Programmation Concurrente et Interactive
Cours 2 : méthodologie de gestion de projet

Nicolas Sabouret

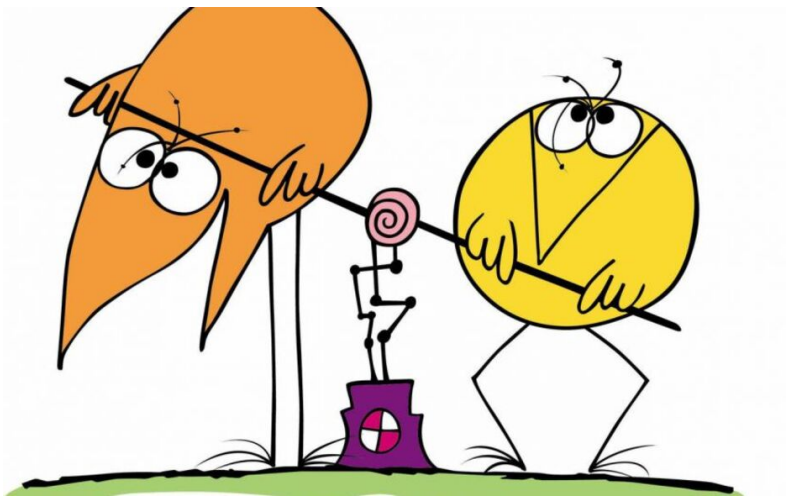
Année 2024-2025

Sommaire

Méthodologie de gestion de projet

Gestion de projet : quésaco ?

- organiser le bon déroulement d'un projet
- en atteindre les objectifs visés en temps et en heures



Méthodologie de gestion de projet

Gestion de projet : quésaco ?

- organiser le bon déroulement d'un projet
- en atteindre les objectifs visés en temps et en heures

Utiliser...

- des **méthodes** de gestion de projet
- des **techniques** (bonnes pratiques)
- des **outils**

Il existe de nombreuses méthodes, outils et techniques !

➜ Ce qui est présenté ici est juste une introduction pour le cours PCII, avec quelques outils basiques !

Méthodologie de gestion de projet

À quoi cela sert-il ?

- Un projet n'est pas un exercice/TP !
  - objectifs à définir soi-même
  - solution non connue à l'avance (faisabilité)
  - organiser le temps (réalisation)
  - définir des mesures/tests (objectif)
- Des outils adaptés pour réussir

. . .mais aussi des savoir-faire !

- Difficulté supplémentaire en PCII : travail en équipe !

Méthodologie de gestion de projet

Quelques méthodes (en informatique)

- PERT (Program Evaluation and Review Technology) (1958)
- Cascade (*waterfall* ou « cycle en V ») (années 1980)
- Approche agile (années 2000)

Quelques techniques

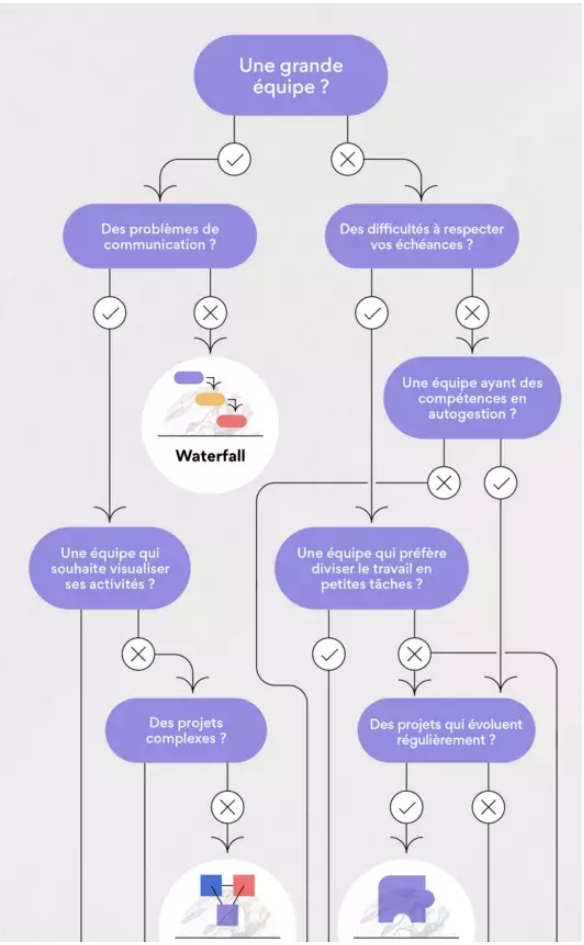
- Techniques centrées sur l'activité (définir, inventorier, planifier, gérer...)
- Jalonnement/planification
- Découpage du projet (en sous-projet)

