Relazione Progetto Esame Paradigmi di Programmazione

Luca Cavagnaro 20028568, Mattias Fara 20029621

1 Indicazioni iniziali

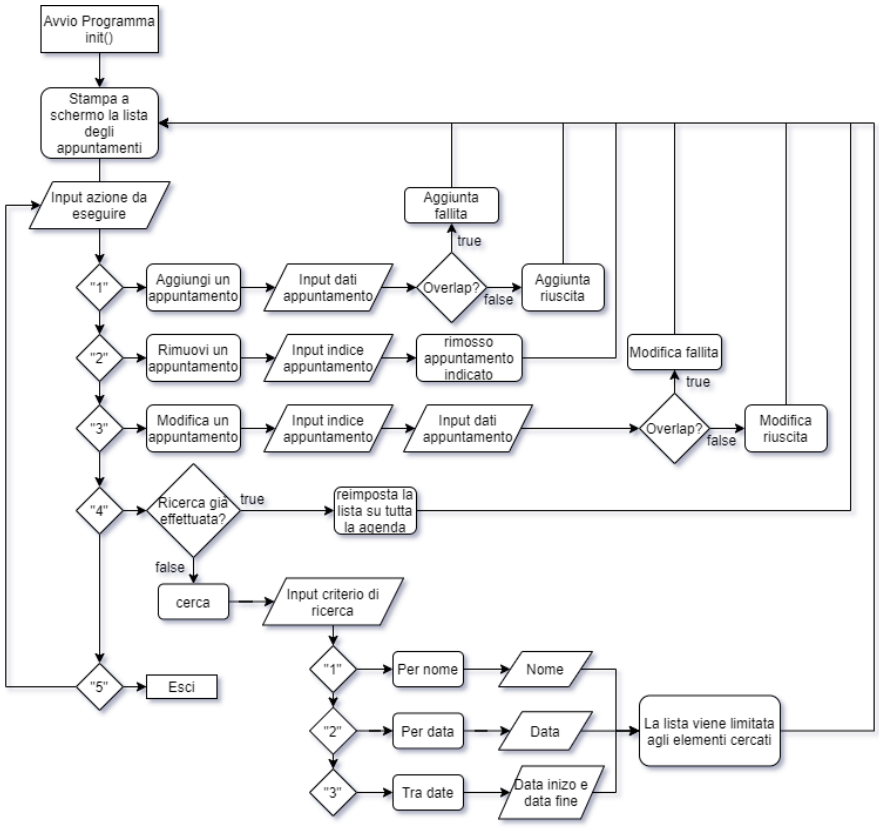

Figure 1: Mappa concettuale sull'uso.

1.1 Comando "back"

Si è scelto di implementare la possibilità di annullare completamente un operazione tramite il comando "**back**". Tale comando può essere inserito in una qualsiasi fase di input.

2 Package Progetto: Classi

2.1 Classe Appuntamento

Ogni campo della classe Appuntamento è dichiarato *private* per motivi di incapsulamento. La classe quindi fornisce metodi **getters** e **setters**, per ognuno dei campi, ad eccezione dei formatter, che non hanno metodi **set**, per evitare problemi legati alla formattazione delle date per l'interazione con file esterno.

2.2 Classe Agenda

La classe Agenda si occupa di gestire la lista degli appuntamenti.
La agenda è stata realizzata tramite ArrayList<Appuntamento>.
Come da consegna la classe fornisce la possibilità di iterare sulla lista tramite **iterator**.

2.2.1 Metodo search()

search() si occupa di fornire un metodo generico per la ricerca sulla lista. Si basa sull'implementazione di un interfaccia funzionale, dando la possibilità di usare espressioni lambda.
Si è scelto di implementare i metodi di ricerca per nome, per data e tra date, utilizzando questo meccanismo per evitare di clonare parti del codice, rendere più facile la manutenzione, ed abilitare la possibilità all'utente della classe Agenda di utilizzare diversi criteri di scelta.

2.2.2 Metodo getList()

Il metodo *getList()* ritorna una nuova copia, non shallow-copy, della lista.
I motivi per tale implementazione riferiscono al fatto che la lista, per sicurezza viene incapsulata all'interno di Agenda, e per poter agire ad esempio a livello grafico, come una stampa, si fornisce una copia della lista.

2.2.3 Metodo checkOverlap()

Due appuntamenti si sovrappongono se:

$$(Di1 > Di2 \quad and \quad Di1 < Df2) \quad or \quad (Di2 > Di1 \quad and \quad Di2 < Df1)$$

dove $Di1$ è la data di inizio primo appuntamento, $Di2$ è la data di inizio secondo appuntamento, $Df1$ è la data di fine appuntamento, cioè $Di1$ + la durata del

primo appuntamento, $Df2$ è la data di fine appuntamento, cioè $Df2 +$ la durata del secondo appuntamento.

2.3 Classe Main

La classe Main rappresenta l'interfaccia tra utente e agenda.

3 Package progetto: Eccezioni

Per la realizzazione del progetto sono state usate diverse eccezioni, alcune delle quali personalizzate. Quest'ulitme sono **OverlapException** e **UndoOperationException**.

3.1 OverlapException

L'eccezione **OverlapException**, usata nei metodi di aggiunta e modifica è stata introdotta per segnalare una sovrapposizione di appuntamenti.

3.2 UndoOperationExcaption

L'eccezione **UndoOperationException** rappresenta un meccanismo di gestione dell'alterazione del normale flusso di esecuzione. Quando l'utente richiede l'annullamento di un'operazione tramite il comando "**back**" viene sollevata l'eccezione UndoOperationException dai metodi di input. L'eccezione viene poi estesa ai metodi chiamanti dell'input, che la gestisco annullando l'esecuzione.

4 Package testers

Il package di test contiene le tre classi *JUnit* di test per le classi del progetto. Sono incluse inoltre due classi che sono le copie delle classi Agenda e Main. Queste copie differiscono dalle originali solo per i modificatori di visibilità di alcuni metodi, altrimenti non raggiungibili (poiché dichiarati *private*).