DesignPatterns: Paint

Klas: HBO-ICT SE 2B
Project/Opdracht: Design Patterns - tekenprogramma
Auteurs: Damiaen Toussaint (4554752), Mail:damiaen.toussaint@student.nhlstenden.com)
Mathijs Grafhorst (4568540),Mail: mathijs.grafhorst@student.nhlstenden.com)
Datum laatste wijziging: 31-01-2020

Datum	Beschrijving van wijziging	Versie Document
27-05-2020	Afronding versie 1.0 document	1.0

Inhoudsopgave

1. Inleiding

Dit document betreft de documentatie van de opdracht voor het vak “designPatterns”. Dit document is opgesteld door Damiaen Toussaint en Mathijs Grafhorst. In dit document zal er iteratief bijgehouden worden wat voor wijzigingen wij aan de code doorvoeren.

Om bij te kunnen houden welke wijzigingen er allemaal zijn doorgevoerd is het handig om deze in een document te noteren. Dit zal gaan volgens de 6 stappen die zijn aangegeven in de projectomschrijving.

Deze stappen zijn als volgt:

- Stap 1: eenvoudig tekenprogramma
- Stap 2: command pattern
- Stap 3: composite pattern
- Stap 4: visitor pattern
- Stap 5: strategy pattern en singleton pattern
- Stap 6: decorator pattern

In dit document zullen we bij elke uitgevoerde stap toelichten wat wij hebben aangepast aan het project en waarom wij bepaalde keuzes hebben gemaakt.

Tijdens de ontwikkeling van dit project zullen we gebruikmaken van git, en zal de code in een private repository op github komen te staan. Omdat er per stap broncode beschikbaar moet zijn hebben we besloten om aan het eind van een iteratie/stap de code naar een aparte branch te pushen. Zo kunnen we onze code history goed in 1 repository bewaren en hoeven we niet te prutsen met losse bestanden en zipjes.

2. Ontwikkelingsomgeving

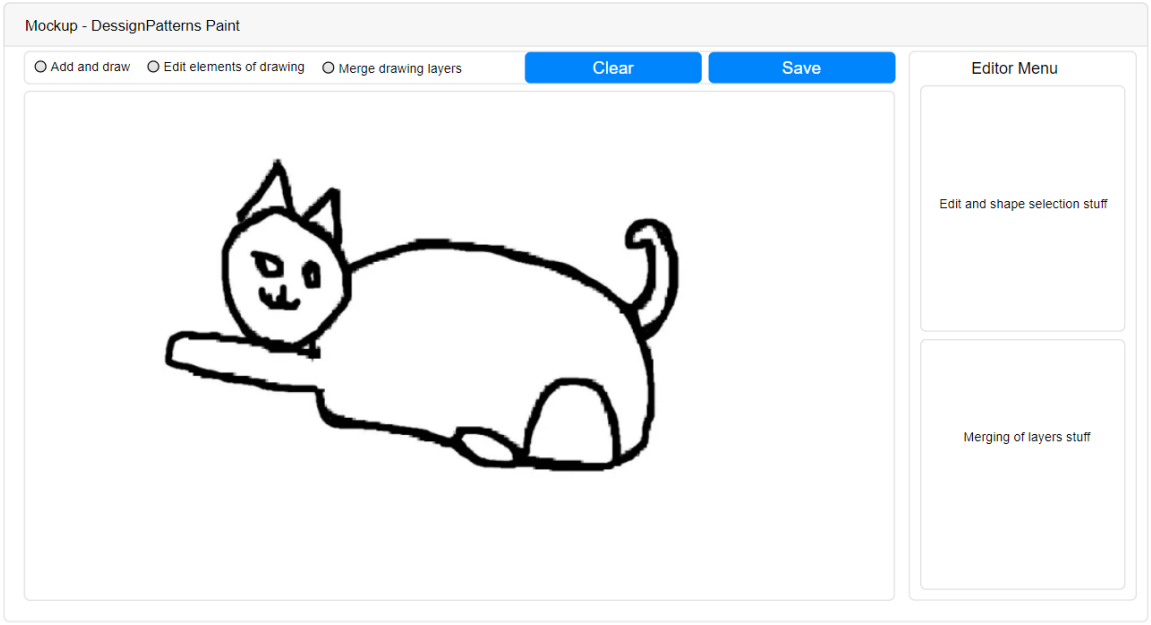
Hieronder is een kleine toelichting te vinden van de eerste keuzes die we als team hebben gemaakt, voordat we zijn begonnen aan de ontwikkeling van het DesignPatterns project. Het zal hier gaan over de gekozen programmeertaal, frameworks die we in het begin gebruiken en de hoe wij de userinterface voor ons zien.