Quelques outils

- Diagramme de Gantt, diagrammes UML
- Systèmes de répartition de tâches (tickets GLPI, Trello...)
- Scrum (pour la méthodologie Agile)

Méthodologie de gestion de projet

Comment choisir ?



copyright Asana

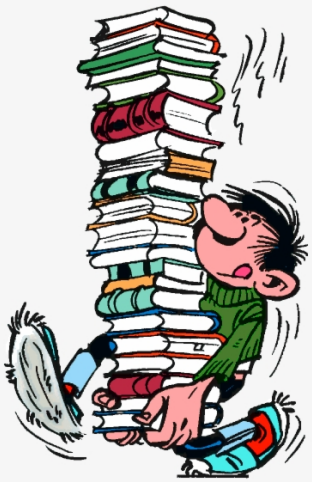
Il existe des méthodes pour choisir la bonne méthode !

Méthodologie de gestion de projet

Concrètement (en PCII)

- Méthode : **cascade**
- Techniques : **planification**
- Outils : **diagrammes**

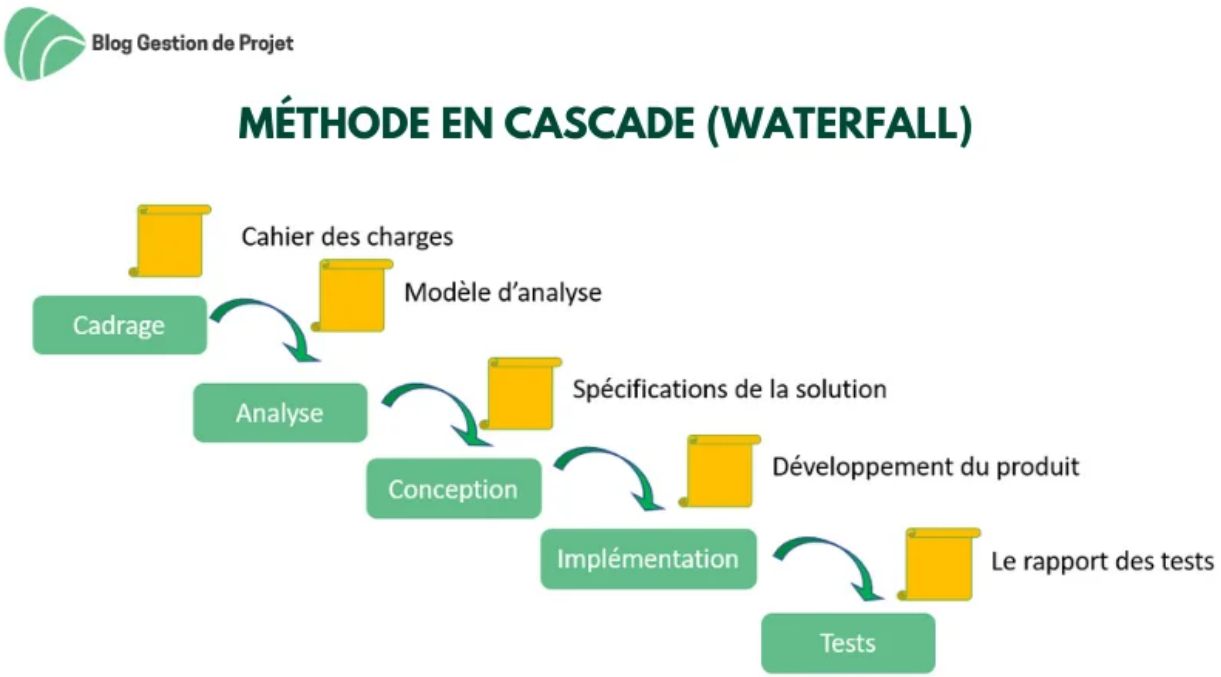
Et surtout...



