

Rapport

projet Chromat-Ynk

Introduction

- Membres de l'équipe :

- Clément LE COADOU
- Kevin CHEN
- Pascal KAU
- Hicham BETTAHAR
- Baptiste GERMAIN

- Objectifs principaux

Création d'une interface graphique qui prend des instructions permettant de dessiner des formes quelconque avec la possibilité de changer les paramètres de dessins. Ces instructions seront définies dans un langage qui sera interprété et exécuté. On aura la possibilité de sauvegarder le dessin et d'importer un fichier d'instruction qui sera exécuté.

- Cahier des charges

Le cahier des charges était actualisé à chaque fin de session sur github. ([annexe](#))

- Organisation

Le projet est développé sous github pour prévenir la perte de travail durant le développement de notre application, faciliter une collaboration efficace en équipe, et permettre un partage aisé de nos progrès de développement avec notre chargé de projet : J. Mercadal, avec lequel diverses réunions ont été programmées pour le suivi et la direction du projet.

- Choix et enjeux

Nous avons choisi ce projet dans le but de développer notre capacité à répondre à un problème complexe nécessitant travail d'équipe et organisation, tout en respectant le délai accordé.

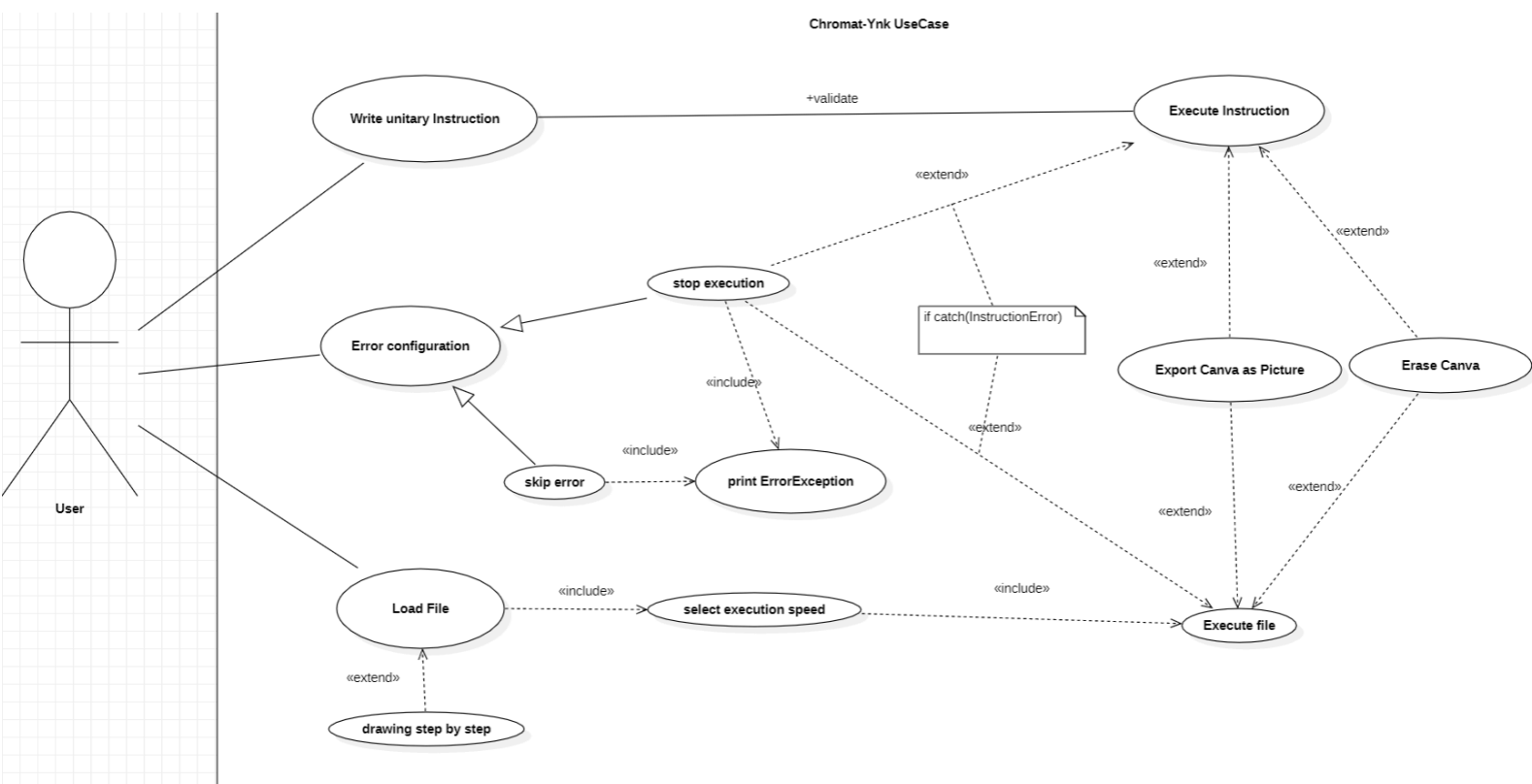
Sommaire

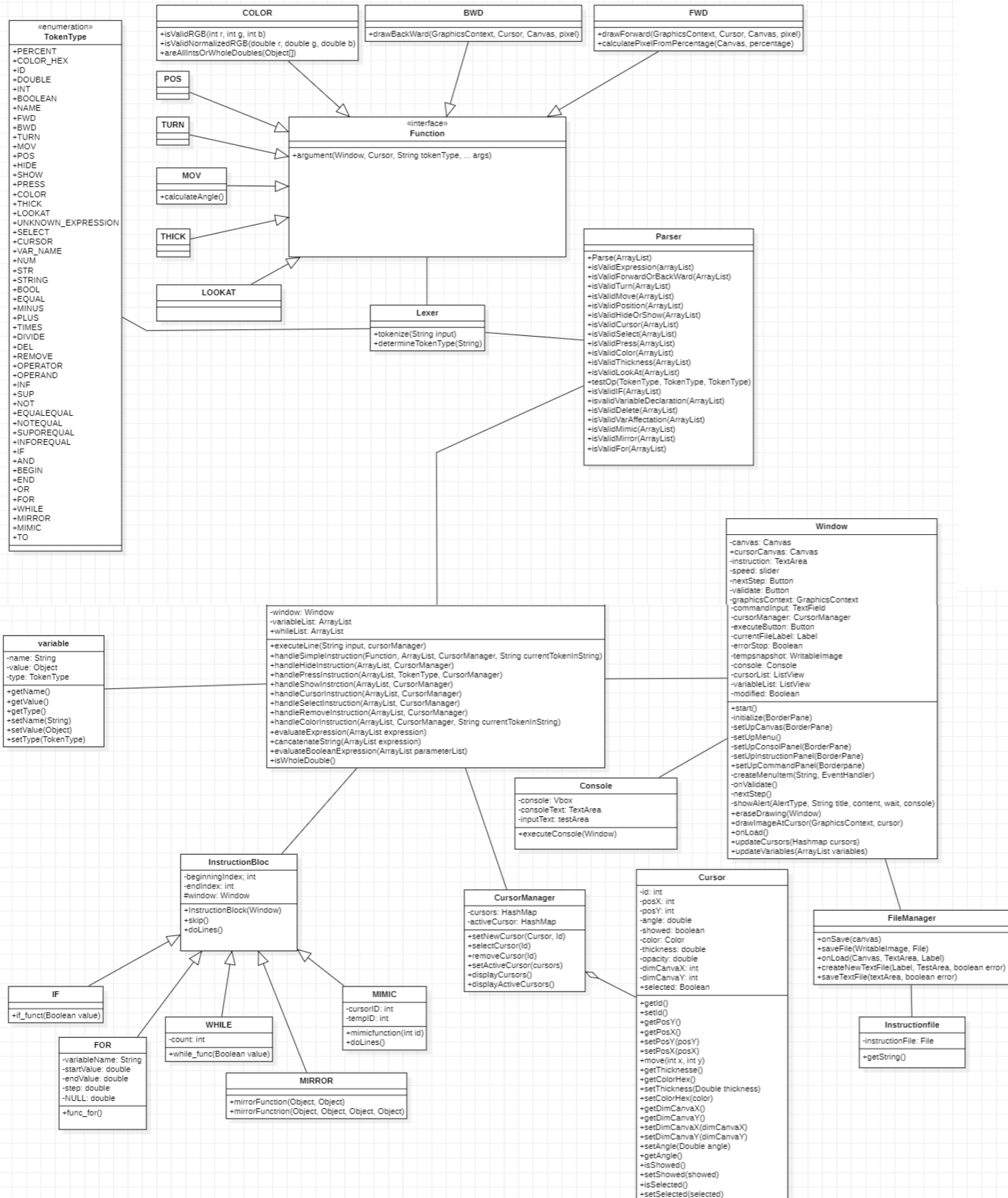
