Ce projet a été réalisé par le groupe 99:

NASR Amira ,NE:2196593l,
mail: nasramira8@gmail.com

HAMITOUCHE Boukhalfa ,NE:21962465,
mail: hamitoucheboukhalfa11@gmail.com

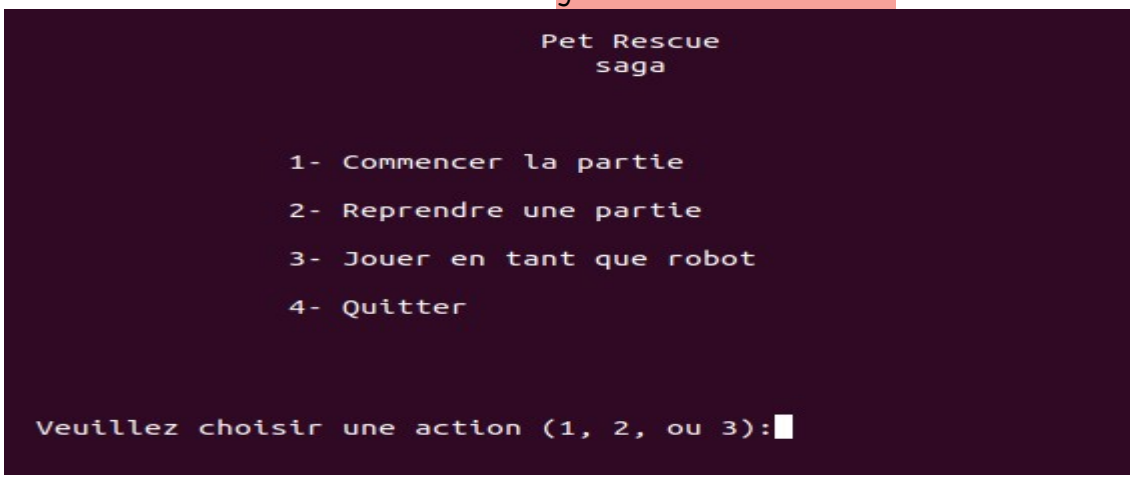
Présentation du projet :

Notre projet est basé sur le jeu "Pet Rescue Saga".Sur l'image ci-dessous vous pouvez observer le plateau du jeu (niveau 3 ici).Tout ce qu'il faut faire pour franchir une étape dans ce jeu c'est d'éliminer le maximum de briques voisines qui ont la même couleur en un minimum de coups afin de ramener les animaux perchés au sommets sains et saufs en bas. Pour ce faire, le joueur doit réaliser des associations de cubes de même couleur afin de les éliminer sans dépasser le nombre de mouvements autorisés pour chaque niveau(**niveau 1: 6 mouvements, niveau 2: 7 mouvements, niveau 3: 10 mouvements**) et si le joueur bloque il peut utiliser des boosters (un fusil pour éliminer tout l'ensemble d'une colonne sélectionnée et un marteau pour détruire une seule case sélectionnée), dans notre version un nouveau joueur qui commence la partie dispose d'un seul booster Fusil et deux boosters Marteau et à chaque fois qu'il gagne une partie il gagne un fusil et un marteau en plus.

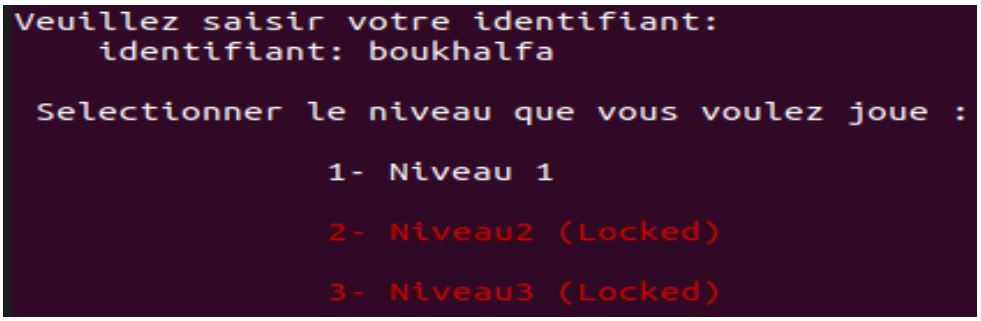
Notre version de jeu peut se jouer soit en Terminal (textuellement) soit graphiquement, le jeu contient 3 niveaux (seul le niveau 1 est accessible pour une première fois, un niveau n se déverrouillera une fois le niveau n-1 est gagné), le joueur a également la possibilité de sauvegarder une partie à tout moment et la reprendre plus tard .