➜ Nous allons produire des documents !

Méthode en cascade

Principe

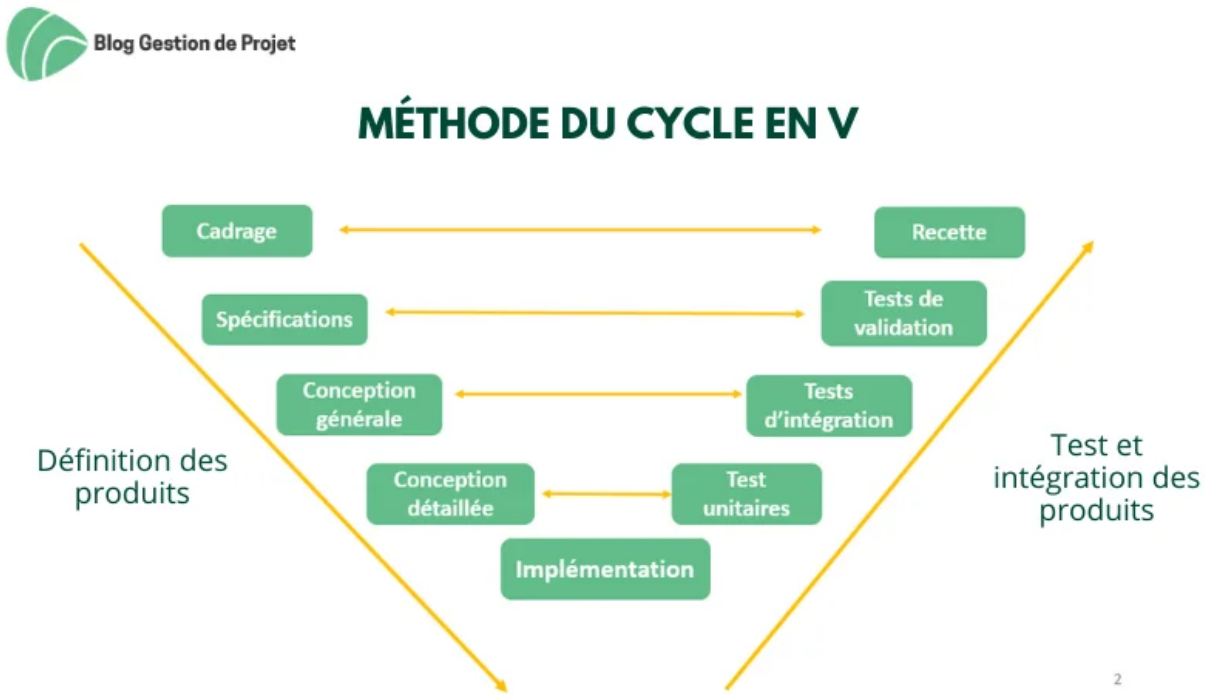


source : `blog-gestion-de-projet.com`

➜ Chaque étape donne lieu à un document spécifique !

Cycle en V

Une variante de la cascade...



source : `blog-gestion-de-projet.com`

Le document d'analyse

Objectif

Définir les fonctionnalités à réaliser

Contenu

- Lister les fonctionnalités à réaliser
- Pour chaque fonctionnalité, identifier :
  - le niveau de difficulté ou le coût (ex : score 1 à 3)
  - le niveau de priorité (ex : score 1 à 5)

En pratique

On peut découper en :

- Analyse globale : identification de fonctionnalités très générales
- Analyse détaillée : sous-fonctionnalités précises pour chaque fonctionnalité générale

Le document d'analyse

Attention !

Ne pas parler de la conception dans le doc d'analyse !

Ce qu'on peut mettre

- ✓ Des « fausses » captures d'écran (ou des dessins)
- ✓ Des questions ouvertes

Ce qu'on doit **pas** mettre

- ✗ Un patron de conception (voir plus loin)
- ✗ Des classes à implémenter
- ✗ Un langage de programmation ou une plate-forme
- ✗ Une architecture logicielle

➜ on reste au niveau des fonctionnalités !

