Indice

[Diagramma delle classi](#_xej6hpqvmyxy)

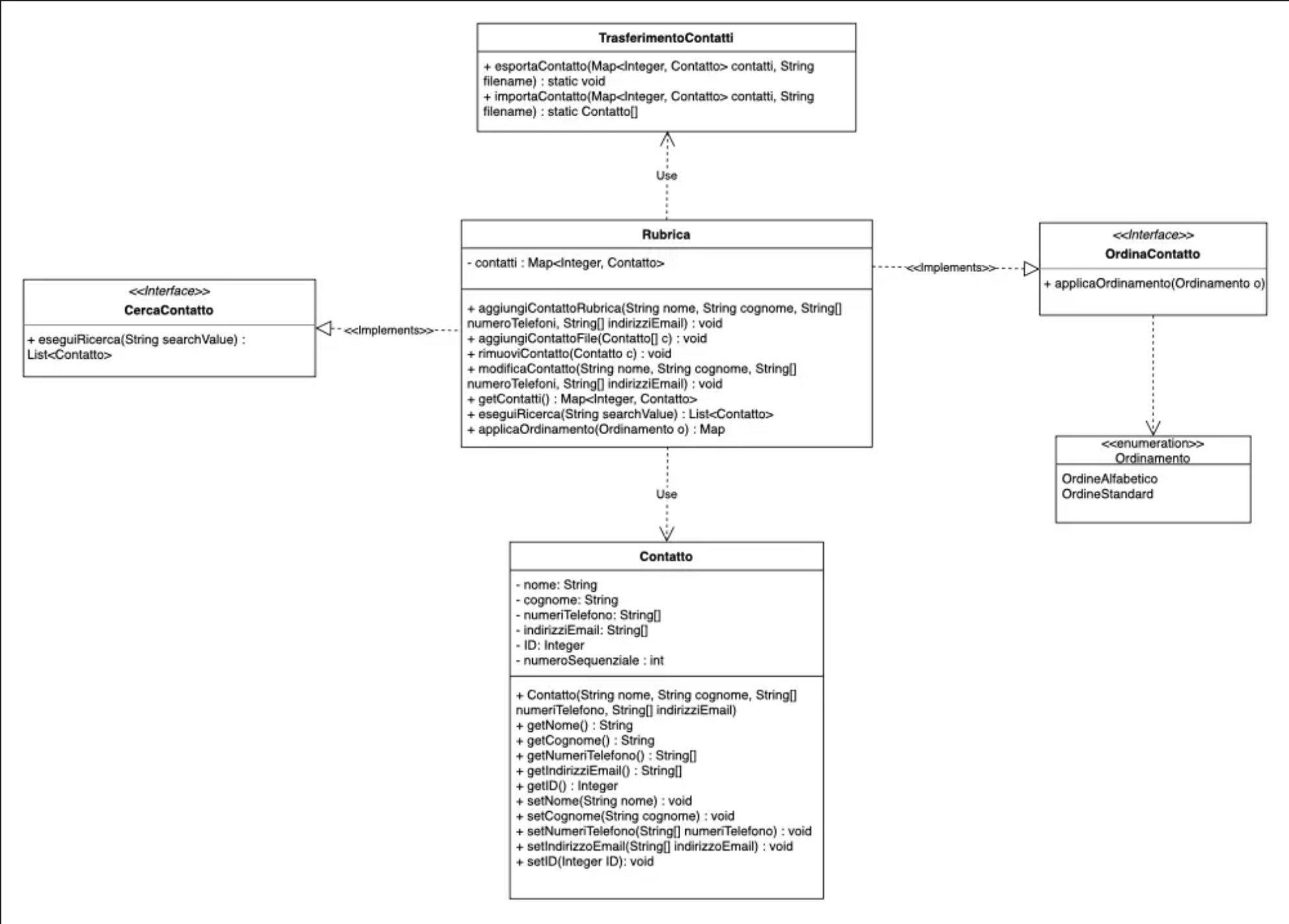
[Diagrammi delle sequenze](#_1ntlik6sg6g5)

1. [**Diagramma n°1: Aggiunta manuale del contatto in rubrica**](#_76j19sa8h46a)
2. [**Diagramma n°2: Modifica del contatto in rubrica**](#_l830ptt6rnck)
3. [**Diagramma n°3: Rimozione del contatto dalla rubrica**](#_7ho7ui736zxs)
4. [**Diagramma n°4: Salvataggio del contatto su file esterno**](#_e5zmf1je7wv1)
5. [**Diagramma n°5: Aggiunta di contatti da un file esterno**](#_efwmid1c0snx)
6. [**Diagramma n°6: Ricerca di un contatto in rubrica**](#_rim4n4c62zeo)
7. [**Diagramma n°7: Applica Ordinamento**](#_5avg6vv096ej)

