“Ingegneria del Software”
2021-2022

Docente: Prof. Angelo Furfaro

MiniCAD

Data	<05/07/2022>
Documento	Documento Finale – D3

Team Members		
Nome e Cognome	**Matricola**	**E-mail address**
Marco Macrì	**220070**	**mcrmrc01b26f112c@studenti.unical.it – macrimarco001@gmail.com**

List of Challenging/Risky Requirements or Tasks

Challenging Task	Date the task is identified	Date the challenge is resolved	Explanation on how the challenge has been managed
Familiarizzare con i mezzi a disposizione	**26/05/2022**	**26/05/2022**	**Ho esplorato il codice cercando di avere le idee più chiare sul funzionamento del back end e front end del software di partenza**
Gestione degli oggetti coesistenti (singoli e gruppi)	**26/05/2022**	**29/05/2022**	**Ho pensato a delle strutture per contenere gruppi e oggetti sfruttando il più possibile l'aliasing e delle ArrayList**
Modifica comandi esistenti da collegare alla nuova struttura	**30/05/2022**	**01/06/2022**	**I comandi esistenti funzionavano, ma non mi davano alcun feedback sulla struttura in modo da tenere traccia di ciò che accadeva tra singoli oggetti e gruppi. Ho fatto in modo di riferire i comandi alla classe che contiene le strutture**
Creazione dei nuovi comandi	**03/06/2022**	**07/06/2022**	**Servivano altri comandi, per esempio quelli dedicati ai gruppi di oggetti. Sono stati aggiunti**
Trovare un modo per interpretare i comandi da stringa	**09/06/2022**	**15/06/2022**	**Essendo più familiare con lo StringTokenizer ho usato quello al posto dello StreamTokenizer**
Vari bug e imperfezioni	**15/06/2022**	**29/06/2022**	**Sfruttando junit sono riuscito a riparare il codice e a migliorare alcuni aspetti.**

A. Stato dell'Arte

L'idea era quella di creare un piccolo cad che consente di creare e manipolare delle figure a schermo mediante comandi interpretati. Il prodotto deve soddisfare tutti i comandi della grammatica imposta dal cliente. Inoltre deve garantire la possibilità di ottenere le informazioni sugli oggetti presenti in finestra mediante delle stampe apposite.
Ogni oggetto deve essere manipolabile e identificabile mediante un suo id, come ogni gruppo. La cosa più sensata era di mantenere id univoci e diversi da oggetto ad oggetto e gruppo a gruppo. Si noti che dividendo le gestioni è possibile che un gruppo e un oggetto abbiano lo stesso id: la distinzione viene fatta specificando se ci si riferisce ad un gruppo o un oggetto. Mi rendo conto che così facendo ho leggermente cambiato una delle regole grammaticali dell'applicativo; tuttavia, ho pesato che fosse la cosa migliore da fare per rendere le cose modulari e scindere la gestione dei gruppi dagli oggetti, anche se si troveranno in una singola classe.

Ho cominciato da qualcosa di "pronto": il programma visto a lezione per la manipolazione delle figure.

Originalmente i design pattern utilizzati sono: Observer – prototype – command – flyweight.

Viene fornito un programma per il disegno di figure a schermo con controller e relativa manipolazione.

Quando noi modifichiamo l'oggetto, non modifichiamo il reale oggetto, ma notifichiamo all'osservatore come noi andiamo a vederlo. Quindi come rappresentarlo. Per esempio, Duke esiste come oggetto, ma manipolandolo attraverso i comandi, si va in realtà ad alterare la sua rappresentazione, non sé stesso.
Il punto è che si sfrutta il design pattern Observer per realizzare la relazione tra gli oggetti a livello di modello e gli oggetti a livello di vista.

Cominciando dagli oggetti grafici, essi hanno dei metodi di utility e diversi metodi per essere manipolati. Oltre a questi metodi ne troviamo alcuni per impostare o rimuovere un listener.

Il listener (come interfaccia) contiene solo un metodo che notifica che c'è stato un cambiamento, si verifica quindi un evento.

L'evento (grafico) deve avere un riferimento all'oggetto che lo ha causato.

