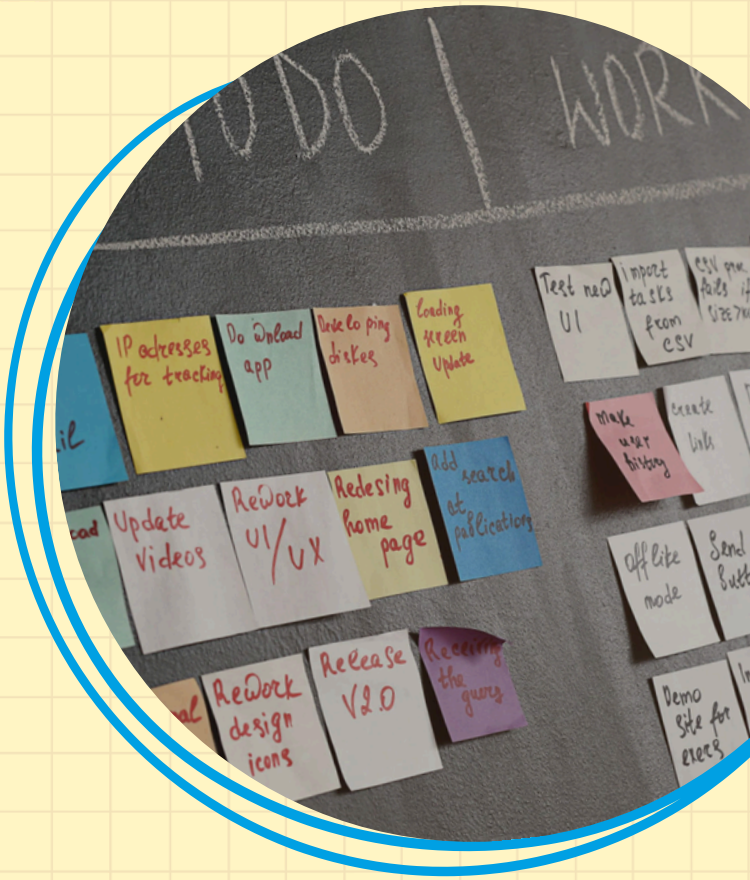
A GEOMETRIC DRAWING PROGRAM

A.A. 2024/2025

Scelte di Design



Docente:

Pierluigi Ritrovato

A cura del gruppo 5:

Apicella Antonio 0622702531

Celano Benedetta Pia 0622702558

Cuomo Carmine 0622702688

Guerra Simone 0622702675

Scelte di Design

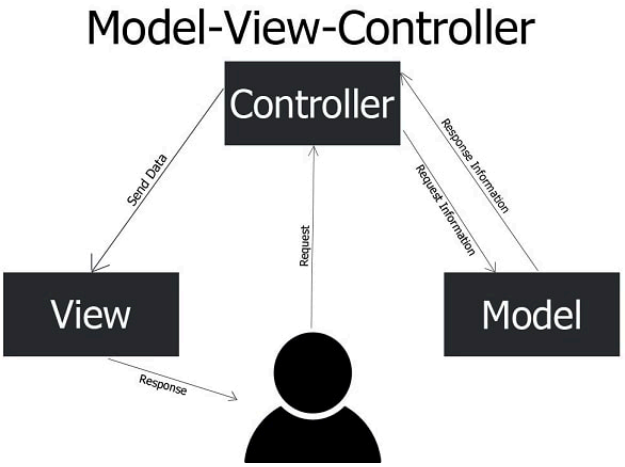
High-level design

Considerazioni sul pattern adottato

Al fine di avere un'idea sull'architettura finale del sistema assegnato - che aiuti ad immaginare come sono organizzate e distribuite le componenti del sistema - e consentire al team di sviluppo di seguire un unico scope si è deciso di investire parte delle energie nell'individuazione, fin dalla fase di PreGame, di quale fosse il pattern architetturale più opportuno.

La scelta è ricaduta sul pattern **MVC (Model View Controller)** per le seguenti motivazioni:

- MVC permette una maggiore manutenibilità del codice grazie alla separazione logica tra dati e visualizzazione;
- la separazione netta tra Model, View e Controller favorisce l'estensibilità senza compromettere l'efficienza complessiva del codice;
- l'attività di testing è agevolata: ogni componente è testabile in maniera indipendente dalle altre;
- si integra perfettamente con il framework JavaFX che si intende utilizzare per l'implementazione;
- è possibile creare diverse View per lo stesso Model ad esempio per lo sviluppo di un'eventuale versione mobile che il cliente potrebbe richiedere in futuri rilasci.