# [Valutazione di Coesione e Accoppiamento](#_gzjr0kdyot8j)

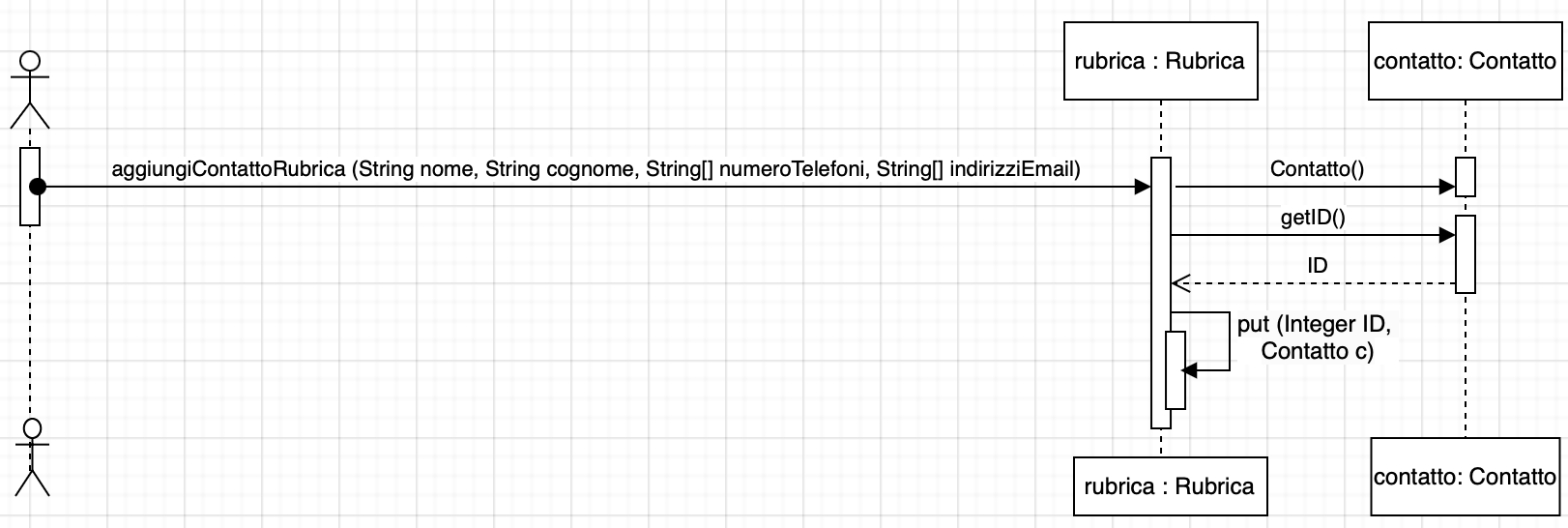
# [Valutazione principi di buona progettazione](#_6ccwwxswm9x6)

