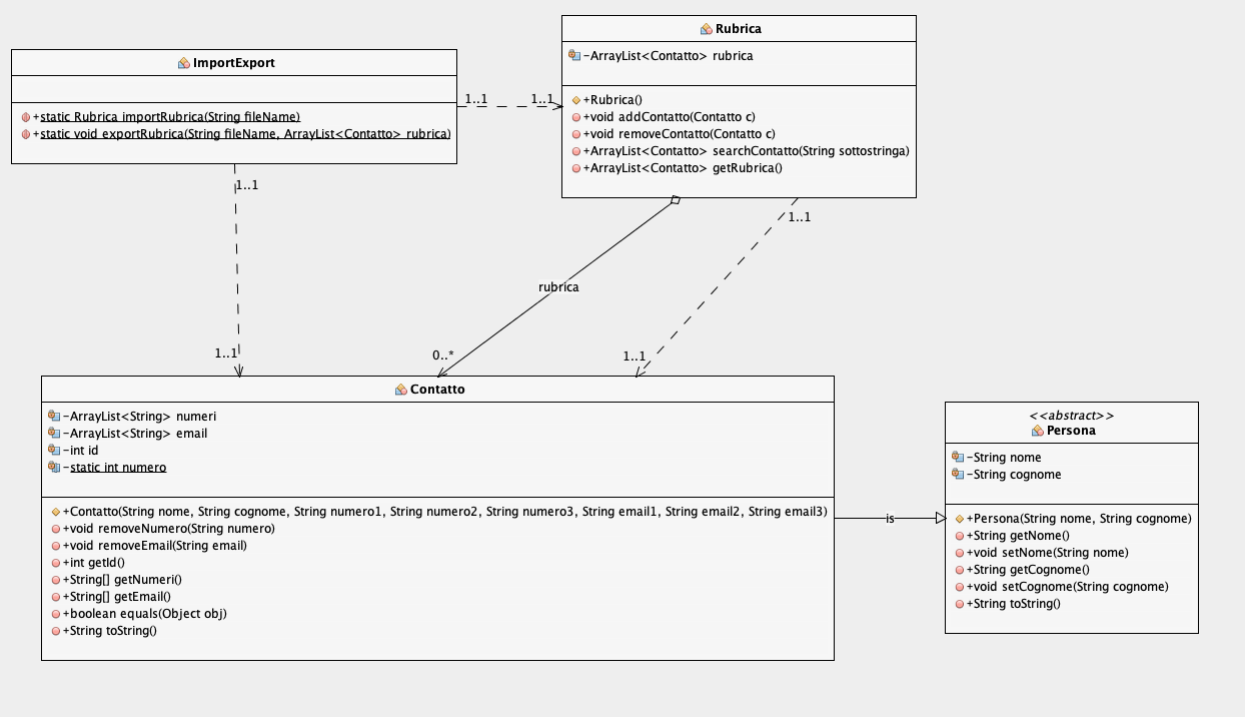

Figure 1: Diagramma delle classi

1.2 Diagrammi di Sequenza

I diagrammi di sequenza sono rappresentazioni grafiche servono a catturare il comportamento di un sistema in un singolo scenario, mostrando gli oggetti coinvolti e i messaggi scambiati tra di essi.

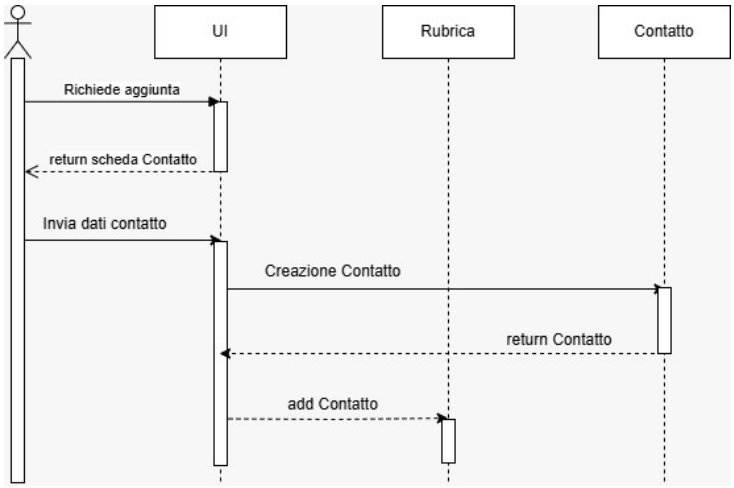

Figure 2: Aggiunta del contatto

La figura 2 illustra l'interazione degli oggetti nella fase di aggiunta di un contatto.