2.1. Gekozen taal en Frameworks

Voordat wij een start maken aan de ontwikkeling van het programma hebben we als groepje even overleg gehad over onze keuze in programmeertaal. Uit dit overleg kwam naar boven dat de voorkeur van beide teamleden naar **Java** uitgaat.

Omdat het project van Periode 3 Data Science ook in Java is geschreven, kunnen we bepaalde onderdelen overnemen en zitten we al een beetje in de Java sfeer. Het Data Science project maakt gebruik van **Java Swing** in combinatie met de UIBuilder van IntelIJ. Deze manier van opbouwen van de user interface zullen wij ook in het DesignPatterns project gebruiken.

2.2. Mockup van de user interface



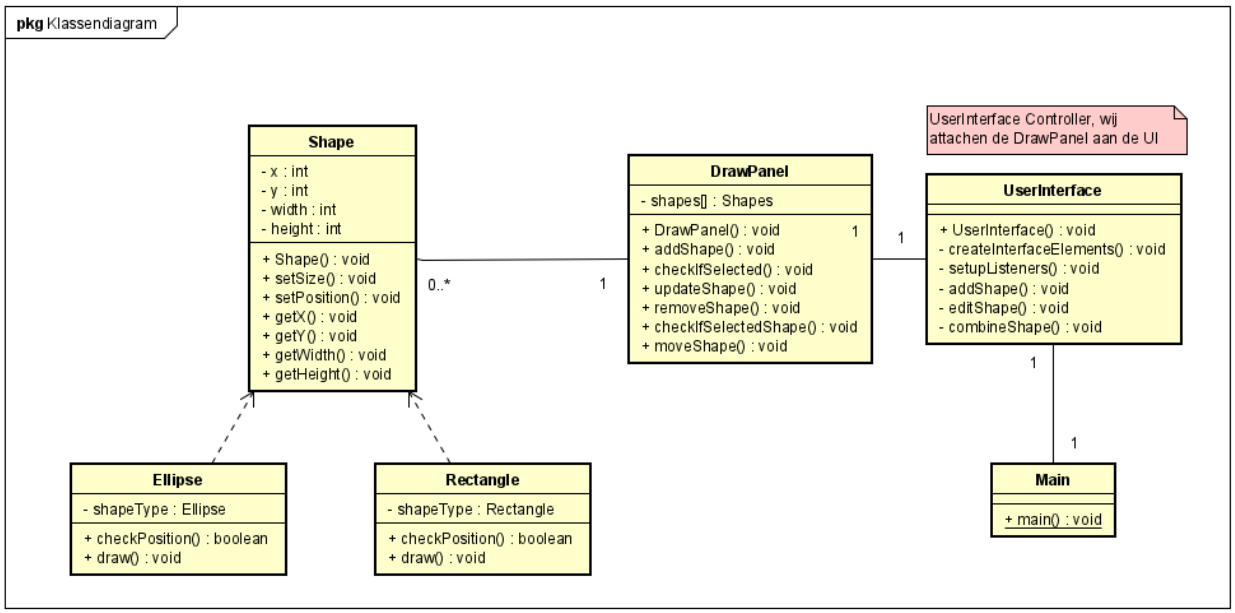
Figuur 1 - userinterface eerste versie, deze is zo gebouwd dat we aan elke kant nog wat kunnen aanpassen

3. Stap 1: eenvoudig tekenprogramma

Bij stap 1 zullen we de eerste versie van het tekenprogramma realiseren. We zullen in de eerste versie van de applicatie al wel gebruik maken van overervering en verdere OOP technieken, om zo het totaalplaatje overzichtelijk te houden.