Le plan de développement

Objectif

Définir le calendrier et les jalons du projet

Contenu

- Chaque fonctionnalité → plusieurs tâches avec durées (en p.m)
  - Recherche de la solution (ex : lecture de doc)
  - Conception (écriture d'algorihmes)
  - Développement (programmation) et tests « unitaires »
- Identifier les autres tâches liées au projet

  (temps de réunion d'équipe, temps de rédaction de la documentation, rencontres avec le client...)

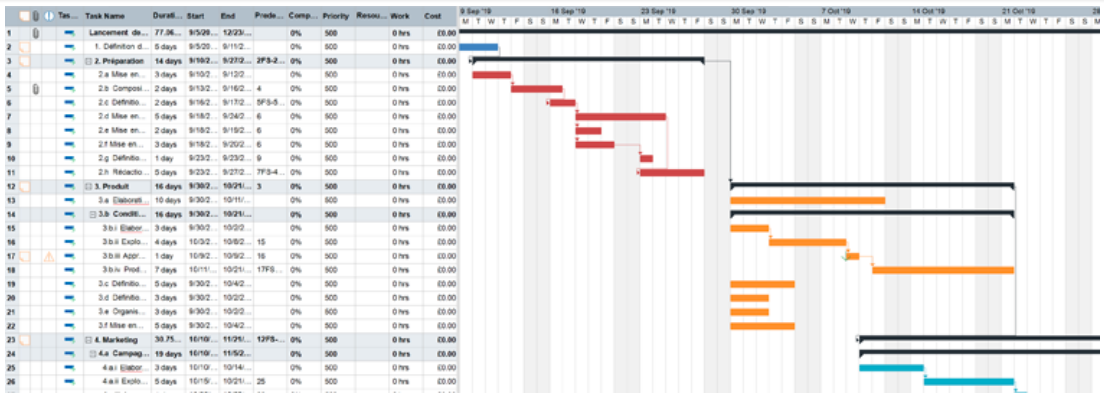
Diagramme de Gantt

Le plan de développement se termine toujours par un calendrier !

Diagramme de Gantt

Principe

Visualisation du déroulé dans le temps



source : `gantt.com`

- Chaque **ligne** est une tâche du projet
- Chaque **colonne** est une unité de temps/date
- On indique les périodes des tâches
- On indique éventuellement des dépendances
- (optionnel) On indique qui fait quelle tâche quand

Le document de conception

Objectif

Décrire la réalisation de la solution

Identifier les éléments concrets de solution

➜ Entre l'analyse et l'implémentation

une équipe de développeurs doit pouvoir programmer la solution à partir du seul document de conception

Deux parties

- Conception générale
- Conception détaillée

Le document de conception

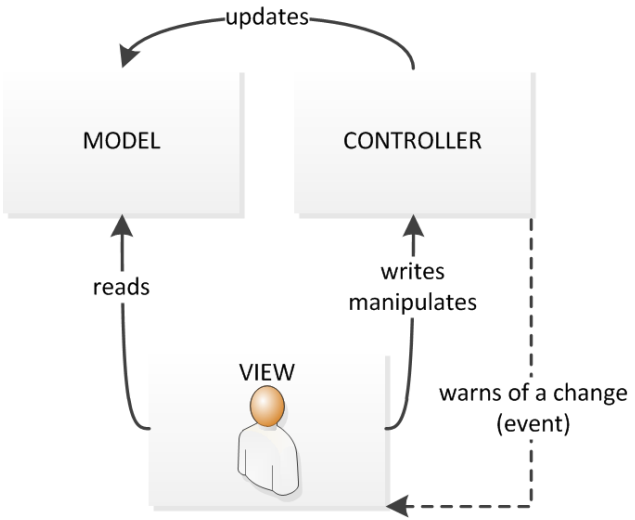
Conception générale

- Décrire les blocs fonctionnels et comment l'information circule entre les blocs
- Exemple : le patron de conception MVC