I. L'interface textuelle:

-Elle se lance avec la commande: java JeuTerminal :--



Comme l'image ci-dessus l'illustre, un joueur peut commencer une nouvelle partie ou reprendre sa dernière partie sauvegardée, il peut même choisir "jouer en tant que robot" pour voir une partie (d'un niveau choisi aléatoirement) jouer par l'ordinateur pour qu'il voit bien comment le jeu se déroule.
Une fois il a choisis de commencer une nouvelle partie, il doit saisir son identifiant de jeu et puis choisir un niveau pour jouer(les niveau accessible), par exemple pour un nouveau joueur :



-En jouant le **niveau 3** par exemple (Le cas d'un joueur qui a déjà gagné les deux premiers niveaux) on aura cet affichage:

pour pouvoir afficher ces figures et ces couleurs dans le terminal on a utilisé des unicode tel que : "`\uD83D\uDC08`" pour afficher l'emoji du **chat** et l'utiliser comme notre **animal** et "`\u001B[31m`" pour afficher un text en rouge, ...etc.

Dans ce niveau, on voit qu'on a le droit que à 10 mouvements(si on les depasse avant de sauver tous les animaux on perd).
Le joueur peut sélectionner les coordonnées de la case qu'il veut éliminer (1) ou utiliser un fusil pour éliminer une colonne entière(s'il en possède) (2)ou utiliser un marteau pour détruire une seule case de son choix (s'il en possède)(3) ou enregistrer la partie ou quitter le jeu (4) et de retourner au menu sinon(5).

La partie se termine par une **victoire** si le joueur a pu sauvé tous les animaux sans dépasser le nombre de mouvements autorisés et sinon la partie s'achève avec sa **défaite** s'il a plus de mouvements ou si la situation du plateau est bloqué (pas de cases simplifiables et pas de boosters).

Pour réaliser cette interface textuelle , on a du utiliser les classes suivantes :

Case.java: Une classe **abstraite** qui sert comme base de notre héritage et elle a CaseNormal.java(pour créér des cases colorées), CaseAnimal.java(pour créér des animaux) et CaseObstacle.java (pour créér des obstacles).

JeuTerminal.java: c'est la classe qui nous permet de lancer le jeu et d'afficher notre menu du jeu.

Joueur.java: c'est la classe qui stockent toutes les informations d'un joueur.

Niveau.java: c'est la classe qui gère les niveaux avec des méthodes de stockage de chaque niveau dans un fichier(sérialisation) et également des méthodes de désérialisation.