Siccome sarebbe troppo laborioso e poco pratico, stabiliamo una politica di rappresentazione per tutti gli oggetti mediante la classe astratta AbstractGraphicObject. La classe implementa GraphicObject e Cloneable (utile per sfruttare il design pattern prototype).
Essa ha una lista di listener che notifica tramite apposito metodo protetto.
Abbiamo poi il metodo clone(), che se non implementato correttamente potrebbe portare a degli errori, per evitarne, non vogliamo che l'oggetto, risultato dalla clonazione abbia gli stessi listener, ci guardiamo da ciò inizializzando il suo riferimento ad una nuova lista di listeners, inizialmente vuota.

Una volta definita la politica, possiamo passare alle classi concrete. Un esempio è il CircleObject che estende AbstractGraphicObject.
Esso ha come variabili d'istanza il suo raggio e un punto nello spazio bidimensionale.
Tra i metodi mutatori abbiamo che essi, non appena eseguono una modifica, notificano i listener.
I metodi di clonazione anche qui devono essere implementati in modo responsabile, per rimanere sempre in classe dell'oggetto cerchio, lo cloniamo avendo l'accortezza di non creare aliasing tra le coordinate della figura. Se così non fosse, la figura si sposterebbe allo spostarsi dell'originale e viceversa. È da notare che la classe non sa nulla di ciò che avviene a livello grafico.

Lo storico delle azioni eseguite e l'esecuzione delle istruzioni sono dati dall'adozione del design pattern command. La richiesta è quindi incapsulata in un oggetto che può essere immesso in una lista e di cui si può tenere traccia. In questo programma, il concetto è implementato dall'HistoryCommandHandler e dall'interfaccia (e concretizzazioni della stessa) Command. Il modo in cui è implementata la politica è indipendente dal contesto in cui lo stiamo utilizzando, potrebbe quindi essere preso così com'è e riutilizzato in un altro.
I comandi disponibili inizialmente sono tre e sono specifici del nostro contesto iniziale.

Abbiamo un altro tipo di listener, il GraphicObjectPanel che estende JComponent (parte di Swing). Esso mantiene tramite una lista i riferimenti ai modelli degli oggetti che dovranno essere visualizzati (nella nostra istanza conosciamo già i modelli).
Il metodo che riceve le notifiche è graphicChanged(…) che riceve come parametro un evento grafico; di seguito al trigger di un evento, il metodo ridisegna il pannello. Per farlo sovrascrive il metodo PaintComoponent(Graphics g). Questo metodo, dopo aver eseguito il corrispondente della superclasse esegue il pattern flyweight: ottiene un GraphicObjectView, che è un flyweight e chiede alla vista di disegnarsi. Si serve di una hash map per associare oggetti e viste.
Il metodo drawGraphicObjectView(…) dell'interfaccia GraphicObjectView riceve come parametri lo stato estrinseco dell'oggetto e il contesto in cui disegnarlo. Le realizzazioni dell'interfaccia si occupano del disegno degli specifici oggetti grafici.
La mappa serve per "installare" la visione (preparare il pannello ad accettare determinate figure), che è importante per la creazione della stessa.

Altri metodi presenti sono add e remove: uno aggiunge gli oggetti al pannello, si passa come listener e forza il ridisegno del pannello, l'altro anziché aggiungerlo, lo rimuove, facendo le stesse altre operazioni.

Il metodo drawGraphicObjectView(…) dell'interfaccia GraphicObjectView riceve come parametri lo stato estrinseco dell'oggetto e il contesto in cui disegnarlo. Le realizzazioni dell'interfaccia si occupano del disegno degli specifici oggetti grafici.

B. Raffinamento dei Requisiti

B.1 Servizi (con prioritizzazione)