Introduction	1
- Membres de l'équipe :	1
- Objectifs principaux	1
- Cahier des charges	1
- Organisation	1
- Choix et enjeux	1
Sommaire	2
Analyse et Conception	3
1. Diagramme de cas d'utilisation	3
2. Diagramme de classes	4
3. Diagramme de séquence	5
Développement et Implémentation:	6
1. Canvas:	6
2. Curseur:	6
3. Langage d'instruction simple:	6
4. Langage d'instruction bloc:	7
5. Interpréteur:	7
6. Commentaire et debug:	7
Problèmes Rencontrés et Solutions Apportées	8
1. Compréhension du sujet et initialisation au projet	8
2. Canvas et affichage du curseurs	9
3. Interpréteur	9
4. Bloc d'instruction	10
5. Variable (a modifier)	10
6. Opérateurs	10
7. Dépendances mal gérées :	10
Résultats et Évaluation	11
Exemple d'exécution d'un fichier chargé avec Boucle et Mirror	12
Exemple pour des instructions unitaires	12
Utilisation des Variables et Opérations et couleur	13
Dessin:	13
Conclusion	14
Annexe	15
Cahier des charges final	17

Analyse et Conception

1. Diagramme de cas d'utilisation

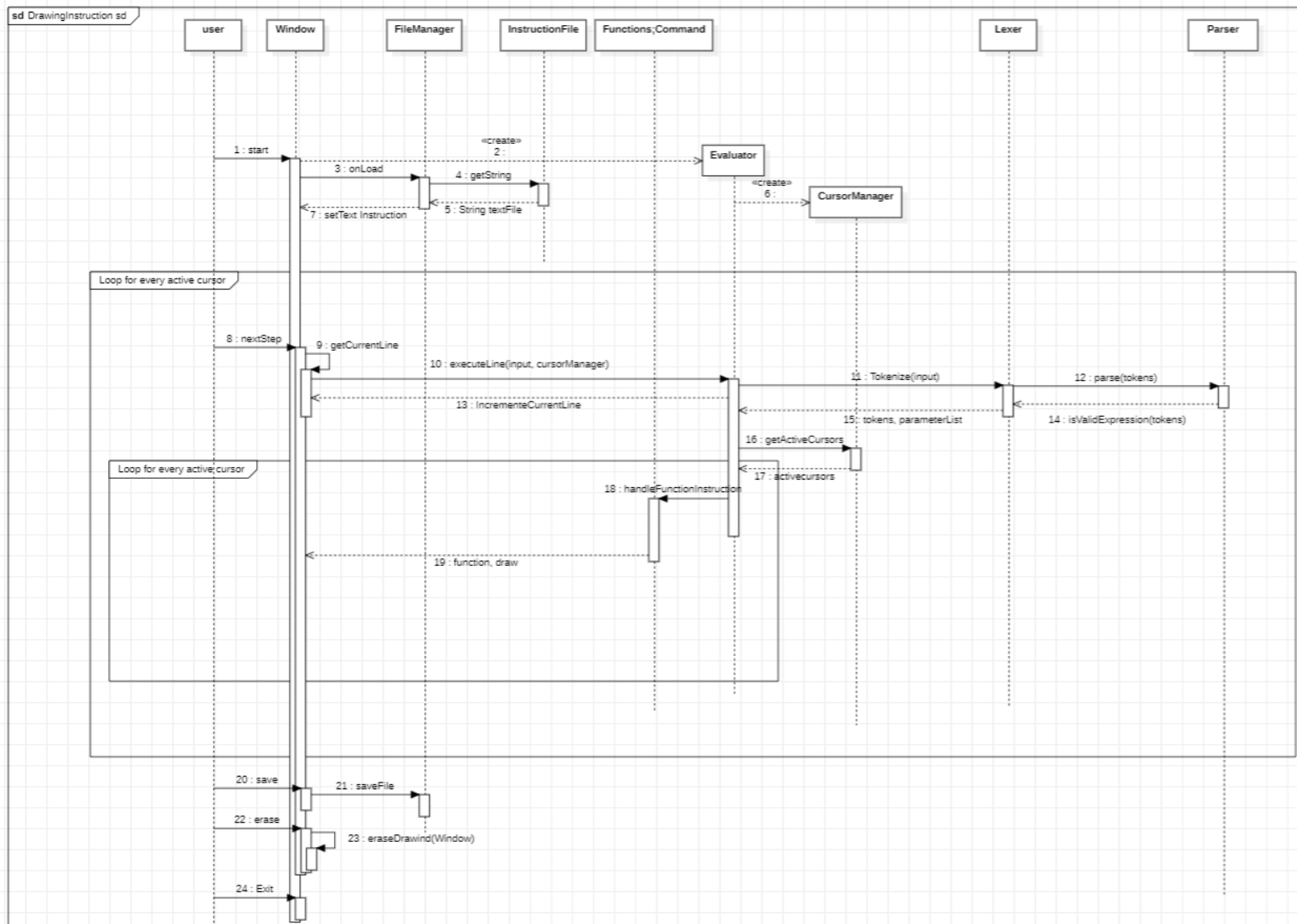
L'utilisateur peut entrer une instruction unitaire dans le bloc d'entrée d'instruction et l'exécuter. Choisir la gestion d'erreur avant l'exécution d'un fichier d'instruction qui pourra être chargé. Le dessin peut être sauvegardé et effacé.