3.1. UML diagrammen

Hieronder zijn de diagrammen van deze stap te vinden, deze diagrammen zijn van tevoren gemaakt en zullen waarschijnlijk niet het uiteindelijke product representeren.

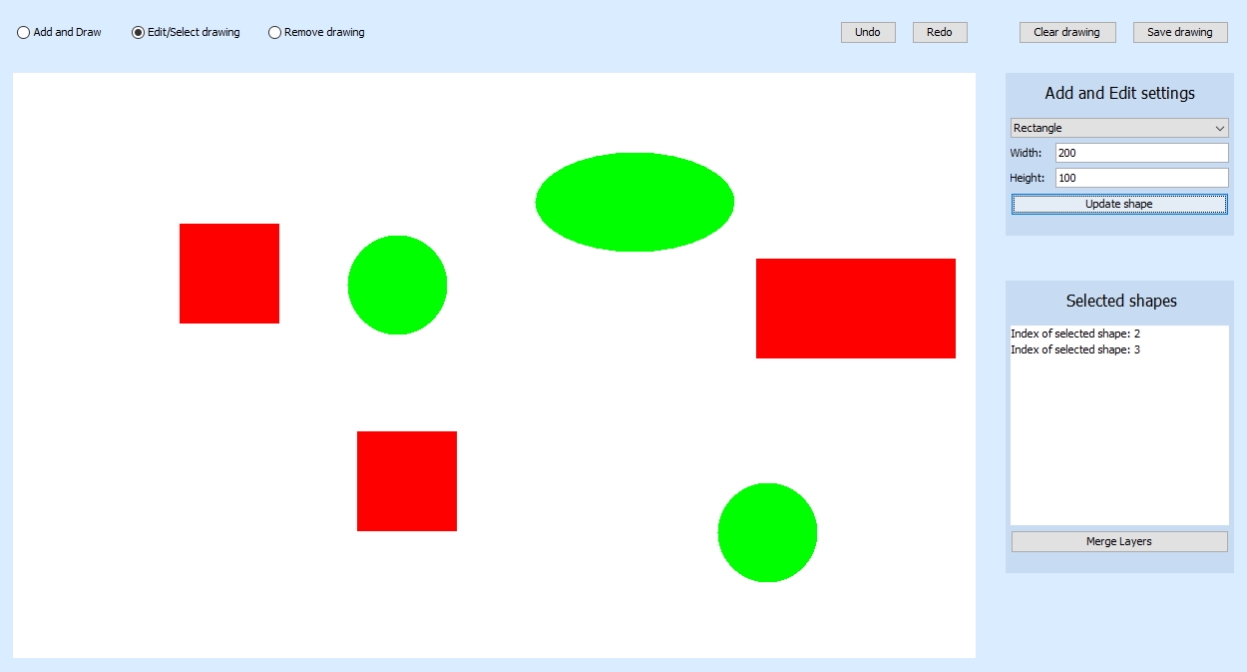


Figuur 2 – Klassendiagram eerste iteratie

De eerste versie van het klassendiagram is redelijk simpel, na wat uitzoekwerk kwam naar boven dat we een deel van onze eerder gemaakte userinterface konden overnemen. Om te kunnen tekenen moeten we aan een custom JPanel maken en deze koppelen aan een bestaande JPanel in de UserInterface klasse.