# Diagramma delle classi

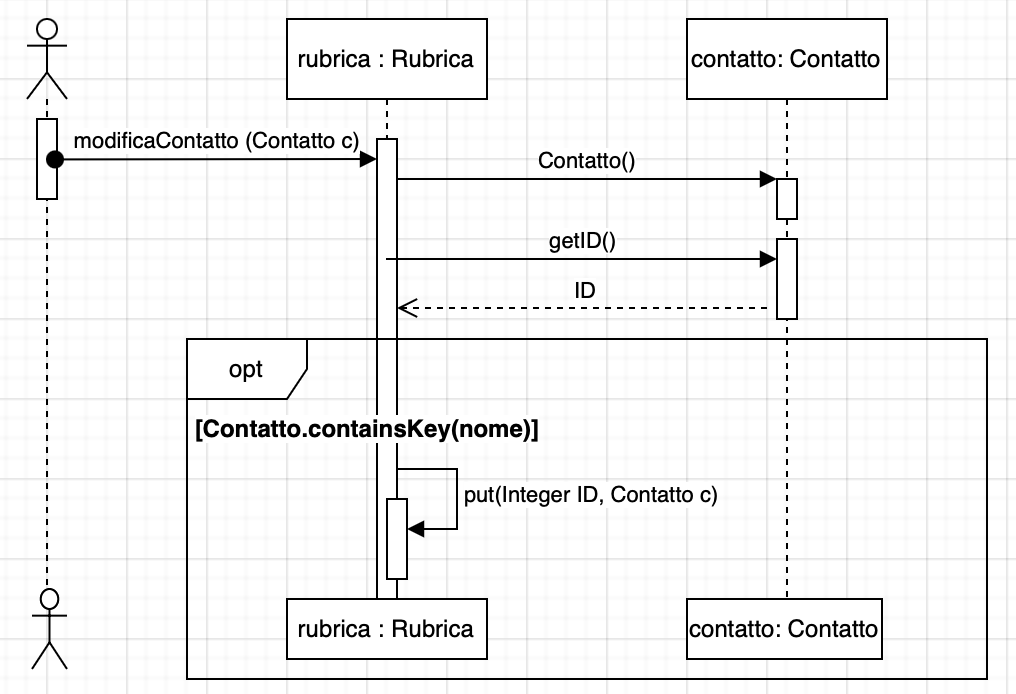


# Diagrammi delle sequenze

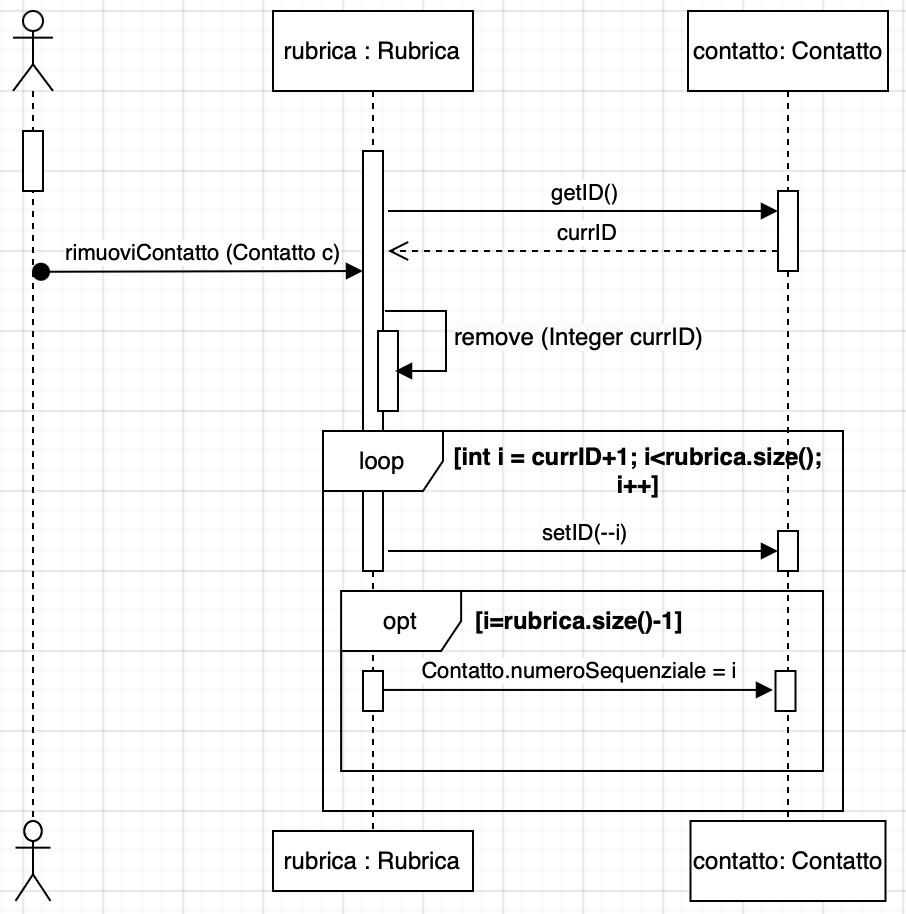
## Diagramma n°1: Aggiunta manuale del contatto in rubrica