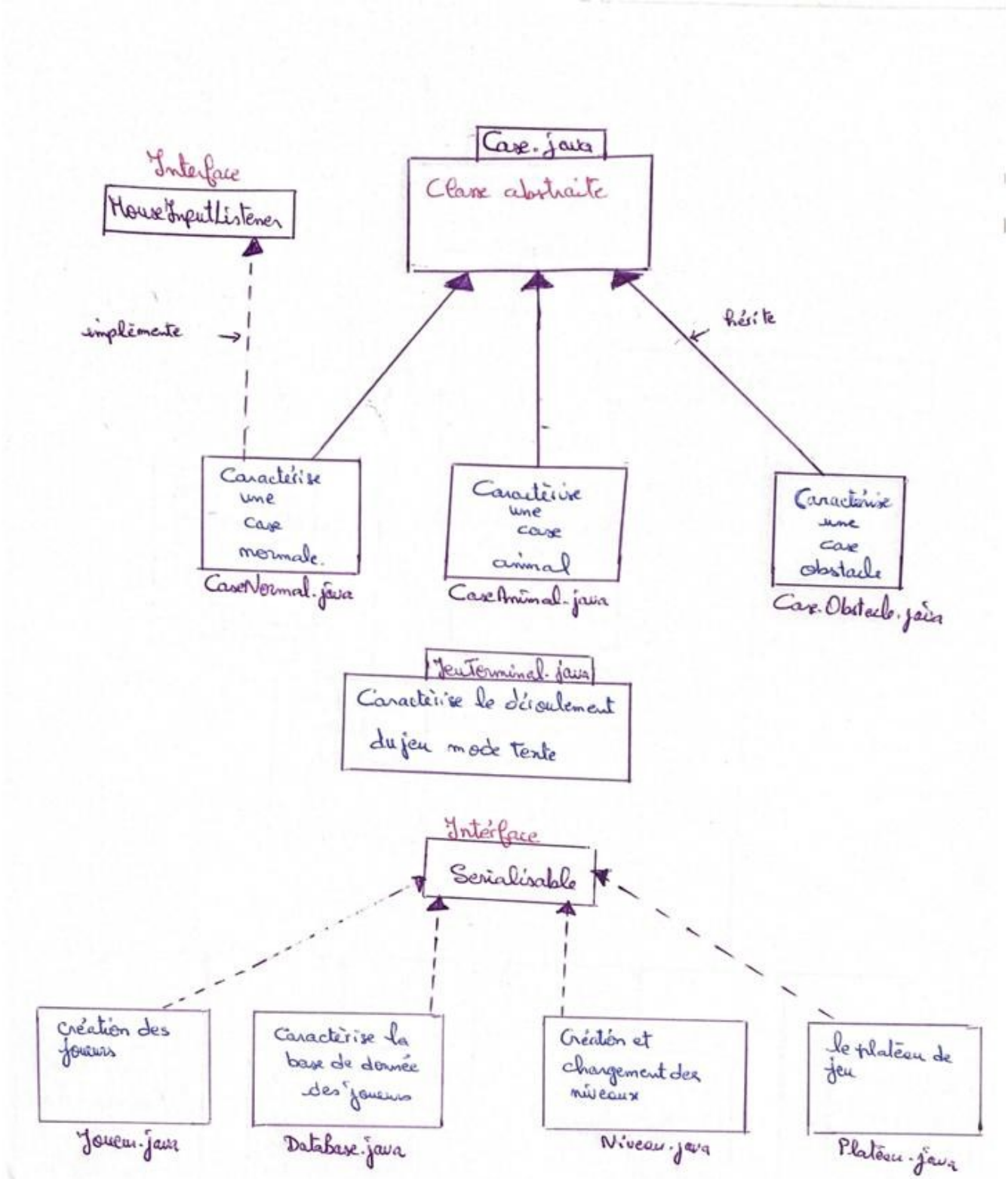
Plateau.java: C'est notre classe principale, c'est la ou toute la logique du jeu est définie avec les méthodes d'éliminations de cases et d'organisation du plateau, de victoire, de défaite, ...etc.

DataBase.java: C'est la classe qui nous sert comme une base de données et qui enregistre une linkedList de nos joueurs dans un fichier.

Le détail complet de nos classes est expliqué dans le fichier schemasClassesProjet.pdf

Voici: une photo manuscrite de notre schema des classes pour cette interface:

Pet Rescue Saga



I. II- L'interface graphique:

Elle se lance avec la commande: java Jeu &

Comme l'immage ci-dessus le montre, le joueur a les mêmes fonctionnalitées que celles de l'interface textuelle (à part l'option de jouer en tant que robot).