3. Diagramme de séquence

Exemple de diagramme de séquence pour le dessin lors d'un chargement de fichier.



Développement et Implémentation:

1. Canvas:

Mise en forme du GUI par un Borderpane, elle nécessite un menu de gestion de fichier, un canvas sur lequel sera représenté le dessin, une console qui affichera les messages d'erreur et une zone d'entrée d'instruction.

2. Curseur:

Les dessins se feront à l'aide de curseurs qui stockent les propriétés des traits à dessiner, notamment la direction, l'épaisseur, couleur ... Ces curseurs peuvent être créés et supprimés, affichés et cachés, déplacés ...

3. Langage d'instruction simple:

On définit les instructions simples par leur nom de fonction et argument, ces fonctions seront définies pour dessiner sur le canva en fonction des valeurs passées en argument.

- FWD INT/DOUBLE(%)
- BWD INT/DOUBLE(%)
- COLOR #HEX , COLOR INT/DOUBLE INT/DOUBLE INT/DOUBLE
- LOOKAT INT , LOOKAT INT/DOUBLE INT/DOUBLE
- MOV INT/DOUBLE INT/DOUBLE
- POS INT/DOUBLE INT/DOUBLE
- PRESS DOUBLE ,PRESS INT/DOUBLE%
- THICK INT/DOUBLE
- TURN INT/DOUBLE

- HIDE INT
- SHOW INT
- CURSOR INT
- SELECT INT
- REMOVE INT

- NUM STRING (= INT/DOUBLE)
- STR STRING (= STRING)
- BOOL STRING (= BOOL)
- DEL STRING

4. Langage d'instruction bloc:

On définit les instructions bloc similairement aux instructions simples, en implémentant la détection d'instruction encapsulée.

Les blocs sont écrit de la manière suivante :

- ```
BLOC {
 INSTRUCTION
}
```
- IF BOOLEAN
  - FOR var\_name TO INT/DOUBLE ,  
 FOR var\_name INT/DOUBLE TO INT/DOUBLE ,  
 FOR var\_name TO INT/DOUBLE INT/DOUBLE,  
 FOR var\_name INT/DOUBLE TO INT/DOUBLE INT/DOUBLE
  - WHILE BOOLEAN
  - MIMIC INT
  - MIRROR INT/DOUBLE(%) INT/DOUBLE(%)
  - MIRROR INT/DOUBLE(%) INT/DOUBLE(%) INT/DOUBLE(%) INT/DOUBLE(%)

## 5. Interpréteur:

Utilisation d'un Lexer pour créer des tokens dont la validité est vérifiée dans le parser, ces tokens seront stockés dans un tableau où l'interpréteur va exécuter ces instructions.

## 6. Commentaire et debug:

Utilisation de commentaires permettant la génération d'une JavaDoc. Gestion des erreurs avec des blocs try et catch.

# Problèmes Rencontrés et Solutions Apportées

## 1. Compréhension du sujet et initialisation au projet

Au tout début, le 07/05, nous sommes partis sur une création de classe pour chaque commande, nous pensions que vous devions créer chaque classe de commande, et ce sont ces classes qui s'occuperont d'agir sur le canvas à travers une classe `DrawingInstruction` directement après avoir entré un texte en entrée comme vous pourrez le voir sur notre diagramme de classe fait au tout début du projet ([voir annexe](#)).

Mais ce n'est qu'après notre réunion avec notre chargé de projet le 08/05, que nous nous sommes rendus compte de l'erreur que nous faisions, ne comprenant pas l'essence du projet même, qui est de créer un interpréteur, qui s'occupera de la lecture des commandes passé en entrée et qui fera ensuite appel à la fonction de dessin définie.

A travers ses directives nous avons une meilleure vision des étapes clés, nous devons ainsi formuler notre cahier des charges:

*Exemple de tâche du cahier des charges:*

### 1. Réalisation 3 semaines

#### 1.0. Créer interpréteur de langages : 3 semaines

##### 1.0.0 Instructions simples

1.0.0.0 Ajout de la classe `Simple_Instruction`

1.0.0.1 Ajout de la commande `FWD`

1.0.0.2 Ajout de la commande `BWD`

1.0.0.3 Ajout de la commande `TURN`

1.0.0.4 Ajout de la commande `MOV`

1.0.0.5 Ajout de la commande `POS`

1.0.0.6 Ajout de la commande `HIDE`

1.0.0.7 Ajout de la commande `SHOW`

1.0.0.8 Ajout de la commande `COLOR #hex`

1.0.0.9 Ajout de la commande `COLOR r g b`

1.0.0.10 Ajout de la commande `THICK`

1.0.0.11 Ajout de la commande `LOOKAT cursorId`

1.0.0.12 Ajout de la commande `LOOKAT x% y%`



Comme vous pouvez le voir, nous ne faisons qu'une série de sous-tâches sans réellement comprendre l'objectif même du cahier des charges, nous avions une tâche qui est censé durer l'entièreté du projet, autrement dit, c'est le projet en lui-même ce qui n'avait pas de sens ici, nous n'avions pas estimé chaque sous-tâche, les tâches n'étaient pas non plus décrites, ni la méthode de validation, ni la description, ainsi que la répartition des tâches.