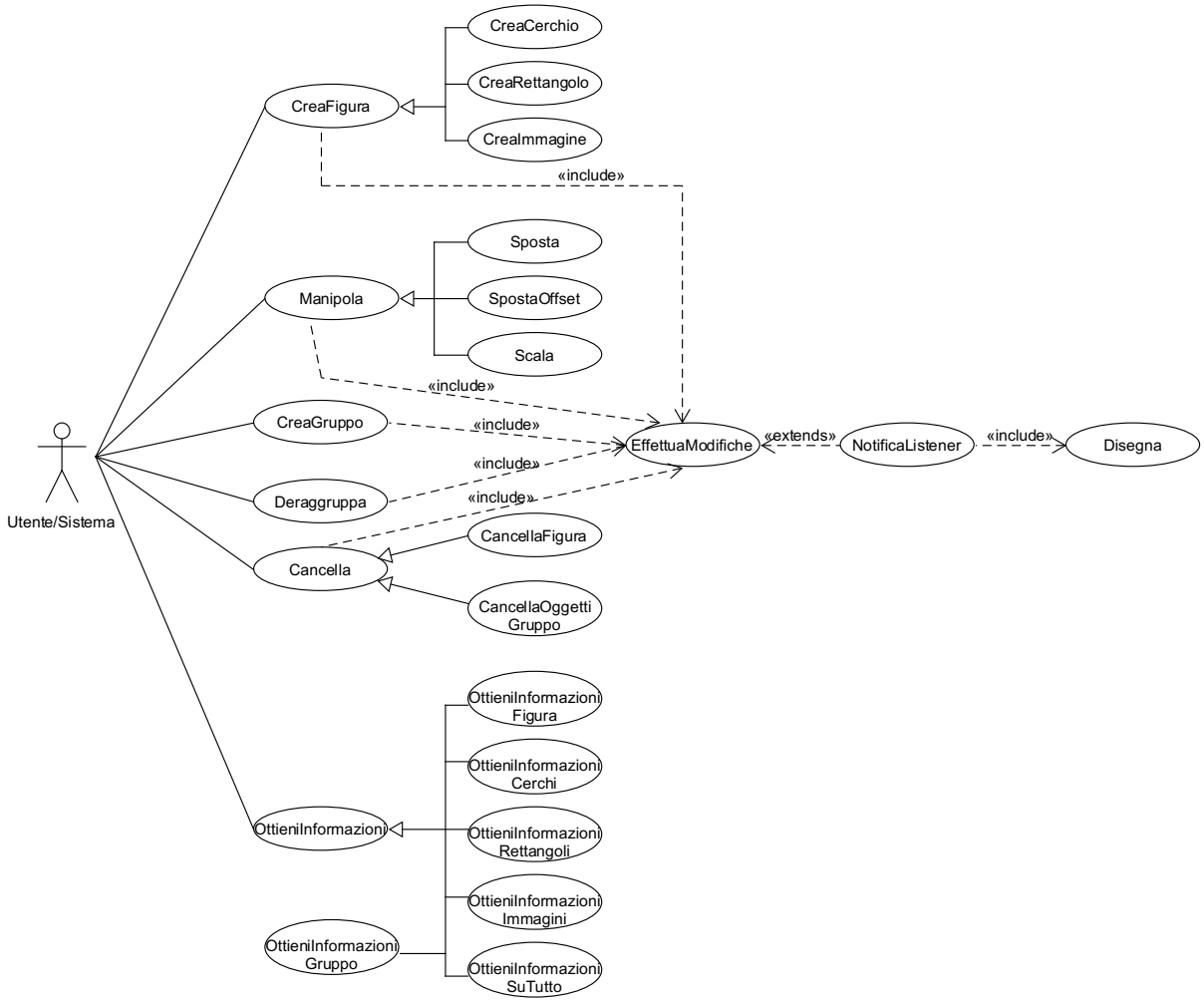
Il sistema offre la possibilità di creare e manipolare delle figure a schermo mediante comandi utente.
È possibile creare, eliminare, raggruppare, deraggruppare, ridimensionare, spostare gli oggetti. Tutte le azioni, tranne quelle per raccogliere informazioni su oggetti e gruppi, sono reversibili. L'organizzazione mediante identificativi univoci di gruppi e figure offre un grado d'interazione elevato e rende l'utilizzo del software semplice ed intuitivo.
Il tutto si verifica all'interno di un workspace di dimensioni impostabili comodamente all'avvio del sistema.

B.2 Requisiti non Funzionali

Il sistema offre un modo semplice ed intuitivo per compiere le azioni di cui è capace.
L'efficienza è tenuta molto in considerazione, infatti molte scelte progettuali sono volte al risparmio delle risorse, tuttavia, a causa di doverose implementazioni, risulta inevitabile dover scendere a piccoli compromessi. Nonostante tutto, il sistema gode di buona responsività e compattezza.
Non sono necessari dei super computers per eseguirlo, è sufficiente un computer con una java virtual machine. La consegna viene effettuata mediante pacchetto con codice sorgente, tuttavia per non tediare l'utente con tutto ciò che avviene dietro le quinte, viene offerta una classe launcher che lancia l'applicativo e in caso si voglia integrare con altri sistemi sono presenti diversi modi di farlo, per un uso molto semplice si può usare la facciata del minicad, altrimenti inizializzare il software e interagire mediante controlli dell'editor manualmente.