En Commençant la partie:

on nous demande de saisir notre **identifiant** et puis cliquer sur **le bouton entrer du clavier** pour débloquer **le bouton entrer de l'écran** (on a fait ça

pour ne pas accepter des champs vide), une fois entré on nous propose les niveaux accessibles (seulement le premier niveau pour un nouveau joueur) :

On choisit le niveau 1 :

Et on peut commencer à jouer ! Avec les mêmes règles que dans l'interface textuelle.

Pour réaliser cette interface on a du utilisé les classes suivantes en plus des classes utilisées dans l'initerface textuelle:

PlateauGraph.java: c'est la classe ou on a gérer l'affichage de notre plateau (cases, score, animaux sauvés, ...),elle utilise la classe plateau.java pour reprendre les mèthodes de la logique du jeu (élimination, réorganisation, ...)

Victoire.java : c'est pour afficher une page de victoire avec des boutons qui demande soit de rejouer soit de retourner au menu, ...

Defaite.java: c'est pour afficher une page de défaite avec des boutons qui demande soit de rejouer soit de retourner au menu.

Reprendre.java: c'est pour afficher la page ou il ya la sauvegarde du joueur qui l'a lancer (si il y'en a)

NiveauGraph.java: c'est la classe qui gère la construction et le chargement d'un niveau (elle utilise la classe niveau.java)

Jeu.java : c'est la classe qui nous permet de lancer le jeu graphiquement

Accueil2Graph.java: C'est la classe qui caractérise la fenetre qui demande l'identifiant du joueur(soit pour commencer soit pour reprendre une partie).

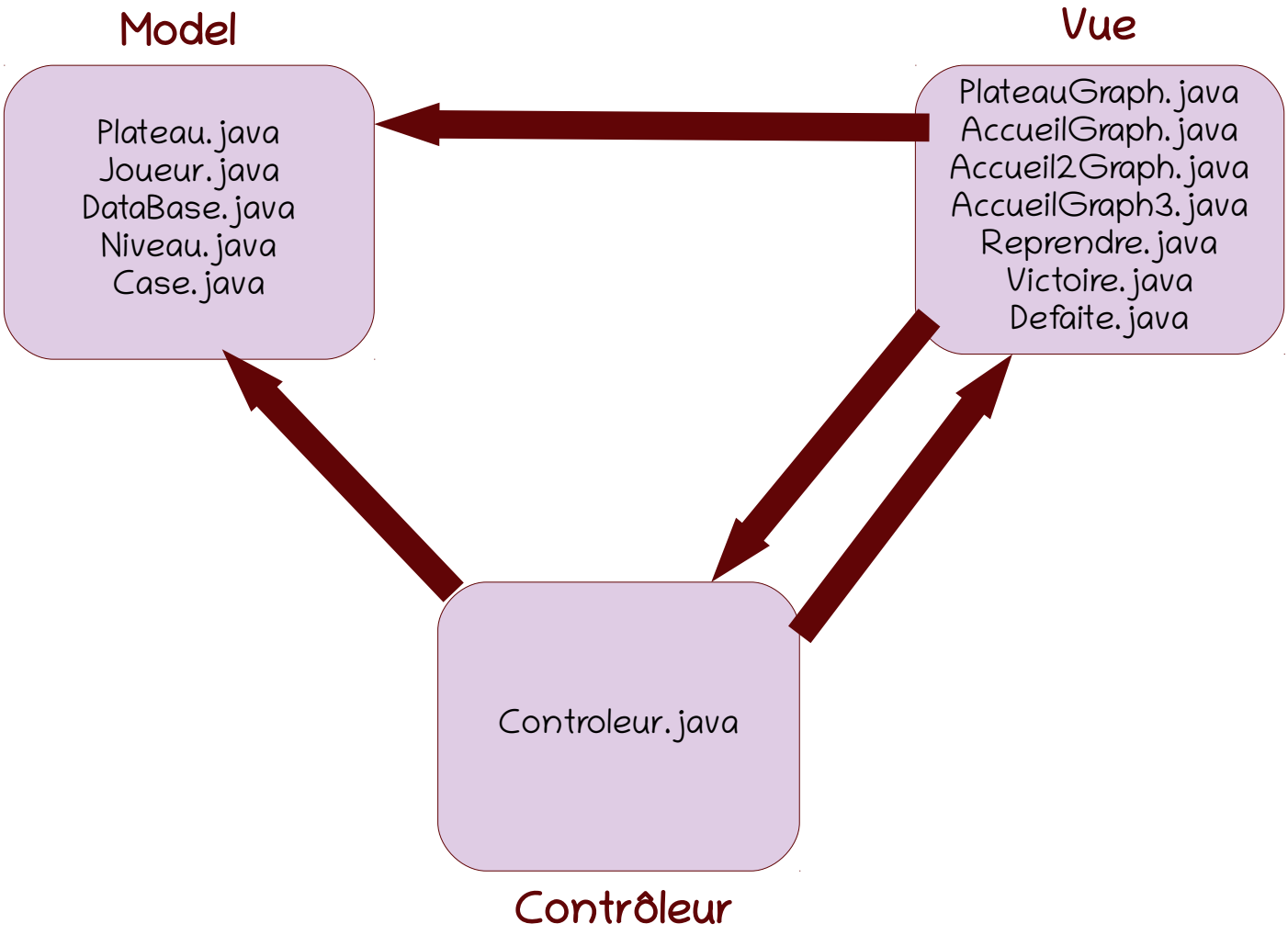
AccueilGraph.java: C'est la classe qui définit la fenetre d'accueil qui se lance dès le lancement du jeu et qui propose au joueur soit de commencer, reprendre la partie, de l'aide, ou quitter le jeu.