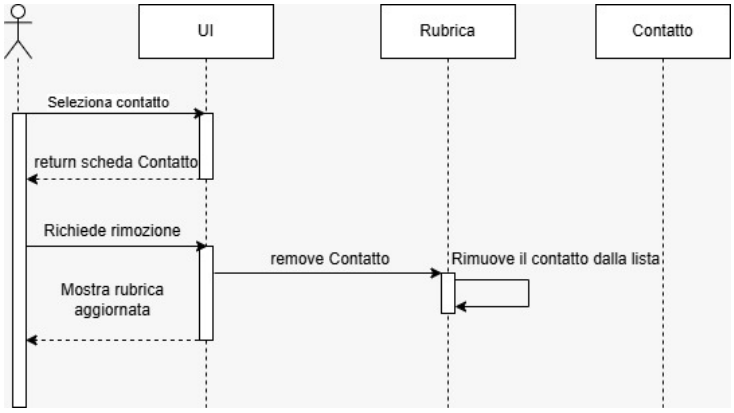

Figure 3: Rimozione contatto

La figura 3 illustra come interagiscono gli oggetti nella fase di rimozione di un contatto.

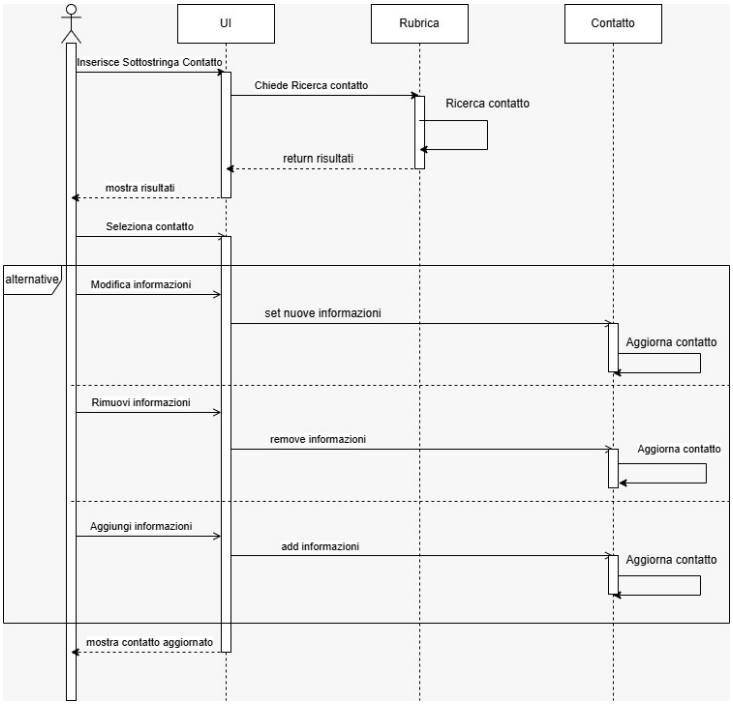

Figure 4: Modifica contatto

La figura 4 illustra come interagiscono gli oggetti nella fase di modifica di un contatto. Il diagramma presenta anche dei casi alternativi in cui l'utente decida di aggiungere o rimuovere un attributo al contatto

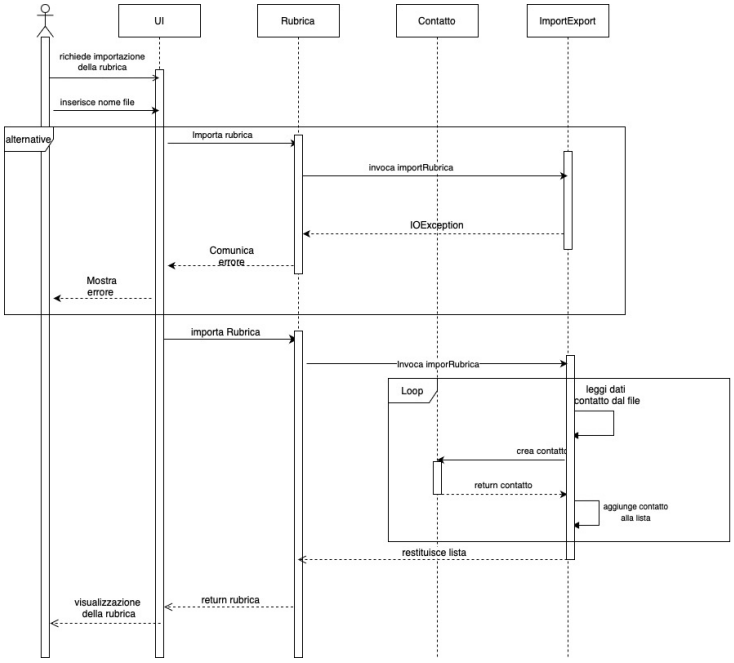

Figure 5: Importazione Rubrica

La figura 5 illustra come interagiscono gli oggetti nella fase di importazione della rubrica. Il diagramma presenta anche un caso alternativo generato da un'emissione errata del nome del file, da cui importare, da parte dell'utente.