## Diagramma n°2: Modifica del contatto in rubrica



## Diagramma n°3: Rimozione del contatto dalla rubrica



## 

## 

## 

## 

## 

## 

## 

## Diagramma n°4: Salvataggio del contatto su file esterno

## 

## Diagramma n°5: Aggiunta di contatti da un file esterno

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

## 

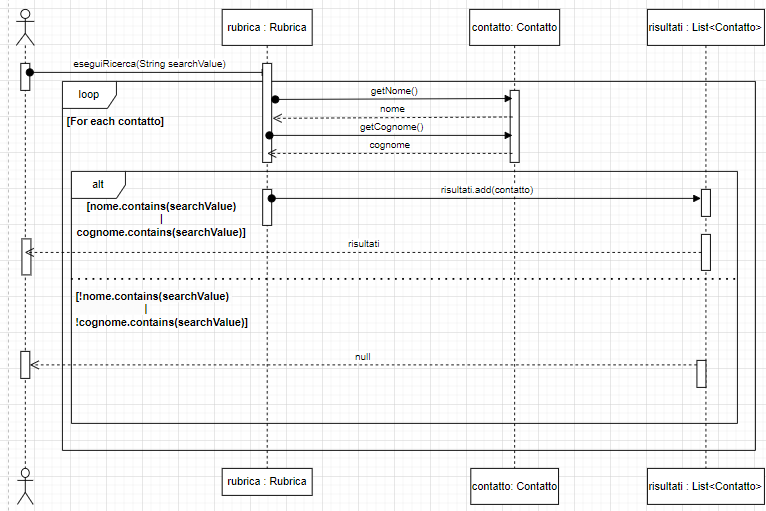
## 

## 

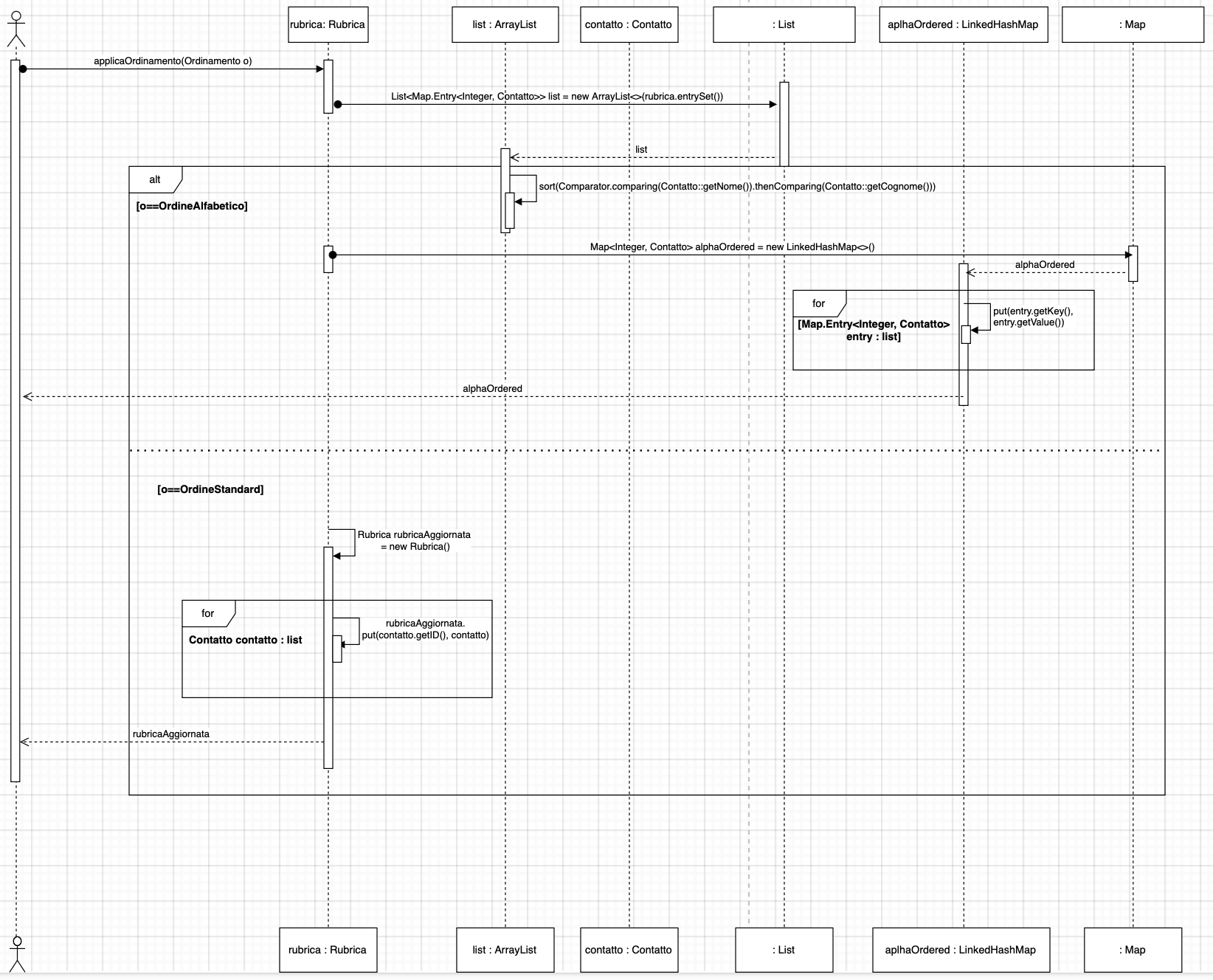
## 

## 

## Diagramma n°6: Ricerca di un contatto in rubrica



## Diagramma n°7: Applica Ordinamento



# Valutazione di Coesione e Accoppiamento

| **Classe** | **Livello di Coesione** | **Livello di Accoppiamento** | **Motivazione** |
| --- | --- | --- | --- |
| *Rubrica* | Coesione Comunicazionale | Accoppiamento per dati | Come livello di coesione è stato scelto il livello Comunicazionale, in quanto nel modulo sono presenti funzionalità che hanno obiettivi diversi (gestione contatti e visualizzazione contatti) ma lavorano sugli stessi dati.  Come livello di accoppiamento è stato scelto il livello per dati, in quanto il suddetto modulo passa solo informazioni strettamente necessarie e si fa passare altrettante informazioni strettamente necessarie. |
| *Contatto* | Coesione Funzionale | Accoppiamento per dati | Come livello di coesione è stato scelto il livello **Funzionale**, in quanto tutte le operazioni presenti sono strettamente necessarie e svolgono la stessa funzione su dati diversi, ma dello stesso ambito (dati del contatto).  Come livello di accoppiamento è stato scelto il livello di accoppiamento per dati in quanto questo modulo passa ad altri moduli solamente informazioni strettamente necessarie per il suo funzionamento ed è completamente indipendente. |
| *TrasferimentoContatti* | Coesione Comunicazionale | Accoppiamento per dati | Come livello di coesione è stato scelto il livello Comunicazionale, in quanto nel modulo sono presenti funzionalità che hanno obiettivi diversi (uno legge e uno scrive) ma lavorano sugli stessi dati (stesso file).  Come livello di accoppiamento è stato scelto il livello per dati, in quanto il suddetto modulo passa solo informazioni strettamente necessarie. |

# 

# Valutazione principi di buona progettazione

* Per rispettare il principio **KISS** (*Keep It Simple, Stupid!*), abbiamo realizzato delle classi e dei metodi che potessero essere gestite con operazioni semplici, evitando di rendere la progettazione eccessivamente astrusa. Inoltre le interfacce e le astrazioni non aggiungono complessità superflua.