AccueilGraph3.java: c'est la classe qui caractérise la fenetre pour choisir un niveau à jouer

ModelAccueil.java: sert comme model pour toutes les fenetres du jeu (un setter pour l'image du fond)

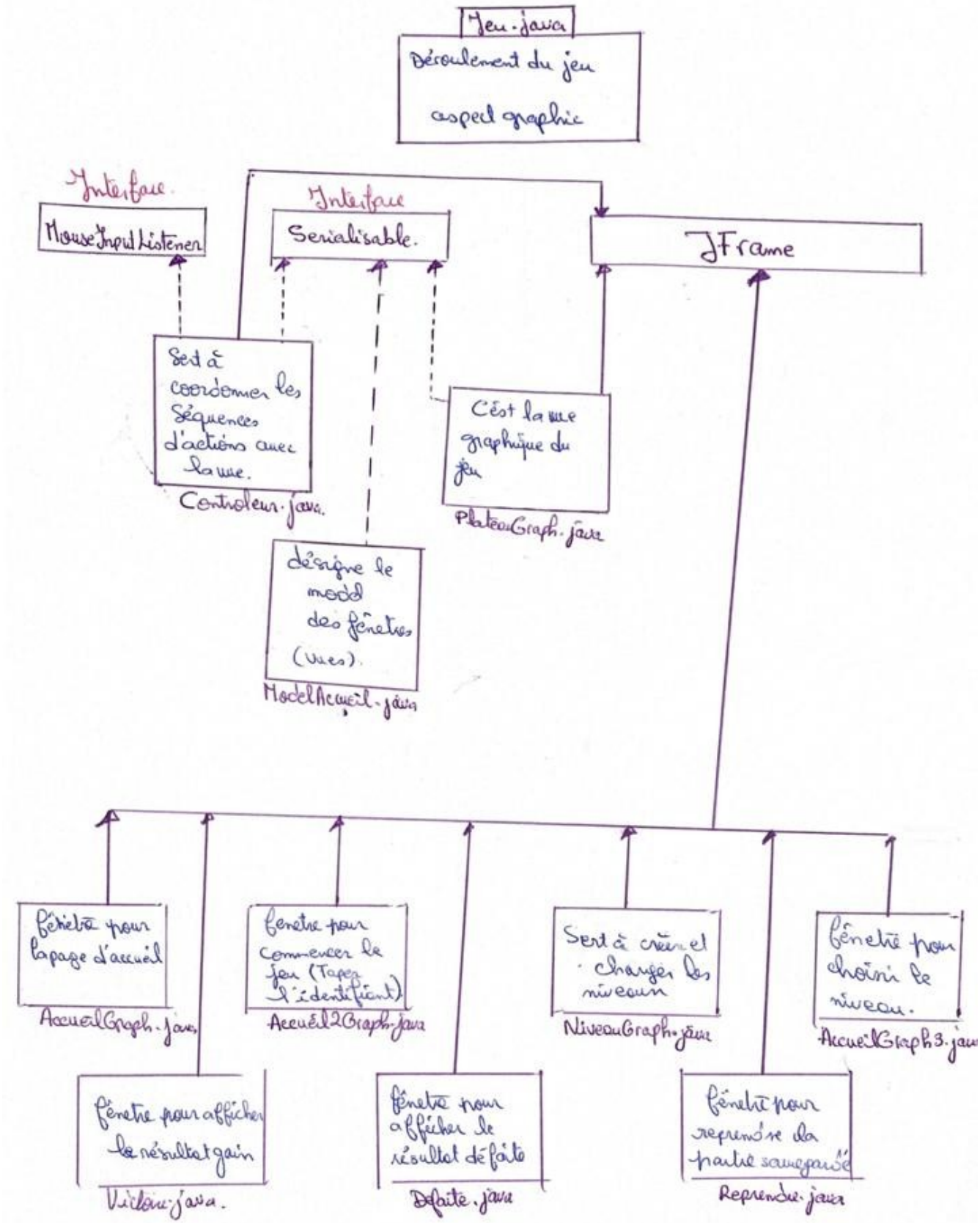
Controleur.java : Sert à coordonner la séquance d'actions /réaction Des componment (boutons et textfiels) avec la vue.