Après vérification avec notre chargé du projet, nous avons compris que le cahier des charges ne faisait pas réellement ce qui était attendu. Nous avons ainsi mis à jour ce dernier, incluant une description plus large dans le ReadMe, et une description partiel dans le cahier des charges, nous avons ajouté plus de détails aux tâches, fait une estimation pour chaque tâche et macro tâche et répartir les tâches.(pour plus de détails, [voir annexe](#))

## 2. Canvas et affichage du curseurs

Nous avons rencontré un problème sur l'affichage du curseur sur le canvas, en effet, nous avons utilisé une image pour représenter le curseur, se faisant il n'y a pas de méthode prédéfinie pour afficher l'image sur le canvas tout en le mettant dans la bonne direction. Nous avons essayé d'implémenter des méthodes qui permettaient de directement modifier la rotation de l'image en elle-même avant de l'afficher, mais en vain et pas efficace. C'est alors que nous avons pensé différemment au lieu de faire tourner l'image, pourquoi ne pas faire tourner le canvas ? c'est en effet, de ce point de vue là que nous nous sommes basés. En utilisant les méthodes save et restore de la classe GraphicsContext que nous avons réussi à résoudre ce problème:

- sauvegarde des paramètres du contexte graphique, position initiale de la fenêtre
- aller à la position du curseur
- faire tourner le canvas par rapport à ce centre
- dessiner le curseur
- restaurer les paramètres du contexte graphique

## 3. Interpréteur

Lors de l'implémentation de la classe interpréteur, pour lire nos ligne de commande, nous prenions en entrée le numéro de ligne, et nous faisons un *tokenize*, ie séparation en tokens, et *parser*, ie validation des tokens sur chaque ligne, ainsi nous parcourons plusieurs fois le code en entier à l'appel de l'exécution de chaque ligne, ce qui fait une redondance de code, pour corriger cela, nous faisons un tokenize et parser avant l'exécution de la ligne, ainsi on n'exécute la ligne qu'après avoir obtenue les tokens vérifiés et non avant.

## 4. Bloc d'instruction

Nous n'avions pas du tout idée de la manière de réaliser les instructions encapsulés, nous avons commencé par faire des reconnaissances d'accolades, mais cela ne résout pas le problème, étant donnée que nous ne savons pas quel fermeture d'accolade correspond à quel type de fonction dont l'accolade est ouverte.

A l'aide d'une vidéo sur la reconnaissance des mots (présent dans la bibliographie) consistant à cerner les caractères commençant par " des caractères commençant par ' pour les chaînes de caractères, l'utilisation d'un compteur permettait, par sa parité de déterminer l'appartenance de l'index de fermeture de bloc, ici ' ou " .

En utilisant le même principe par un compteur, on incrémente lors d'une accolade ouvrante ( { ) et on décrémente lors d'une accolade fermante ( } ). Lorsque le compteur revient à zéro, on est à la sortie de la boucle.

## 5. Variable

Difficultés lors de la reconnaissance des variables de type double des variables de type int lors de l'affectation de valeur, par exemple `NUM var1 = var1 * 2 / 4 + 24`, et `NUM var2 = 2.0 + var2`, nous pensions qu'il fallait modifier chacune des fonctions pour les faire correspondre au type de la variable pour pouvoir les différencier car on commence d'abord par parser, ie validé la validité de l'expression.

Résolution du problème : nous avons pensé à une solution après avoir vérifié d'abord que la variable existe, avant de parser, on analyse l'expression pour s'assurer qu'elle est bien formée, l'évalue, et enfin assigne le résultat à la variable, de cette manière il n'y a pas de problème d'affectations de type.

Cela va de même pour les types booléens et pour leur assigner des expressions logiques.

## 6. Opérateurs

Difficultés sémantiques sur l'implémentation des opérations de comparaison, comme le nombre d'argument a comparer du type: "A ou B ou C et D".

Nous nous sommes limités à une comparaison maximale, du type: "A ou B". Étant donné que le choix d'implémentation était laissé libre, le cas précédent peut être fait par un bloc imbriqué.

## 7. Dépendances mal gérées :

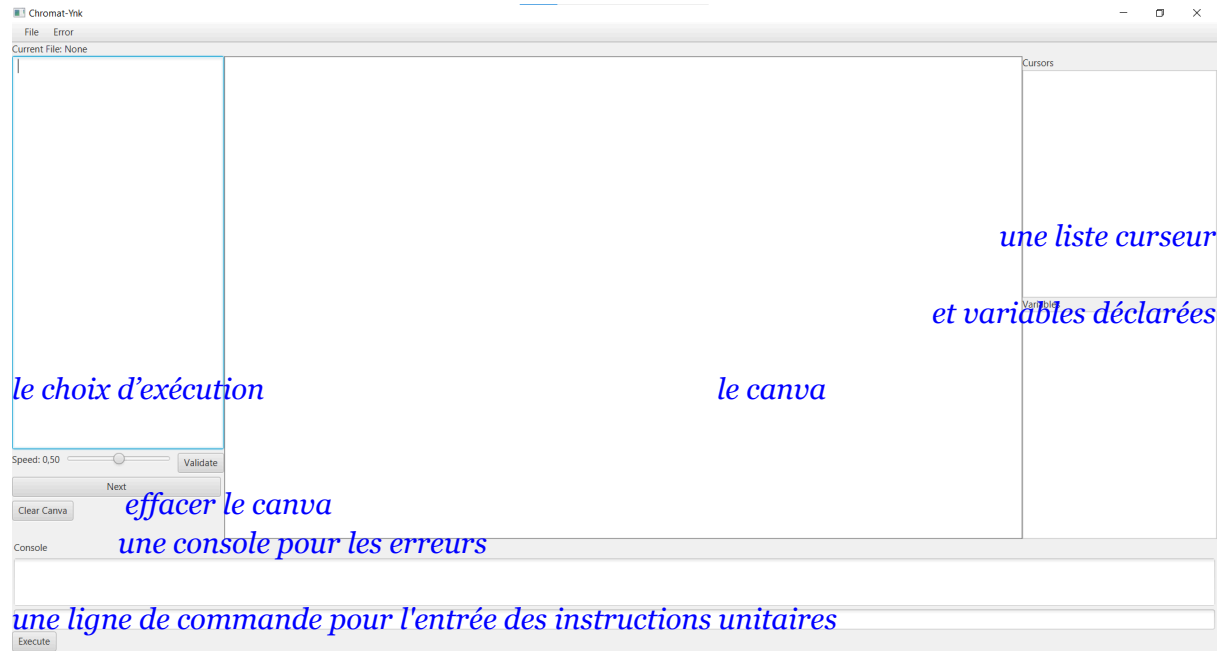
Les dépendances entre différentes parties du projet ne sont pas bien gérées, causant des blocages et des retards, notamment pour la javadoc ou le code n'était pas commenté au début, nous nous sommes vite perdus.