B.3 Scenari d'uso dettagliati

I casi d'uso del software sono molteplici e riguardano la creazione di un oggetto, la sua eliminazione, il suo spostamento, la modifica del suo fattore di scala. Tali interazioni possono essere svolte anche con i gruppi di oggetti. Inoltre, è possibile ottenere delle informazioni sugli oggetti e gruppi a schermo.



(via UMLet)

Caso d'uso	*CreaFigura*
Tipo	*Primario*
Precondizione	*La figura non è nella finestra né in memoria*
Svolgimento normale	*La figura viene creata e poi aggiunta alla finestra.*
Svolgimento alternativo	
Post condizione	*La figura è creata e poi aggiunta alla finestra*
Descrizione	*Serve a creare la figura che l'utente intende porre sullo schermo. È un'azione reversibile.*

Caso d'uso	*CreaCerchio*
Tipo	*primario*
Precondizione	*Il cerchio non appare a schermo*
Svolgimento normale	*Di seguito all'inoltro del comando il cerchio è creato e poi aggiunto a schermo.*
Svolgimento alternativo	
Post condizione	*Il cerchio appare a schermo*
Descrizione	*Il comando serve a creare un cerchio date le informazioni in input e a porlo sullo schermo. È un'azione reversibile.*

Caso d'uso	*CreaRettangolo*
Tipo	*primario*
Precondizione	*Il rettangolo non appare a schermo*
Svolgimento normale	*Di seguito all'inoltro del comando il rettangolo è creato e poi aggiunto a schermo.*
Svolgimento alternativo	
Post condizione	*Il rettangolo appare a schermo*
Descrizione	*Il comando serve a creare un rettngolo date le informazioni in input e a porlo sullo schermo. È un'azione reversibile.*

Caso d'uso	*CreaImmagine*
Tipo	*primario*
Precondizione	*L'immagine non appare a schermo*
Svolgimento normale	*Di seguito all'inoltro del comando l'immagine è creata e poi aggiunta a schermo.*
Svolgimento alternativo	
Post condizione	*L'immagine appare a schermo*
Descrizione	*Il comando serve a creare un'immagina date le informazioni in input e a porla sullo schermo. È un'azione reversibile.*

Caso d'uso	*Manipola*
Tipo	*Primario*
Precondizione	*La modifica da fare deve essere fatta su un oggetto o un gruppo di oggetti. Prima del comando non ci sono mutazioni*
Svolgimento normale	*La modifica viene fatta all'oggetto o agli elementi del gruppo*
Svolgimento alternativo	*La modifica viene fatta agli elementi del gruppo*
Post condizione	*Le modifiche sono state fatte e visualizzate a schermo*
Descrizione	*Serve a manipolare la figura o il gruppo di riferimento. È un'azione reversibile.*

Caso d'uso	*Sposta*
Tipo	*Primario*
Precondizione	*La figura o gli oggetti del gruppo si trovano in una posizione originaria*
Svolgimento normale	*La figura si sposta nelle coordinate assolute date in input*
Svolgimento alternativo	*gli oggetti del gruppo si spostano nelle coordinate assolute date in input*
Post condizione	*La figura o gli oggetti del gruppo si trovano nelle coordinate assolute date in input*

Descrizione	*Serve a spostare nelle coordinate assolute la figura o il gruppo scelto. L'azione è reversibile*

Caso d'uso	*SpostaOffset*
Tipo	*Primario*
Precondizione	*La figura o gli oggetti del gruppo si trovano in una posizione originaria*
Svolgimento normale	*La figura si sposta nelle coordinate date dalla somma di quelle originarie e di quelle date in input*
Svolgimento alternativo	*gli oggetti del gruppo si spostano nelle coordinate date dalla somma di quelle originarie e di quelle date in input*
Post condizione	*La figura o gli oggetti del gruppo si trovano nelle coordinate date dalla somma di quelle originarie e di quelle date in input*
Descrizione	*Serve a spostare nelle coordinate date dalla somma di quelle originarie e di quelle date in input la figura o il gruppo scelto. L'azione è reversibile*

Caso d'uso	*scala*
Tipo	*Primario*
Precondizione	*La figura o gli oggetti di un gruppo hanno un fattore di scala originario*