↳ Pour la répartition des classes en Model-Vue-Controleur, On a :



Voici une photo manuscrite de notre schéma des classes de cette interfaces:

Pet Rescue Saga



Les parties traitées:

Les deux interfaces (**textuelle et graphique**) sont implémentées.
Toutes les parties sont traitées sauf le joueur robot dans l'interface graphique.

Les problèmes connus:

- la sauvegarde d'une partie dans l'interface graphique(problème du à la non sérialisation de la classe BufferedImage), pour le premier coup on a ajouté le modificateur «transient» pour ne pas prendre en considération l'image dans la sauvegarde et après ça on avait bien la sauvegarde qui fonctionnait mais on avait des problème de reprendre la partie sauvegardé et la jouer correctement à cause de l'absence des images non sauvegardées, mais on a réglé ça en recréant un autre plateau à partir du notre et puis sauvegarder ce dernier et bien évidemment enlever le modifieur «transient» mais à force de jouer une partie qu'on a repris on a des exceptions qui s'affichent (BufferedImage est non serializable) mais tout fonctionne bien..

  Lorsqu'on clique sur nos boosters dans l'interface graphique si on en a plus on obtient un message dialogue qui ne se ferme qu'au bout de plusieurs tentatives :) mais on a réglé ça en supprimant toutes les actions listners du bouton avant d'ajouter une nouvelle.

- Lors de l'ajout du son, on pourrait bien jouer avec mais au bout d'un moment le son devient désagréable et d'après une recherche on a trouvé que c'est à cause des fichiers «.wav» et donc on a laissé la ligne qui fait appel à la fonction du son en commentaire.

Les pistes d'extensions qu'on a pas implémenté.

- Implémenter un éditeur de niveau (sommaire) :
  Proposer la possibilité de revenir en arrière en annulant le dernier coup et/ou de demander de l'aide en obtenant en retour une zone intéressante à jouer.
- Le joueur robot dans l'interface graphique.
- Revenir en arrière pour annuler le dernier coup
- Demander de l'aide en obtenant en retour une zone intéressante à jouer