## Résultats et Évaluation

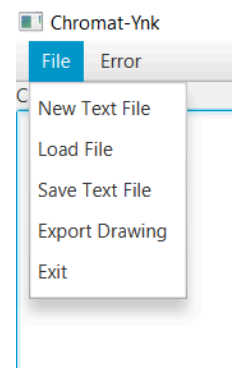
Fenêtre de base:

Inclut:

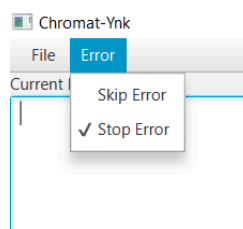
*le chargement/sauvegarde de fichier    la gestion d'erreur*



menu File:



menu Error:



## Exemple d'exécution d'un fichier chargé avec Boucle et Mirror

Chromat-Ynk

File Error

Current File: None

```

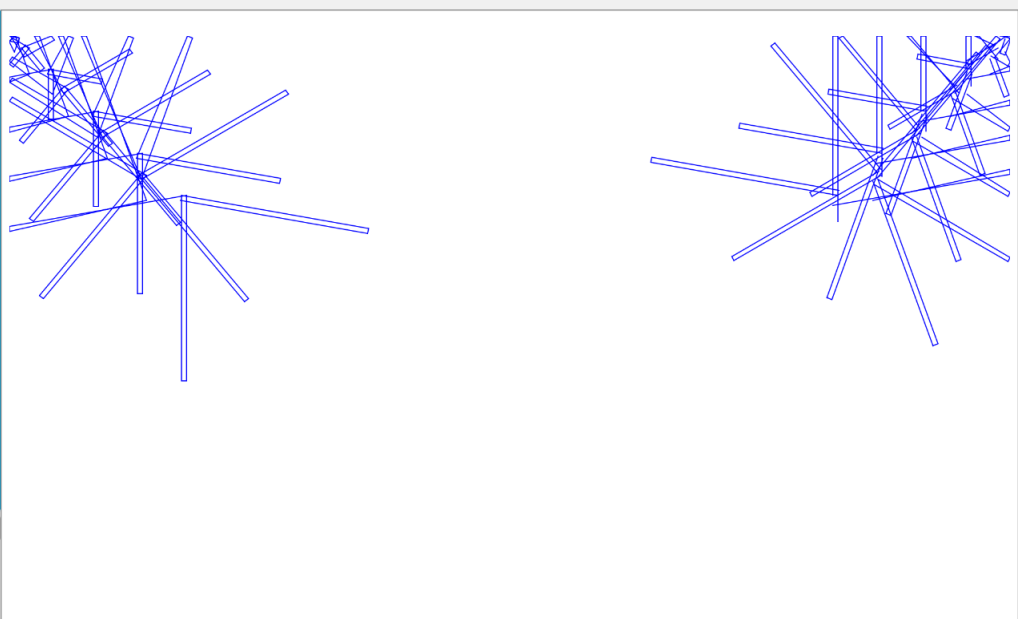
NUM radius = 10
NUM step = 5
NUM angle = 10
NUM total_steps = 36
CURSOR 0
SELECT 0
FOR i 0 TO total_steps 1 {
 MIRROR 50% 0% 50% 100% {
 TURN angle
 FWD radius
 TURN 90
 FWD step
 TURN 90
 FWD radius
 TURN -90
 radius = radius + step
 }
}

```

Speed: 0.00

Console

END seen  
command FOR done.



Cursors

Cursor n°0

Variables

DOUBLE radius=195.0  
DOUBLE step=5  
DOUBLE angle=10  
DOUBLE total\_steps=36  
DOUBLE i=36.0

## Exemple pour des instructions unitaires

Chromat-Ynk

File Error

Current File: None

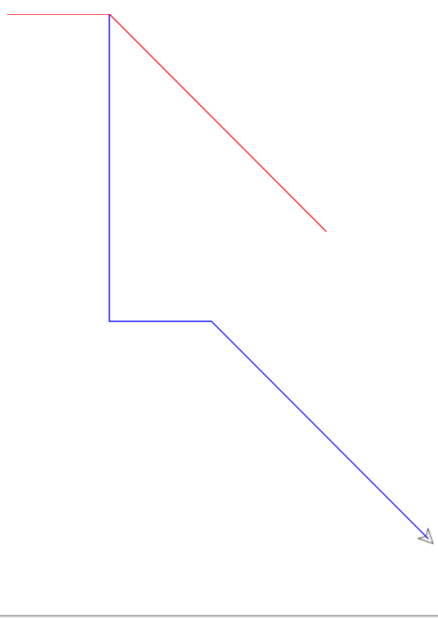
Enter instructions or load a file

Speed: 0.50

Console

command CURSOR done.  
command SELECT done.  
command SHOW done.

FWD 300



Cursors

Cursor n°0  
Cursor n°1

Variables



## Conclusion

Tout d'abord le développement de ce projet a été une expérience enrichissante qui nous a permis de relever plusieurs défis techniques et organisationnels. En créant une interface graphique permettant de dessiner des formes à partir d'instructions interprétées, nous avons pu appliquer et approfondir nos connaissances en programmation Java, en manipulation graphique et en conception de langages spécifiques.