Svolgimento normale	*Il fattore di scala della figura viene mutato, ingrandendo o rimpicciolendo la visione della figura o degli oggetti di un gruppo*
Svolgimento alternativo	*Il fattore di scala degli oggetti del gruppo viene mutato, ingrandendo o rimpicciolendo la visione delle figure*
Post condizione	*La figura o gli oggetti di un gruppo appaiono più grandi o più piccoli*
Descrizione	*Serve a modificare la grandezza delle figure selezionate. L'azione è reversibile*

Caso d'uso	*CreaGruppo*
Tipo	*Primario*
Precondizione	*Gli oggetti sono indipendentemente manipolabili*
Svolgimento normale	*Gli oggetti vengono raggruppati sotto un identificativo di gruppo*
Svolgimento alternativo	
Post condizione	*Gli oggetti sono adesso relazionati tra loro*
Descrizione	*Serve a raggruppare degli oggetti sotto un identificativo di gruppo. Gli oggetti possono anche essere modificati individualmente mediante id specifico dell'oggetto. L'azione è reversibile.*

Caso d'uso	*Deraggruppa*
Tipo	*Primario*
Precondizione	*Gli oggetti appartengono ad un gruppo*

Svolgimento normale	*Gli oggetti vengono separati*
Svolgimento alternativo	
Post condizione	*Gli oggetti non sono più relazionati tra loro*
Descrizione	*Serve a deraggruppare degli oggetti. Gli oggetti possono essere solo modificati individualmente mediante id specifico dell'oggetto. L'azione è reversibile.*

Caso d'uso	*Cancella*
Tipo	*Primario*
Precondizione	*Gli oggetti o tutti gli oggetti del gruppo sono presenti a schermo*
Svolgimento normale	*Gli oggetti tutti gli oggetti del gruppo verranno cancellati e non più visualizzati*
Svolgimento alternativo	*Tutti gli oggetti del gruppo verranno cancellati e non più visualizzati*
Post condizione	*Gli oggetti o tutti gli oggetti del gruppo selezionato non verranno visualizzati.*
Descrizione	*Serve a cancellare degli oggetti o un gruppo di oggetti. L'azione è reversibile.*

Caso d'uso	*CancellaFigura*
Tipo	*Primario*
Precondizione	*L'oggetto è presente a schermo*

Svolgimento normale	*L'oggetto viene cancellato e non più visualizzato*
Svolgimento alternativo	
Post condizione	*L'oggetto non verrà visualizzato e non sarà in struttura dati*
Descrizione	*Serve a cancellare un oggetto. L'azione è reversibile.*

Caso d'uso	*CancellaGruppo*
Tipo	*Primario*
Precondizione	*Gli oggetti del gruppo sono presenti a schermo*
Svolgimento normale	*Gli oggetti del gruppo sono presenti a schermo vengono cancellati e non più visualizzati*
Svolgimento alternativo	
Post condizione	*Il gruppo non verrà visualizzato e non sarà in struttura dati*
Descrizione	*Serve a cancellare un gruppo di oggetti. L'azione è reversibile.*

Caso d'uso	*OttienInformazioni*
Tipo	*Primario*
Precondizione	*Gli oggetti del gruppo sono presenti a schermo ed esistono in struttura dati*
Svolgimento normale	*Il software stampa a schermo le informazioni richieste*

Svolgimento alternativo	
Post condizione	*Sono visualizzate le informazioni desiderate*
Descrizione	*Serve a richiedere delle informazioni riguardo agli oggetti a schermo o gruppi. L'azione non è reversibile in quanto non muta gli oggetti.*

Caso d'uso	*OttienInformazioniFigura*
Tipo	*Primario*
Precondizione	*La figura esiste in struttura dati ed è visualizzata.*
Svolgimento normale	*Il software stampa a schermo le informazioni riguardanti la figura*
Svolgimento alternativo	
Post condizione	*Sono visualizzate le informazioni desiderate*
Descrizione	*Serve a richiedere delle informazioni riguardo un oggetto a schermo. L'azione non è reversibile in quanto non muta gli oggetti.*

Caso d'uso	*OttienInformazioniCerchi*
Tipo	*Primario*
Precondizione	*Le figure Cerchio esistono in struttura dati e sono visualizzate.*
Svolgimento normale	*Il software stampa a schermo le informazioni riguardanti le figure di tipo cerchio*

Svolgimento alternativo	
Post condizione	*Sono visualizzate le informazioni desiderate*
Descrizione	*Serve a richiedere delle informazioni riguardo oggetti a schermo di tipo Cerchio. L'azione non è reversibile in quanto non muta gli oggetti.*