In deze DrawPanel zit dan weer de logica waarin al het tekenen gebeurd, vanuit de UserInterface krijgen wij met behulp van listeners binnen welke actie wij moeten uitvoeren. In de DrawPanel zelf zit een ArrayList met de verschillende Shapes. Op deze shape zelf kunnen wij weer verschillende functies uitvoeren, denk hierbij aan het aanpassen van de size of het instellen van de coordinaten.

3.2. Eindproduct

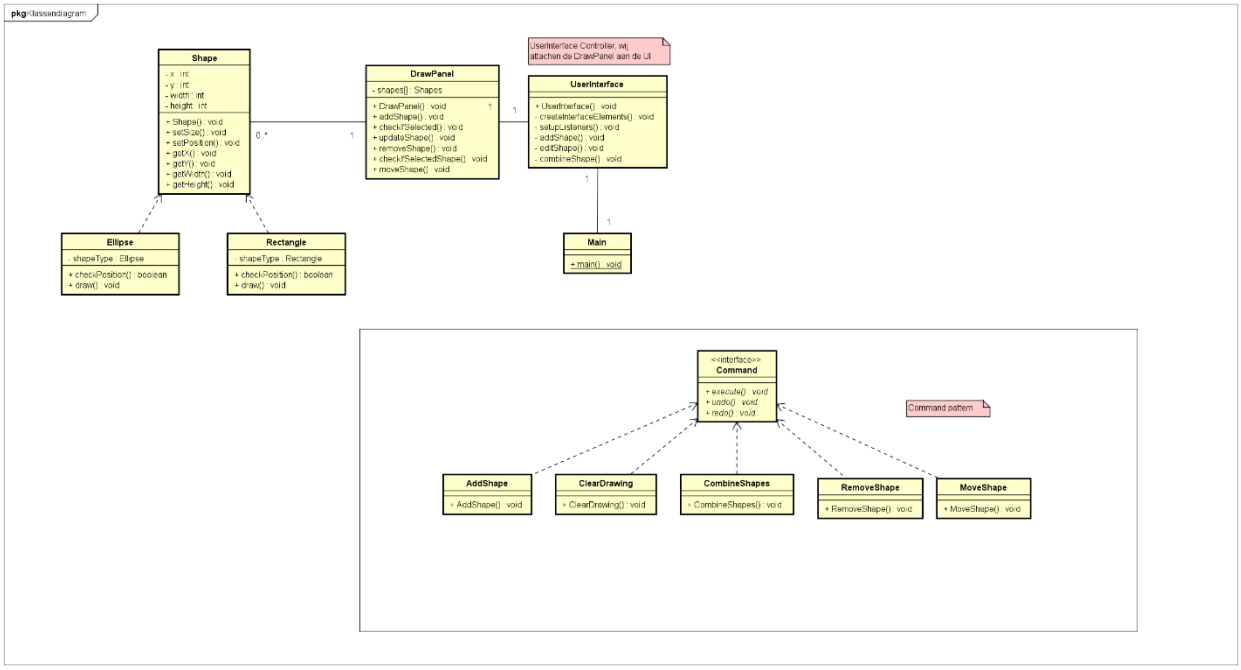


Bij versie stap 1 van de code kun je nu shapes selecteren door er op te klikken, toevoegen, verplaatsen en als je aan de rechterkant klikt op updaten, dan update de applicatie alle shapes die je geselect hebt