Puis le travail sous GitHub a facilité la collaboration en équipe, prévenu les pertes de travail et permis un suivi constant de l'avancement du projet. Les réunions régulières avec notre chargé de projet, M. Mercadal, ont été cruciales pour orienter le projet et surmonter les obstacles rencontrés.

De plus, la création de diagrammes de cas d'utilisation, de classes et de séquence nous a aidés à structurer notre projet de manière claire et méthodique. Cette étape a été essentielle pour définir les fonctionnalités et les interactions entre les différentes parties de notre application.

Ensuite, nous avons mis en place des éléments clés comme le Canvas, les curseurs, et les langages d'instruction simple et bloc. L'implémentation de l'interpréteur a été particulièrement formatrice, nous obligeant à comprendre et à créer un système capable de lire, valider et exécuter des instructions complexes.

Le projet nous a confrontés à divers problèmes, notamment la gestion du curseur sur le Canvas, l'interprétation des blocs d'instructions, et la différenciation des types de variables. Chaque problème a été une opportunité d'apprentissage, nous poussant à trouver des solutions innovantes et efficaces.

Les résultats obtenus sont satisfaisants. Nous avons réussi à implémenter un système fonctionnel permettant de dessiner des formes complexes et de manipuler divers paramètres de dessin. Les exemples d'exécution montrent que notre application peut gérer des instructions unitaires et des blocs d'instructions, ainsi que l'utilisation de variables et de différentes opérations.

Enfin le projet pourrait être enrichi par l'ajout de nouvelles fonctionnalités, telles que la gestion avancée des erreurs, l'optimisation des performances et l'amélioration de l'interface utilisateur permettant l'amélioration de l'expérience utilisateur.

En conclusion, ce projet nous a permis de développer des compétences cruciales en programmation, en gestion de projet et en travail d'équipe. Nous estimons le chemin parcouru et les connaissances acquises seront un atout pour la programmation graphique.

Clément LE COADOU Kevin CHEN Pascal KAU  
Hicham BETTAHAR Baptiste GERMAIN

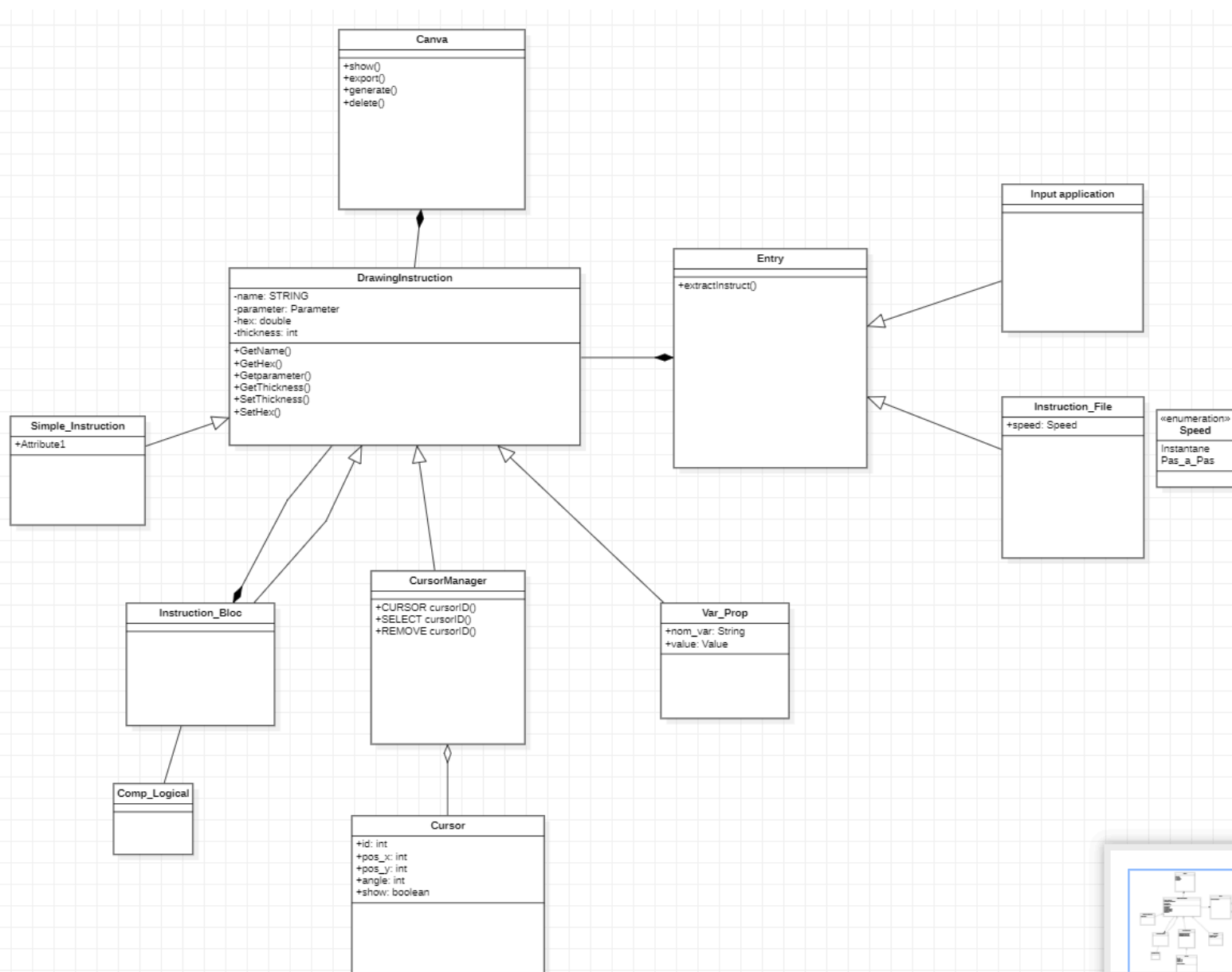
## Annexe

### Bibliographie:

- **Sakib Hadživdić: Writing an interpréter from scratch:**  
<https://www.toptal.com/scala/writing-an-interpretter>
- **ThelvoryCoder: Vidéo sur l'écriture d'un interpréteur en python**  
[https://youtube.com/playlist?list=PLqAnJa9r9pt0jDijGtRHlr3JpYLYL\\_HB&feature=shared](https://youtube.com/playlist?list=PLqAnJa9r9pt0jDijGtRHlr3JpYLYL_HB&feature=shared)
- **Color class documentation**  
[https://docs-oracle-com.translate.goog/javafx/2/api/javafx/scene/paint/Color.html?\\_x\\_tr\\_sl=en&\\_x\\_tr\\_tl=fr&\\_x\\_tr\\_hl=fr&\\_x\\_tr\\_pto=rq](https://docs-oracle-com.translate.goog/javafx/2/api/javafx/scene/paint/Color.html?_x_tr_sl=en&_x_tr_tl=fr&_x_tr_hl=fr&_x_tr_pto=rq)