Caso d'uso	*OttienInformazioniRettangoli*
Tipo	*Primario*
Precondizione	*Le figure Rettangolo esistono in struttura dati e sono visualizzate.*
Svolgimento normale	*Il software stampa a schermo le informazioni riguardanti le figure di tipo rettangolo*
Svolgimento alternativo	
Post condizione	*Sono visualizzate le informazioni desiderate*
Descrizione	*Serve a richiedere delle informazioni riguardo oggetti a schermo di tipo Rettangolo. L'azione non è reversibile in quanto non muta gli oggetti.*

Caso d'uso	*OttienInformazioniImmagini*
Tipo	*Primario*
Precondizione	*Le figure Immagine esistono in struttura dati e sono visualizzate.*

Svolgimento normale	*Il software stampa a schermo le informazioni riguardanti le figure di tipo Immagine*
Svolgimento alternativo	
Post condizione	*Sono visualizzate le informazioni desiderate*
Descrizione	*Serve a richiedere delle informazioni riguardo oggetti a schermo di tipo Immegine. L'azione non è reversibile in quanto non muta gli oggetti.*

Caso d'uso	*OttienInformazionGruppi*
Tipo	*Primario*
Precondizione	*I gruppi o il gruppo esistono in struttura dati e sono visualizzati.*
Svolgimento normale	*Il software stampa a schermo le informazioni riguardanti il o il gruppo*
Svolgimento alternativo	
Post condizione	*Sono visualizzate le informazioni desiderate*
Descrizione	*Serve a richiedere delle informazioni riguardo un gruppo specifico o tutti i gruppi. L'azione non è reversibile in quanto non muta gli oggetti.*

Caso d'uso	*OttienInformazioniSuTutto*
Tipo	*Primario*

Precondizione	*Le figure e gruppi esistono in struttura dati e sono visualizzate.*
Svolgimento normale	*Il software stampa a schermo le informazioni riguardanti le figure tutti gli oggetti e gruppi*
Svolgimento alternativo	
Post condizione	*Sono visualizzate le informazioni desiderate*
Descrizione	*Serve a richiedere delle informazioni riguardo a tutti gli oggetti e gruppi. L'azione non è reversibile in quanto non muta gli oggetti.*

Caso d'uso	*EffettuaModifiche*
Tipo	*Primario*
Precondizione	*Le figure i gruppi esistono in struttura dati e sono visualizzate.*
Svolgimento normale	*Il software muta le figure in struttura dati*
Svolgimento alternativo	
Post condizione	*Le figure in struttura sono mutate*
Descrizione	*Serve ad effettuare le modifiche desiderate*

Caso d'uso	*NotificaListeners*
Tipo	*secondario*
Precondizione	

Svolgimento normale	*I listeners si accorgono dei cambiamenti nelle figure*
Svolgimento alternativo	
Post condizione	*Le figure in struttura sono mutate*
Descrizione	*I listeners si devono accorgere delle modifiche fatte per poi far disegnare le modifiche apportate*

Caso d'uso	*Disegna*
Tipo	*secondario*
Precondizione	*Le figure sono state mutate*
Svolgimento normale	*La visione a schermo degli oggetti viene mutata.*
Svolgimento alternativo	
Post condizione	*Le figure in struttura sono mutate*
Descrizione	*Dopo che i listeners si accorgono delle modifiche, esse vengono rappresentate in finestra.*