* Per rispettare il principio **DRY** (*Don’t Repeat Yourself*), abbiamo utilizzato delle interfacce per far sì che queste astrazioni che implementano comportamenti generici possano essere implementate nelle classi senza duplicare la logica.
* Per rispettare il principio **YAGNI** (*You Aren’t Going to Need It*), abbiamo evitato di implementare funzionalità non necessarie, concentrandoci sulle richieste del progetto.
* Abbiamo inoltre gestito i moduli in maniera tale da far svolgere compiti diversi a ognuno di essi, evitando di caricare troppo i vari moduli a livello di quantità di lavoro da fare. L’unica eccezione si trova in *applicaOrdinamento*, dove lo stesso modulo è incaricato di ordinare i contatti in ordine alfabetico oppure standard. Tuttavia ciò non va a gravare sulla gestione del progetto in quanto le classi e i moduli restano di semplice implementazione e manutenibilità.
* Per quanto riguarda la **solidità**, ci siamo concentrati sull'applicazione dei principi di solidità per garantire un sistema robusto, manutenibile e semplice da estendere nel tempo. La progettazione è stata pensata seguendo un **approccio modulare**.
* Abbiamo dato importanza al principio della **singola responsabilità**, infatti ogni classe è stata progettata per svolgere un unico compito ben definito. Ad esempio, la classe *Rubrica* si occupa esclusivamente della gestione dei contatti, mentre la classe *TrasferimentoContatti* è dedicata all’importazione e all’esportazione.
* Il principio **Open/Closed** è stato rispettato attraverso l’uso di interfacce come *CercaContatto* e *OrdinaContatto*, che ci permettono di aggiungere nuove funzionalità senza modificare il codice esistente. Questo garantisce flessibilità e adattabilità del sistema a futuri cambiamenti.
* Abbiamo prestato attenzione al principio di **segregazione delle interfacce**, evitando di creare interfacce troppo grandi o generiche. Le interfacce specifiche ci hanno consentito di ridurre la dipendenza tra i moduli.
* Infine, il principio di **inversione della dipendenza** è stato applicato facendo dipendere le classi di alto livello (come *Rubrica*) da astrazioni piuttosto che da implementazioni concrete. Questo non solo riduce l'accoppiamento, ma favorisce anche la riusabilità del codice.