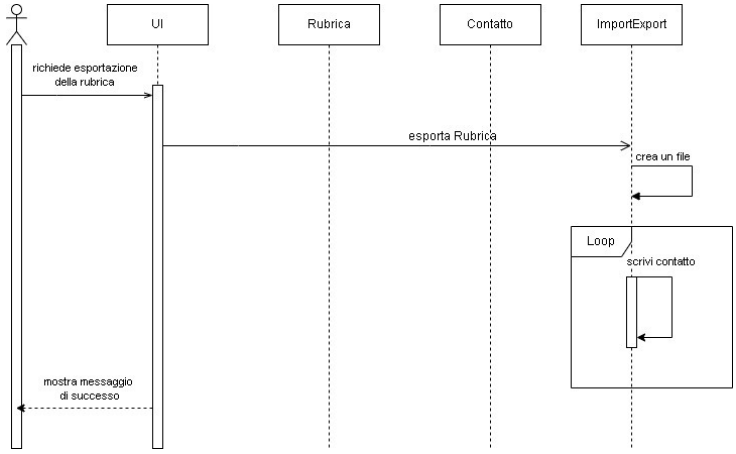

Figure 6: Esportazione Rubrica

La figura 6 illustra come interagiscono gli oggetti nella fase di esportazione della rubrica.

1.3 Diagramma dei package

Il diagramma dei package rappresenta l'organizzazione delle classi e delle risorse all'interno del progetto.

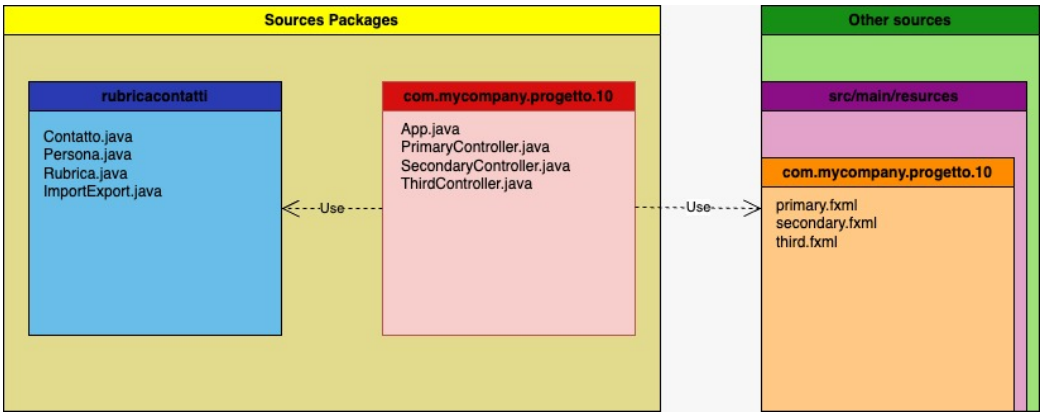

Figure 7: Diagramma dei package

- rubricacontatti
  - Contiene le principali classi per la gestione della rubrica
    - Contatto.java
    - Persona.java
    - Rubrica.java
    - ImportExport.java
- com.mycompany.progetto.10
  - Contiene i controller per la gestione della parte grafica
    - App.java
    - PrimaryController.java
    - SecondaryController.java
    - ThirdController.java
- Other sources/src/main/resources/com.mycompany.progetto.10
  - Contiene i file che caratterizzano la UI
    - Primary.fxml
    - Secondary.fxml
    - Third.fxml

2 Principi di buona progettazione

Un buon software non deve solo funzionare correttamente, ma deve anche favorire la manutenibilità nel tempo. Per questo nella progettazione della rubrica ci siamo basati sui principi SOLID:

2.1 S-ingle Responsability Principle

Un componente del codice deve svolgere un singolo compito ben definito. Come notiamo dalla Figura 1, abbiamo diviso i compiti in maniera tale che ogni classe svolga attività inerenti a se stessa.

- Rubrica gestisce la collezione dei contatti
- Contatto si occupa della gestione di email e numeri di telefono
- ImportExport è responsabile dell'interfacciamento con file esterni

2.2 O-pen Closed Principle

Le classi devono essere aperte all'estensione e chiuse alla modifica.

- La classe astratta Persona è progettatta per essere estesa, (vedi Contatto) ma allo stesso tempo si possono apportare modifiche per consentire il riutilizzo in un altro scenario.

2.3 L-iskov Substitution Principle

Gli oggetti devono poter essere sostituiti con istanze dei loro sottotipi senza alterare la correttezza del programma.

- La classe Contatto estendendo Persona, rende possibile che ogni istanza di Contatto possa essere trattata come una Persona.

2.4 I-nterface Segregation Principle

Un client non dovrebbe dipendere da metodi che non utilizza.

- Anche se nel progetto non sono presenti interfacce, ogni classe possiede metodi ben precisi e non contiene funzionalità inutili o ripetitive.

2.5 D-ependency Inversion Principle

i moduli di alto livello non devono dipendere dai moduli di basso livello, entrambi devono dipendere da astrazioni.

- Nel nostro caso non abbiamo sviluppato interfacce, in quanto intraprendere questa strada avrebbe comportato dispendio in termini di complessità del codice, nettamente superiore ai benefici che avremmo potuto ottenere.