Le patron MVC

Modèle, Vue, Controller

- Patron de conception (*design pattern*)
  - Outil du génie logiciel pour structurer son code
  - Séparer les problèmes et les fonctionnalités
  - ➜ Ne pas se retrouver avec un ensemble de classes anarchique où l'on ne sait plus qui fait quoi ni comment...
- Séparer le code en trois parties (M, V et C)



source : Wikipedia

Le patron MVC

Trois parties

- Le **modèle**
  Ensemble des données qui caractérisent l'état du système
  ex : la position de l'ovale
- La **vue**
  Rendu visuel des données (à partir du modèle)
  ex : le JPanel pour dessiner l'ovale
- Le **contrôleur**
  Lien entre l'interface graphique et le modèle
  - Capture les événements dans l'interface et modifie le modèle
  - Informe la vue de changements

  ex : le listener sur les clics souris

Le document de conception

Conception générale

- Décrire les blocs fonctionnels et comment l'information circule entre les blocs
- Exemple : le patron de conception MVC

Objectif

Comprendre l'architecture générale du projet, le lien entre les grandes fonctionnalités

Attention

- Un peu de texte est toujours le bienvenu
- Pas de diagramme de classe ici !

➜ Cela va se traduire par des packages !

Le document de conception

Conception détaillée

- Faire une section pour **chaque fonctionnalité**
- Décrire précisément la solution pour implémenter cette fonctionnalité :
  - Principales structures de données utilisées
  - Attributs et constantes définies
  - Algorithme éventuel

    Attention : pas de code source !
  - Conditions limites (et donc tests)
  - Utilisation par les autres fonctionnalités

Diagramme de classe

- Faire un diagramme par fonctionnalité
- Simplifier : donner uniquement les classes utiles pour cette fonctionnalité

Les autres éléments d'un document d'analyse et conception

Prévoir aussi

- Une courte introduction
- Une courte conclusion (pour le prof)
- Une section « résultats » avec des copies d'écran
- La documentation utilisateur (**indispensable !**)

  ➜ comment installer et utiliser le programme
- La documentation développeur

Mise en pratique

Nous allons préparer ensemble la documentation du projet pour le projet tutoré et la première séance

1. Introduction = Cahier des Charges

1. Introduction = Cahier des Charges

- Nous voulons réaliser un mini-jeu inspiré de Ketchapp Circle
- Un ovale se déplace le long d'une ligne brisée générée aléatoirement
- Le but du jeu est d'éviter de sortir de la ligne / de capturer des objets
- Le joueur peut cliquer sur l'écran pour faire monter l'ovale, qui redescend ensuite tout seul
- Un dessin de l'interface graphique du jeu (ou une capture d'écran de KC)

2. Analyse globale

2. Analyse globale

- Fonctionnalités principales :
  1. interface graphique avec l'ovale et la ligne brisée
  2. contrôle du mouvement de l'ovale
  3. génération et défilement automatique de la ligne brisée
  4. menu du jeu (optionnel)
  5. système de score (gameplay)

2bis. Analyse détaillée

2bis. Analyse détaillée

- Pour la séance 1 (fonctionnalités 1 et 2, sans ligne brisée)
  1. dessiner un ovale dans une fenêtre graphique
     → difficulté basse, priorité 1
  2. faire monter l'ovale lorsqu'on clique
     → difficulté moyenne, priorité 1
  3. faire descendre l'ovale lorsqu'on ne clique pas
     → difficulté basse, priorité 1
  4. utiliser une vitesse pour créer un mouvement de saut
     → difficulté moyenne, priorité 2

Attention

Ne surtout pas organiser les fonctionnalités par séance de cours !

3. Plan de développement

3. Plan de développement

- 5 tâches :
  - apprentissage de Swing : 1h
  - apprentissage des Threads : 30mn
  - dessin de l'ovale : 15mn
  - événements souris : 15mn
  - thread de mouvement : 15mn
  - rédaction du document analyse et conception : 1h
- 1 seule ressource (vous)
- Diagramme de Gantt

Attention

Le faire sur l'ensemble du projet, pas juste sur la séance !

3. Plan de développement

Diagramme de Gantt

	14h00	14h30	15h00	15h30	16h00	16h30	17h00	17h30
0. Recherche								
Swing								
Threads								
1. Vue								
Dessin Ovale								
3. Contrôleur								
Souris								
Thread mouvement								
4. Documentation								
Rédaction doc.								