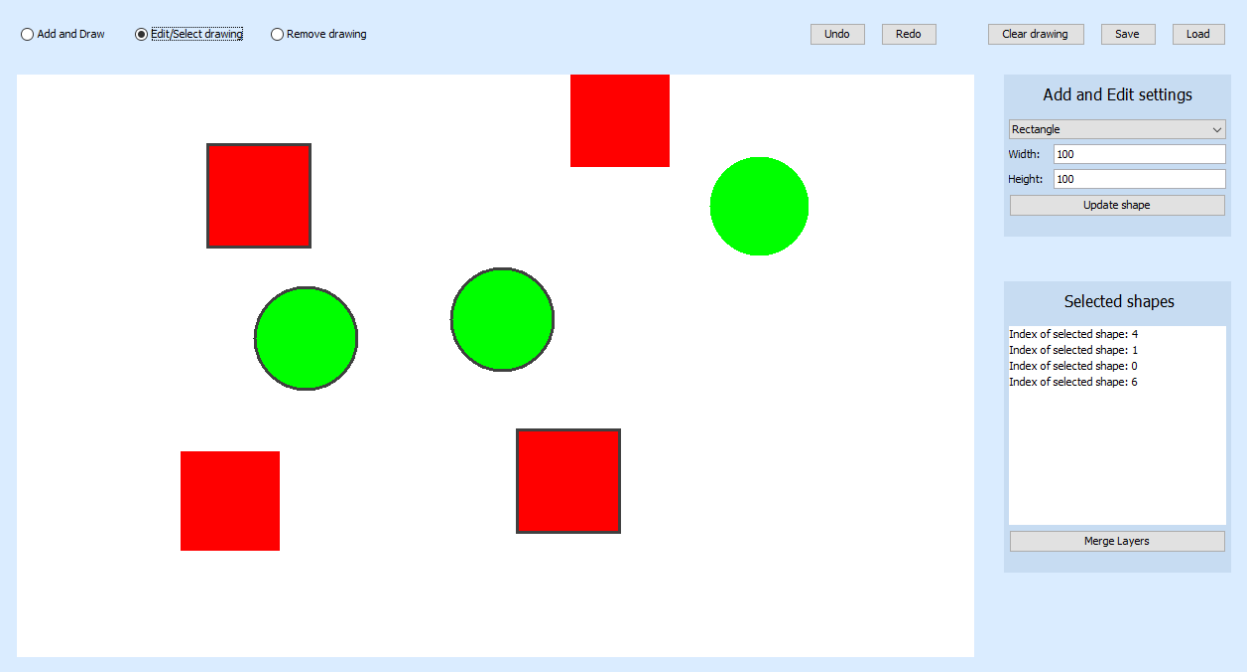
4. Stap 2: Command pattern

Bij stap 2 zullen de versie van de vorige stap verder ontwikkelen en nieuwe functionaliteit toevoegen. In deze stap zal deze functionaliteit vooral bestaan uit het toevoegen van de File IO (load/save) en het implementeren van het command pattern.

4.1. UML diagrammen



4.2. Eindproduct



Code:

Aan het einde van deze stap hebben we het command pattern geimplementeerd in ons project. Het Adden, clearen, combinen, remove en moven van de shapes gaat nu via het command pattern. Verder is er ook een start gemaakt aan het toevoegen van de File IO, zo is de structuur van de shapes wat aangepast en kun je nu ook opslaan en laden.

Het opslaan en laden gaat via JSON, omdat een van de teamleden al wat ervaring had met het gebruik van JSON is er besloten om dit te implementeren. Voordat we dit konden doen moesten we eerst van het plain Java project, een Maven project maken. Om JSON functionaliteit in Java te krijgen, moesten we de plugin “org.json.simple” gebruiken.

Visueel:

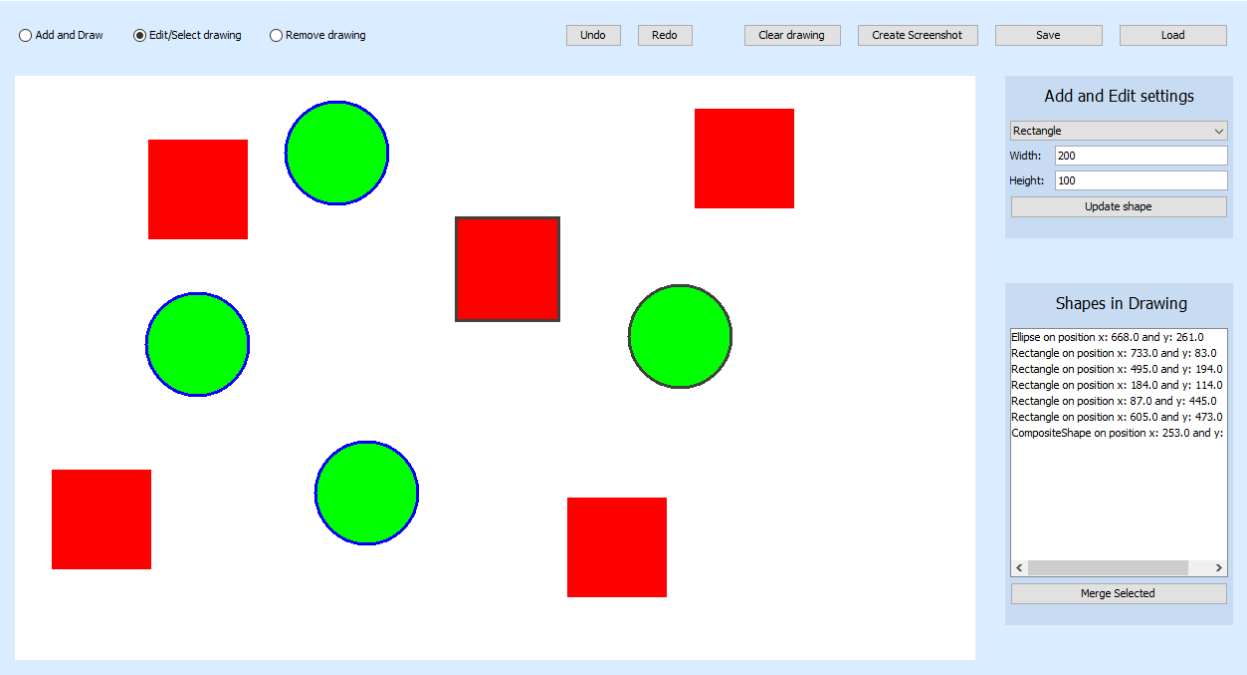
Äan de visuele kant zijn er ook enkele wijzigingen doorgevoerd, zo krijg je nu visuele feedback bij het selecteren van shapes en is het plaatsen/editen van de shapes wat gebruiksvriendelijker gemaakt. Voorheen als je een shape plaatste, kwam deze niet op het punt van de cursor terecht. Met gebruik van wat berekingen komt deze nu wel op de correcte xy positie en gaat het slepen ook wat soepelerer.

5. Stap 3: Composite pattern

Bij stap 3 zullen de versie van de vorige stap verder ontwikkelen en nieuwe functionaliteit toevoegen. In deze stap zal deze functionaliteit vooral bestaan uit het toevoegen van het groepensysteem, het implementeren van het composite pattern en het refactoren van de bestaande code.

5.1. UML diagrammen

5.2. Eindproduct



Code:

Visueel:

Aan de visuele kant zijn ook wat dingen aangepast, zo kan je nu via het sidemenu shapes selectere, is er een screenshot save button en zijn er verschillende outlines die de shapes omringen gebaseerd op de geselecteerde shapes. Als een shape of group geselecteerd is via het zijmenu, zal deze blauw zijn. Als dit via de DrawPanel gaat, zal deze blauw zijn. Bij het selecteren van meerdere shapes vanuit de view en het zijmenu, zal de applicatie deze bij elkaar pakken en samenvoegen tot 1 group. Als een gebruiker een group selecteerd, zal elke shape een border krijgen en kan de gebruiker deze ook moven.

6. Stap 4: visitor pattern

Bij stap 4 zullen de versie van de vorige stap verder ontwikkelen en nieuwe functionaliteit toevoegen.

6.1. UML diagrammen

6.2. Eindproduct

Code:

Tijdens deze stap kwamen we er achter dat het eigenlijk de bedoeling is om een text bestand te laden en niet een json bestand. Want de uiteindelijke FileIO structuur moet overeenkomen met die in de uitleg. Als je nu opslaat, dan saved de applicatie nu een json en een tekstbestand. Tijdens het laden kan je zowel tekst als json selecteren, de applicatie zelf controleert of wat voor type bestand het is en begint met laden.