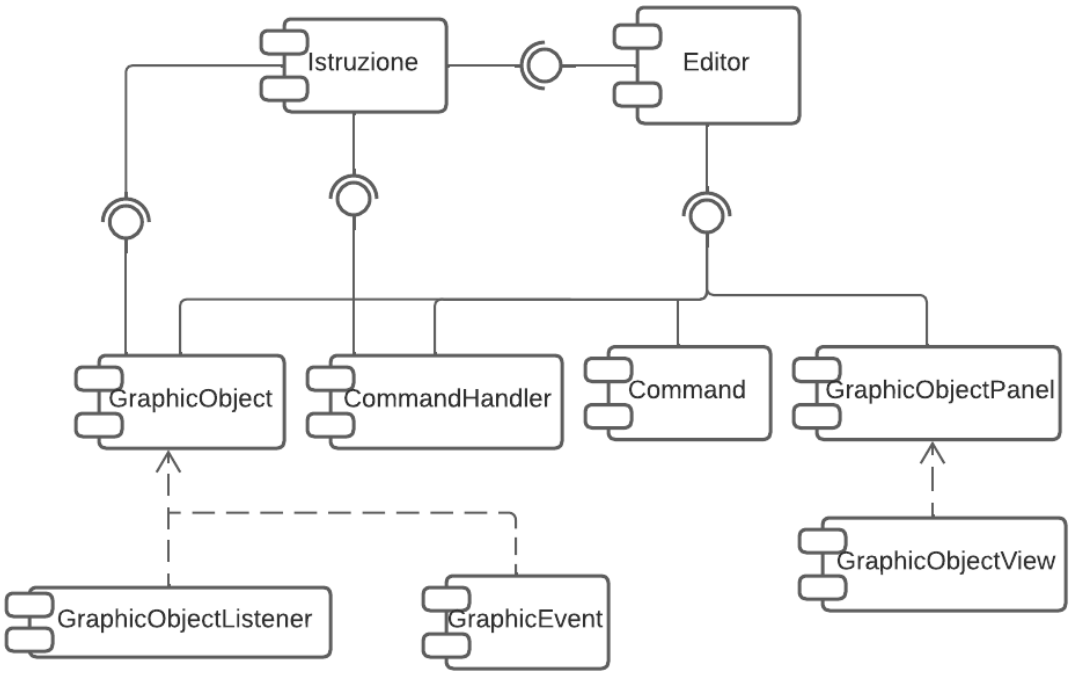
A.5 Assunzioni

Le assunzioni principali che ho fatto sono sulla gestione dei gruppi e degli oggetti, visto che nella richiesta del cliente si faceva in modo vago. Ho deciso quindi di separare oggetti e gruppi mediante i prefissi "id" (oggetto) e "g" (gruppo) più id della selezione che si intende effettuare.

Un'altra assunzione riguarda lo spostamento del gruppo senza offset: ho inteso che si volesse spostare ogni oggetto del gruppo nelle coordinate date.
La cancellazione del gruppo inoltre prevede la cancellazione di tutti gli oggetti anziché l'ungroup.
C'è da dire che non viene fornita alcuna indicazione sulle dimensioni della finestra; perciò, il sistema dovrà occuparsi di chiedere le dimensioni desiderate all'utente… è previsto che questa sia un'opzione visto che sarà possibile impostarle anche a priori. È assunto che per poter implementare undo e redo sia necessario aggiungere dei comani omonimi.

C. Architettura Software

C.1 The static view of the system: Component Diagram



(via LucidChart)

Questi sono i componenti che lavorano all'interno del MiniCAD.

C.2 The dynamic view of the software architecture: Sequence Diagram

Anche se non sembra, il progetto ha un gran numero di azioni al suo interno… avvengono moltissime interazioni tra gli oggetti presenti… così tante che il sequence diagram risulta essere molto grande da essere contenuto su questo documento word: a tal proposito lascio il collegamento al file presente sul mio Google drive.
https://drive.google.com/file/d/1zaUZR7xIlBfCIQvsv8rDnoau8AMNqqG4/view
(via SequenceDiagram by VanStudio)

il file in questione mostra in modo "unfold" tutte le azioni che avvengono dall'esecuzione del Launcher… si consideri che rappresenta quasi ogni tipo d'interazione ed è sprovvisto della visualizzazione dei cicli e delle scelte alternative proprio per ricoprire tutti gli scenari possibili. Le scelte di norma vengono fatte nell'editor visto che alcuni metodi hanno un parametro booleano che funge da selettore di gruppo. Inoltre è presente un ciclo che accoglie comandi fino alla chiusura della finestra del MiniCAD. Un'altra forma di scelta è nel comando stesso che per essere eseguito deve essere diviso in token e interpretato.

D. Scelte Progettuali (Design Decisions)

La scelta principale su come gestire ed organizzare le figure in singolo e gruppi in modo semplice è quella di creare una classe che si occupi di svolgere tutte le azioni desiderate grazie a delle strutture dati (arraylist) che contengono gli oggetti grafici e i gruppi di oggetti grafici.
La scelta delle arraylist non è casuale, infatti sono una struttura dati che offre la "presa" di un elemento ad un dato indice con poco tempo e usare gli stessi indici come id di figure e gruppi è molto conveniente.
Gli id delle figure cancellate o gruppi, diventano poi null per evitare ulteriori slittamnenti tra gli id successivi. Qualora dovesse essere creato un nuovo gruppo o figura, se trovasse uno spazio che contiene null, lo occuperebbe permettendo il rutilizzo dell'id.
Siccome la parola chiave in applicazioni del genere è semplicità, ho deciso di continuare sulla strada dell'aggiunta di comandi. Abbiamo l'utilizzo del design pattern Command. Si è deciso di lasciare ad esso la gestione di comandi che hanno effetto visivo o di raggruppamento poiché vanno a modificare lo stato dell'editor. La richiesta di