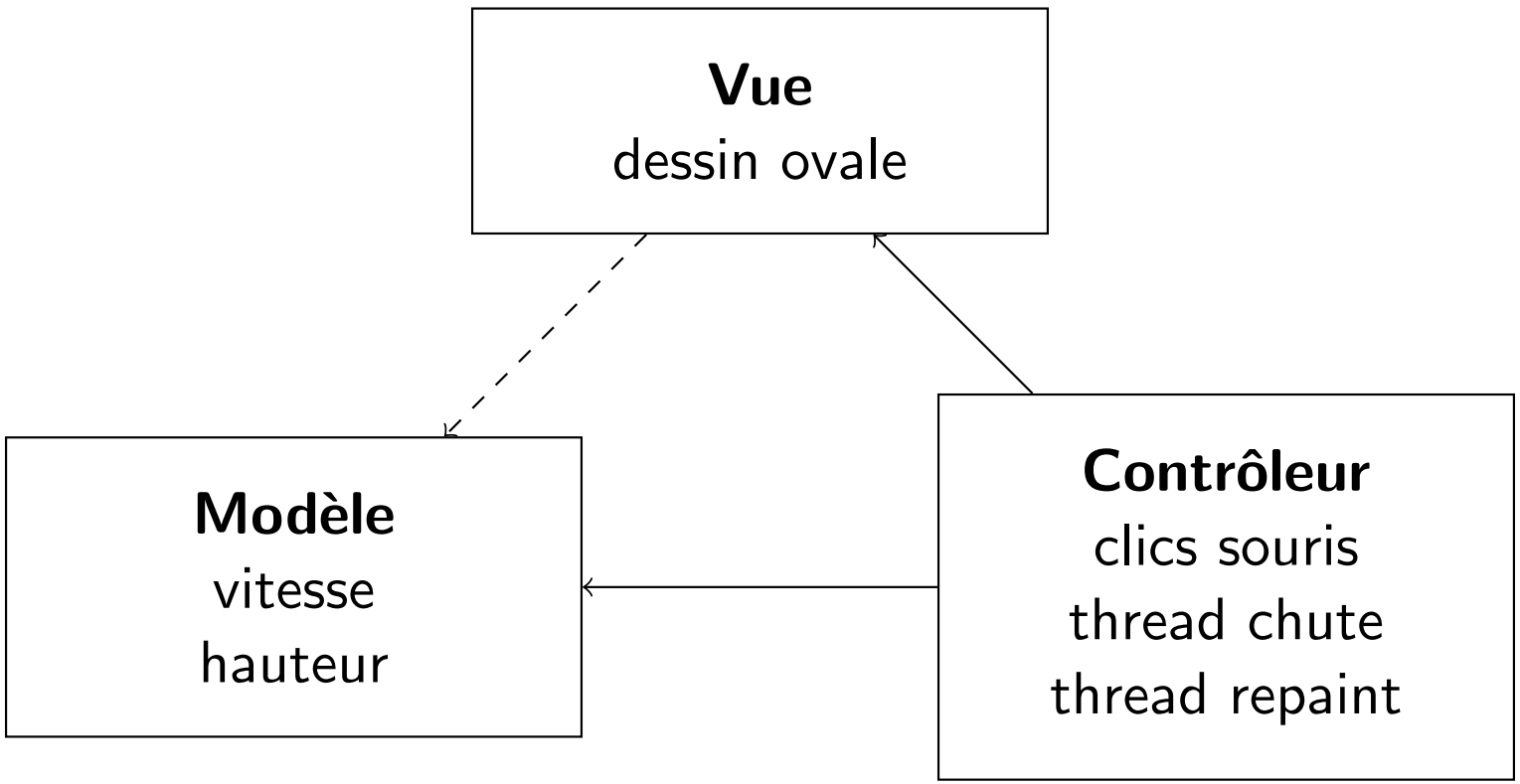
Attention

Le faire sur l'ensemble du projet, pas juste sur la séance !