Problème: compréhension du sujet et initialisation au projet

Diagramme de classe, début de projet



## Organisation du cahier des charges

### Discussions

|                                                                                       |                                                                                                                                                                                                     |                                                                                       |                                                                                         |
|---------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------|
|  1   |  <b>24 mai 2024</b><br><a href="#">kevinCYt</a> started yesterday in <a href="#">TaskList</a>                      |    |  0   |
|  1   |  <b>23 mai 2024</b><br><a href="#">kevinCYt</a> started 3 days ago in <a href="#">TaskList</a>                     |    |  0   |
|  1   |  <b>22 mai 2024</b><br><a href="#">kevinCYt</a> started 4 days ago in <a href="#">TaskList</a>                     |    |  0   |
|  1   |  <b>21 mai 2024</b><br><a href="#">kevinCYt</a> started 5 days ago in <a href="#">TaskList</a>                     |    |  0   |
|  1   |  <b>17 mai 2024</b><br><a href="#">kevinCYt</a> started last week in <a href="#">TaskList</a>                      |    |  0   |
|  1  |  <b>16 mai 2024</b><br><a href="#">kevinCYt</a> started last week in <a href="#">TaskList</a>                     |   |  0  |
|  1 |  <b>15 mai 2024</b><br><a href="#">kevinCYt</a> started 2 weeks ago in <a href="#">TaskList</a>                  |  |  0 |
|  1 |  <b>14 mai 2024 (MAJ TO-DO-LIST)</b><br><a href="#">kevinCYt</a> started 2 weeks ago in <a href="#">TaskList</a> |  |  0 |
|  2 |  <b>07 mai 2024</b><br><a href="#">bapger</a> started 3 weeks ago in <a href="#">TaskList</a>                    |  |  2 |



# Cahier des charges final

24 mai 2024 #26

kevinCYt started this conversation in TaskList



kevinCYt yesterday

Collaborator

edited ▾ ...

Pour une plus large description, [ReadMe](#)

## Initialisation du projet

- ✓ Création du Template (Estimations : 3h+)
  - ✓ Identification des tâches (2h) @bapger
  - ✓ Répartition des tâches ( 30min ) @kevinCYt
  - ✓ Mise en place des délais (30min) @kevinCYt
- ✓ Diagramme de cas d'utilisation (1h) @clement-le-coadou @kevinCYt @Poupouski @bapger
- ✓ Diagramme de classe (6h) @kevinCYt
- ✓ Diagramme de sequence (3h) @kevinCYt

## Définir les Classes et les Structures de Données

- ✓ Classe Window : (Estimation : 13 heures)

*Elle permet l'affichage graphique ainsi que la fenêtre de dessin*

  - ✓ Création de classe implémentant le canvas pour la représentation de la zone de dessin. (3 heures) @clement-le-coadou 07/05
  - ✓ Créer des méthodes pour dessiner, interface fonction qui prend différent argument pour les différentes méthodes. (8h)
    - ✓ FWD @Pascalcytech @clement-le-coadou 13/05
    - ✓ BWD @Pascalcytech @clement-le-coadou 13/05
    - ✓ TURN @Pascalcytech @clement-le-coadou 14/05
    - ✓ MOV @Pascalcytech @clement-le-coadou 14/05
    - ✓ POS @Pascalcytech 15/05
    - ✓ HIDE @clement-le-coadou
    - ✓ SHOW @clement-le-coadou
    - ✓ PRESS @kevinCYt
    - ✓ COLOR @Pascalcytech 17/05
    - ✓ THICK @Pascalcytech 17/05
    - ✓ LOOKAT @Pascalcytech 15/05
  - ✓ Effacer le canvas et sauvegarder le canvas sous forme de fichier image. (2 heures) @kevinCYt @clement-le-coadou 17/05
- ✓ Classe Cursor : (Estimation : 6 heures) @clement-le-coadou @Pascalcytech 14/05

*Création du curseur qui permet de connaître la direction des trait du dessin et le commencement du trait, elle stock toute les données liée a la construction du dessin*

✓ **Classe Cursor :** (Estimation : 6 heures) @Pascalcytech 14/05

*Création du curseur qui permet de connaître la direction des trait du dessin et le commencement du trait, elle stock toute les données liée a la construction du dessin*

✓ Stocker les propriétés du curseur telles que la position (x, y), l'orientation, la couleur et l'épaisseur. (1 heure) @Pascalcytech 14/05

✓ Implémenter des méthodes pour déplacer le curseur à l'image(5 heures) @bapger @kevinCYt @clement-le-coadou 16/05

✓ image du curseur sur interface graphique @bapger @kevinCYt @clement-le-coadou 16/05

✓ **Classe Variable :** (Estimation : 4 heures) @Poupouski 21/05-22/05

*variable servant à récupérer les propriétés des curseurs*

✓ Définir une classe pour représenter les variables avec des propriétés comme le nom, le type et la valeur. (2 heures)

✓ Stocker les variables, où la clé est le nom de la variable. (2 heures) 22/05

## Implémenter la Logique de Dessin

✓ **Interpreteur:** (Estimation: 17 heures)

✓ Implémenter le Lexer (4 h) @Poupouski 13/05

✓ Implémenter le Parser (5 h) @Poupouski 13/05

✓ Implémenter l'Interpreteur (8h) @clement-le-coadou @bapger @Poupouski 24/05

✓ instruction simple @clement-le-coadou 14/05

✓ boucle 24/05 @bapger 23/05

✓ IF 23/05 @bapger

✓ WHILE @bapger

✓ FOR @bapger

✓ variable 24/05 @Poupouski

## Développer la Gestion des Fichiers

✓ **Chargement des Instructions :** (Estimation : 2 heures)