informazioni è gestita nella classe editor e lavora direttamente sulle strutture per questioni di ordine tra classi.
Bisogna che l'utente finale non abbia problemi nel gestire tutte le dipendenze del cad, a tale motivo è di grande aiuto la facciata del minicad. Avendo pochi metodi a disposizione riesce a preparare con poche righe l'inizializzazione del miniCAD ed a guidare l'utente con un CLI (command line interface) semplice da utilizzare.

E. Progettazione di Basso Livello

Come detto nello stato dell'arte, si comincia dal progetto visto a lezione. Il primo probema è stato quello di capire come organizzare gruppi e oggetti singili. Ho creato un editor che contiene delle arraylist: una contiene gli oggetti grafici, l'altra un'altra arraylist cche contiene alcune istnze degli oggetti nella prima arraylist. In questo caso c'è un utilizzo voluto dell'aliasing. È stato essenziale condividere i puntatori agli oggetti esistenti. I metodi che vanno effettivamente ad alterare lo stato degi oggetti, manipolazione, grouping, ungrouping, sfruttano dei comandi creati appositamente per interfacciarsi alle strutture dati. Vediamo come esempio la creazione di una figura: se trova un indice che ha null come oggetto ci piazza l'oggetto grafico ed esegue il comando di creazione. Verifico l'esistenza di un null vicino per non creare degli spazzi occupati da nulla… quindi è possibile riciclare id. in modo leggermente analogo vado a creare i gruppi sulla struttura dei gruppi. La cancellazione di oggetti pone semplicemente a null l'elemento nell'id dato in input. Se in questione si ha un gruppo si fa lo stesso nella struttura dedicata ai gruppi. Essenziamente i concetti fondamentali di come ho programmato questi comportamenti sono nella creazione e cancellazione, grouping e ungrouping, cosa piuttosto difficile. Una volta implementati questi comportamenti è facile passare alle manipolazioni leggere, quali spostamento, scala…
La richiesta delle informazioni avviene direttamente nell'editor. È infatti più facile navigare le figure direttamente nelle strutture che le contengono piuttosto che creare altri comandi, che allungherebbero la gerarchia dele classi.
La fase di riconoscimento comandi si avvale di un semplice string tokenizer che riconosce tutti i comandi possibili e si avvale dell'editor per eseguirli.

I test pensati e realizzati in junit si occupano di verificare che ogni comando funzioni come aspettato. Questo implica che ci siano dei controlli fatti a basso livello sulle figure memorizzate nella struttura all'interno dell'editor e nei loro attributi.

F. Spiegare come il progetto soddisfa i requisiti funzionali (FRs) e quelli non funzionali (NFRs)

Il sistema rispetta le richieste emerse dalla specifica quali:

- creazione di figure con la restituzione dell'id figura
- cancellazione della figura o del gruppo di figure
- spostamento di figure o gruppi di figure con e senza offset
- modifica del grado di scala della figura o del gruppo di figure
- richiesta di informazioni su gruppi (singoli e tutti), figure specifiche, tipi di figure particolari o tutte le figure, area e perimetri delle figure.
- Il raggruppamento di figure sotto un unico id
- Il la "dispersione" delle figure dal gruppo.
- È presente la possibilità di fare gli undo e redo dei comandi che alterano lo stato della struttura delle figure.

Il sistema è inoltre pratico, efficiente e facile da usare… inoltre è portabile. È possibile usare MiniCAD ovunque ci sia compatibilità con OpenJDK 18 o JDK 18. È molto probabile funzioni con versioni precedenti, purtroppo non si è testato. Lo spazio occupato in memoria è basso, dato l'utilizzo di flyweight.

Appendix. Prototype

La consegna consiste in un pacchetto is che contiene il pacchetto minicad. In esso è possibile trovare la classe Launcher che si occupa di far partire il software… è possibile lavorare manualmente con tutti i componenti dei pacchetti. È permesso svolgere tutte le attività richieste dal cliente in modo semplice ed intuitivo tramite la gui (guided user interface) espressa in cl.