4. Conception générale

4. Conception générale

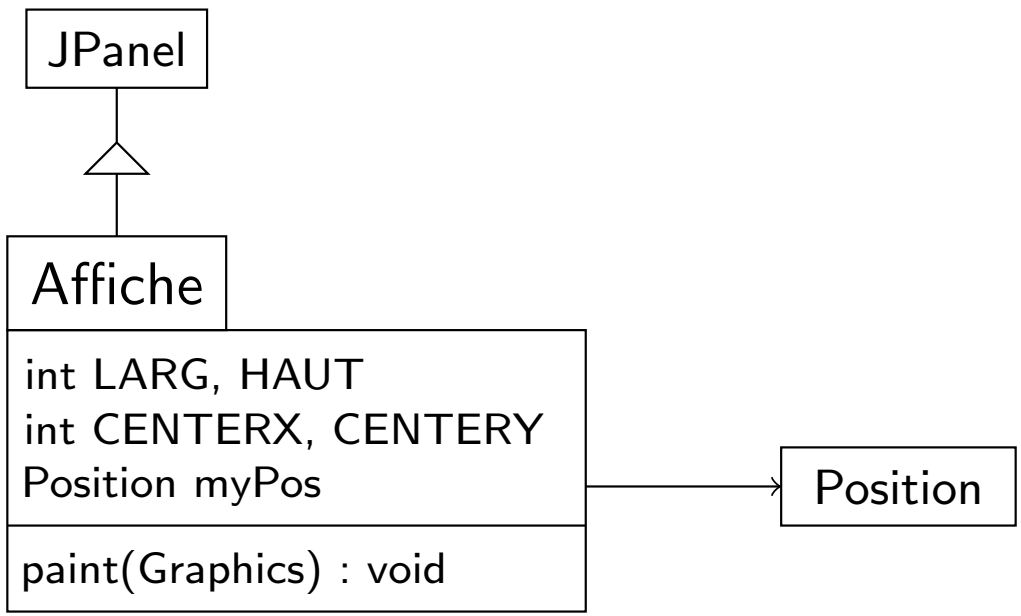
- Le patron MVC !
- Reproduire un schéma qui montre comment vos fonctionnalités rentrent dans ce motif



4.1 Conception détaillée : Dessin de l'ovale

4.1 Conception détaillée : Dessin de l'ovale

- Utilisation de l'API Swing et de la classe JPanel
- Définition des dimensions de l'ovale et de la fenêtre
  $\rightarrow$ lister les constantes
- Un diagramme de classe simplifié :



4.2 Conception détaillée : Clic souris

4.2 Conception détaillée : Clic souris

- Programmation événementielle avec la classe `MouseListener`
- Hauteur définie dans une constante
- Thread de mise à jour de l'affichage
  Délai 50ms dans une constante

Diagramme de classe

4.3 Conception détaillée : Chute de l'ovale

4.3 Conception détaillée : Chute de l'ovale

- Thread (et constante pour la vitesse)
- Hauteur définie dans une constante ?

Diagramme de classe

Attention

Nous n'avons pas eu besoin d'écrire un algorithme pour l'instant...

Cela viendra bientôt...

5. Résultats

5. Résultats

- Une ou deux copies d'écran

6. Documentation utilisateur

6. Documentation utilisateur

- Prérequis : Java avec un IDE
  (ou Java tout seul si vous avez fait un export en `.jar` exécutable)
- Mode d'emploi (cas IDE) : Importez le projet dans votre IDE, sélectionnez la classe Main à la racine du projet puis cliquez sur l'icône « Run ».
- Mode d'emploi (cas `.jar` exécutable) : double-cliquez sur l'icône du fichier `.jar`
- Cliquez sur la fenêtre pour faire monter l'ovale.

7. Documentation développeur

7. Documentation développeur

- La classe principale
- Les packages
- Où sont définies les constantes
- Prochaine étape : la génération de la ligne brisée
  - Dans le package model : nouvelle classe
  - Dans la vue : modification de la classe `Affichage`

8. Conclusion et perspectives

8. Conclusion et perspectives

- J'ai vraiment fait du super boulot
- J'ai eu du mal avec le bord de l'écran mais j'ai pu le résoudre grâce à un test if dans mon modèle.
- J'aurais bien aimé implémenter un rond multicolore

Avant de nous quitter...

Travail de la semaine

- Rédiger **votre** doc du projet (à partir des notes du cours)
- Créer votre compte GitHub Copilot (délai validation)
- Répartir les groupes de TP (voir les liens sur la page du cours)

Bonne semaine !