J’ai créé un nouveau dossier *Projet* car on a décidé de repartir sans CTAT.

# Général

Le sudoku se compose d’instance d’éléments Case

# Interface (Gilles)

L’interface restera celle établie par Gilles.

Il faudra gérer l’affichage d’un sudoku comme suit :

*Case* sudoku[][] = new *Case*[9][9] ;

Les Case se compose d’un chiffre principal (=0 si la case est blanche) et d’un table de 9 bool (*true* si la chiffre peut être dans cette case)

* Voir le fichier Case.java

On affichera donc par case le chiffre se trouvant dans la case OU l’ensemble des chiffre hypothétique contenu dans le tableau.

Le top serait d’établir plusieurs fonctions au clic gauche de la souris :

* Un mode crayon dans laquelle l’utilisateur peut cliquer sur des boutons représentant les hypothèses à l’intérieur des cases,
* Un mode stylo pour lorsque l’on clique sur une case, on puisse rentrer un chiffre à valider,
* Un mode gum pour effacer le chiffre et revenir à l’affichage des hypothèses.

Gilles vois si c’est jouable.

# Tuteur (Jules)

Le tuteur réagira dans 3 cas avec des phrases toute faites

1. Le joueur entre une réponse fausse
   * On informe le joueur qu’il a faux,
   * On lui indique la règle violé
2. Le joueur rentre une réponse incertaine (potentiellement bonne mais n’appartenant à aucune stratégies)
   * On lui dit de faire attention,
   * On lui dit que c’est un imprudent,
   * On lui dit qu’il peut résoudre le sudoku sans recourir à l’aléatoire (car on se concentre sur les sudoku faciles pour le moment
3. Le joueur ne joue pas pendant un moment
   * On regarde les coups possibles et on le renseigne dans la région où il peut jouer ?
   * Ou on l’informe sur la stratégie à appliquer (genre *essayes de repérer dans tes hypothèse les couples de chiffre présents dans une même région*)

# Stratégies (Maxime)

Jules, à toi !!!

//Desciption des stratégies

On utilise Jess mettre à jours les possibilités et tester la violation des règles

Fonctionnalités :

* A partir de l’état actuel de la matrice, il nous renvoie une liste de coups possibles en appliquant la stratégie 1 (on verra pour la 2 après),

*Il faut voir si tu préfères stocker l’état de la matrice en interne ou si tu veux qu’on te la passe en argument*

* Faut que tu nous dise le type de retour que tu veux renvoyer (celui qui te facilite la vie, on fera avec ;) )
* A partir d’une action faite par l’utilisateur (un chiffre est rentré), Jess nous informe si la réponse viole les règles,
  + Si oui, c’est tout, Java fera le reste,
  + Si non bah c’est tout aussi le joueur a joué aléatoirement mais n’a pas faux (au tuteur de prendre le relais,

*Pareil ici à toi de voir si on te passe juste l’action faite par l’utilisateur pour mettre ta matrice interne à jour ou si tu prèfère qu’on te passe la nouvelle matrice potentiel en argument*

* Ici, renvoie juste un bool

# Apprenant (Ludovic)

Dépend des stratégies mais voilà les fonctions prévues :

* Stockage de la liste des connaissances,
* Stockage de l’avancement dans l’apprentissage de chacune d’entre elles,
* Stockage de l’utilisation de l’aléatoire.
* Stockage de la maitrise des stratégies.

# Boucle de jeu

Voilà comment se déroule une boucle de jeu :

* Au joueur de jouer,
  + S’il met trop de temps on envoie un message à l’aide du tuteur,
* Le joueur modifie ces hypothèses
  + On verra si on a le temps de le prendre en compte,
* Le joueur modifie le chiffre d’une case
  + On test si l’action appartient au actions de la stratégie 1 ou 2,
    - Si oui, on met à jour :
      * la matrice de l’état actuel du sudoku,
      * l’affichage,
      * L’apprentissage des connaissances (en fonction de la stratégie utilisé)
    - Si non,
      * On teste avec Jess
        + s’il a fait une erreur

Le tuteur lui dit son erreur,

On met à jour l’apprentissage des connaissances (en fonction de la règle violé)

On empêche la mise à jour de l’affichage ?

* + - * + S’il n’a pas faux

Le tuteur lui dit qu’il a pas besoin de répondre au hasard,

On met à jour l’utilisation du hasard (apprentissage)

On l’autorise à continuer ?

* On met à jour la liste des coups possibles par stratégie,
* On attend que le joueur joue.