*Permet l'importation d'un fichier text dans le but d'exécuter les instructions contenu*

✓ Implémentation de la console. @bapger 17/05 (1h)

✓ Lire les fichiers d'instructions ligne par ligne en utilisant BufferedReader. (1h) @bapger @clement-le-coadou

✓ **Sauvegarde du Dessin :** (Estimation : 4 heures) @bapger

*Permet de sauvegarder le dessin en tant qu'image s'il n'est pas vide*

✓ Utiliser des bibliothèques de traitement d'images comme javax.imageio pour sauvegarder le canvas sous forme de fichier image. (2h) @bapger 07/05

✓ verifier le cas vide (1h) @bapger

## Gérer les Conditions d'Erreur @bapger @kevinCYt 18/05

- ✓ **Gestion des Erreurs :** (Estimation : 5 heures)  
*Permet de définir le comportement de l'interpréteur face aux erreurs*
  - ✓ Utiliser des blocs try-catch pour intercepter les exceptions pendant l'exécution, telles que les instructions invalides ou les erreurs hors limites.(3h) @bapger @clement-le-coadou @Poupouski
  - ✓ Fournir des messages d'erreur descriptifs pour informer les utilisateurs des problèmes rencontrés.(1h) @bapger @clement-le-coadou @Poupouski
  - ✓ Choix du mode de gestion d'erreur (1h) @kevinCYt

## Gérer les Variables

- ✓ **Gestion des Variables :** (Estimation : 4 heures) @Poupouski 17/05- 22/05  
*Permet l'utilisation des variables dans les instructions*
  - ✓ Implémenter des méthodes pour créer, supprimer et mettre à jour les variables. (2h) @Poupouski
  - ✓ Analyser les instructions pour identifier les noms de variables et les valeurs, et les stocker dans la structure de données des variables. (2h) 23/05 @Poupouski @bapger

## Contrôler l'Exécution

*Cette section permet de voir la construction de la figure à l'écran*

- ✓ **Vitesse d'Exécution :** (Estimation : 3 heures) @Poupouski 21/05
  - ✓ Utiliser Timer ou AnimationTimer pour contrôler la vitesse d'exécution des fichiers d'instructions. (2h) @bapger
  - ✓ Permettre aux utilisateurs d'ajuster la vitesse à l'aide d'un curseur ou d'un champ de saisie. (1h) @Poupouski
- ✓ **Exécution Pas à Pas :** (Estimation : 4 heures) @bapger
  - ✓ Implémenter l'exécution des instructions une par une, en faisant une pause entre chaque étape.

## Gérer les Curseurs @clement-le-coadou

- ✓ **Création et Sélection des Curseurs :** (Estimation : 4 heures) @clement-le-coadou  
*Permet de sélectionner les curseurs pour les utiliser au même moment après les avoir créés*
  - ✓ Maintenir une liste ou une carte des curseurs pour gérer plusieurs curseurs sur le canvas. (2h) @clement-le-coadou 14/05
  - ✓ Fournir des méthodes pour créer de nouveaux curseurs avec des identifiants uniques et sélectionner des curseurs existants dans la liste. (2h) @clement-le-coadou 14/05
- ✓ **Suppression des Curseurs :** (Estimation : 2 heures) @clement-le-coadou 18/05
  - ✓ Implémenter une fonctionnalité pour supprimer les curseurs du canvas lorsqu'ils ne sont plus nécessaires.

## Gérer les Blocs d'Instructions @bapger @clement-le-coadou

- ✓ Instructions Conditionnelles et Boucles : (Estimation : 6 heures) 22/05 @bapger  
*implémentation des instructions conditionnel, détection des instructions par des accolades*
  - ✓ Implémenter la logique d'analyse des instructions IF, FOR et WHILE pour exécuter des blocs d'instructions de manière conditionnelle ou itérative. (4h) @bapger 23/05
  - ✓ Évaluer les expressions booléennes et exécuter les instructions incluses en conséquence. (2h) @bapger
- ⇄ ✓ Instructions Miroir (Estimation : 6 heures) @clement-le-coadou  
*les instructions de copie miroir permettent la duplication par symétrie / copie des curseurs*
  - ✓ Implémentation des instructions miroir MIMIC, MIRROR (axiale / centrale). @clement-le-coadou

## Implémenter l'Interface Utilisateur @bapger @clement-le-coadou

- ✓ Composants de l'Interface Graphique : (Estimation : 4 heures)  
*Mise en forme des différents composants de l'interface graphique*
  - ✓ Utiliser des composants JavaFX pour créer la fenêtre. (30 min) @clement-le-coadou 07/05
  - ✓ Inclure des champs de saisie, des boutons, des menus déroulants et d'autres éléments interactifs pour l'interaction utilisateur et la saisie des instructions. (3h) @clement-le-coadou @bapger 07/05
  - ✓ Utiliser un composant canvas pour visualiser la zone de dessin. (30 min) @bapger
- ✓ Boutons et Contrôles : (Estimation : 4 heures)  
*Permet le chargement des instructions, sauvegarde, effaçage*
  - ✓ Ajouter des boutons pour charger/sauvegarder des fichiers, contrôler la vitesse d'exécution et gérer les curseurs. (2h) @Poupouski @clement-le-coadou @bapger 07/05
  - ✓ Fournir des contrôles pour ajuster les paramètres et les préférences comme la gestion des erreurs. (2h) @bapger @clement-le-coadou @Poupouski 16/05
  - ✓ Mise en place d'un canvas pour le curseur et d'un autre pour les dessins. (30 min) @clement-le-coadou

## Documentation et Test @everyone

- ✓ Documentation : (Estimation : 7 heures)
  - ✓ Rédiger une documentation détaillée pour les classes, les méthodes et les fonctionnalités. (4h)
  - ✓ Inclure des commentaires de code et des commentaires Javadoc pour expliquer le but de chaque composant. (3h)
- ✓ Tests : (Estimation : 3 heures)
  - ✓ Réaliser des tests d'intégration pour s'assurer que toutes les parties de l'application fonctionnent ensemble de manière transparente.

Temps total estimé : 106 heures