Nello specifico, il team ha ipotizzato che:

- il **Model** memorizzi le forme geometriche create dall'utente con le proprietà ad esse associate (colore di contorno, di riempimento ...);

- la **View** si occupi della rappresentazione a schermo delle figure e del layout di bottoni, spinner e degli altri elementi UI;
- il **Controller** riceva tramite GUI gli eventi dell'utente (click del mouse, drag), interpreti l'azione e richiami i metodi del Model, aggiornando la View in risposta ai cambiamenti del Model.

Considerazione sulle tattiche architetturali

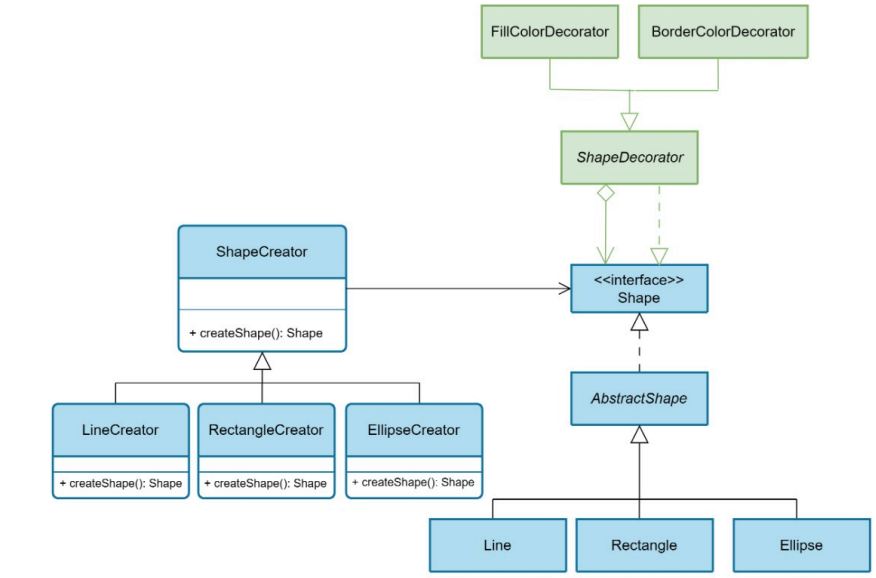
Vista la natura del progetto, si rende noto che non sono presenti problematiche risolvibili mediante tattiche architetturali note agli sviluppatori (ping/echo, heartbeat...).

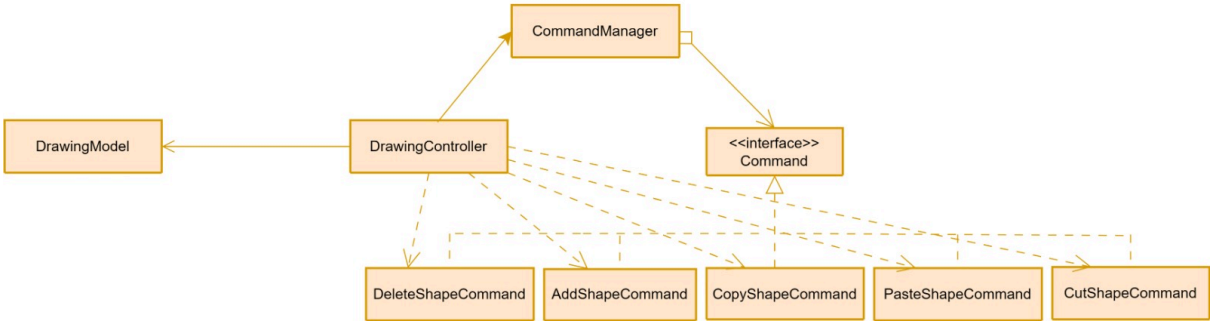
Detailed Design

A seguito di una revisione iniziale delle user stories contenute nel primo sprint e per guidare la definizione delle tasks di queste, il team ha determinato tre pattern utilizzabili, vale a dire:

- **Factory Method** come pattern creazionale per incapsulare la creazione degli oggetti, delegando a sottoclassi la scelta della classe concreta da istanziare (e dunque quale forma geometrica);
- **Decorator** come pattern strutturale per aggiungere dinamicamente responsabilità a un oggetto (il colore di riempimento/contorno), senza alterarne la classe di base;
- **Command** come pattern comportamentale per incapsulare ciascuna richiesta come oggetto, separando chi la invoca dalla logica e permettendo di tenere traccia dei comandi eseguiti, annullarli etc...;

Si noti l'intenzione del team di *guardare avanti* vista la presenza nello scheletro di Class Diagram di seguito riportato anche di classi per il taglio, la copia e l'incollaggio di figure, funzionalità da non implementare nel primo sprint.





Il team si riserva la possibilità di modificare e/o migliorare il class diagram presentato, aggiungendo attributi e metodi ove ritenuti necessari, durante il primo sprint. Eventuali aggiornamenti saranno dunque visualizzabili in un opportuno file nella directory dedicata su Git con annessa la motivazione della modifica.