Visueel:

Er zijn geen nieuwe knoppen bijgekomen, maar aan de visuele kant zijn er wel kleine dingen aangepast.

7. Stap 5: Strategy pattern & Composite pattern

Bij stap 5 zullen de versie van de vorige stap verder ontwikkelen en nieuwe functionaliteit toevoegen.

7.1. UML diagrammen

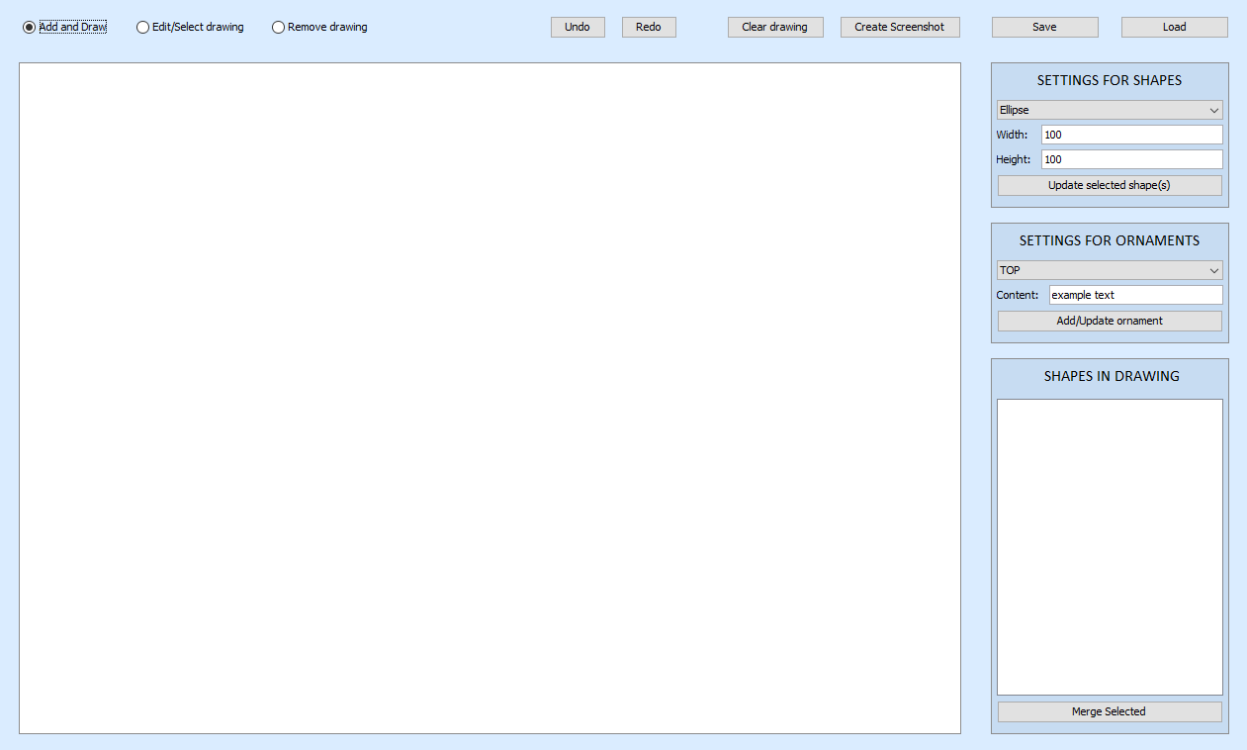
7.2. Eindproduct

8. Stap 6: Decorator pattern

Bij stap 6 zullen de versie van de vorige stap verder ontwikkelen en nieuwe functionaliteit toevoegen.

8.1. UML diagrammen

8.2. Eindproduct



Code:

Visueel:

Aan de visuele kant hebben zijn er nog enkele styling dingetjes aangepast, om de applicatie wat netter te maken.