M1 BIMS
Génie Logiciel
TPs

Génie Logiciel
Git et tests unitaires

Exercice 1

- Se créer un compte sur github : https://github.com
- Donner l'identifiant pour être ajouter au projet
- Créer un répertoire pour le projet Game Of Life
- Cloner le dépôt suivant :
  - https://github.com/gastoben/M1-conway.git
  - Contenu : fichier `Grid2d.py` contenant la classe `Grid2d`
- Créer une branche de développement en local à votre nom (format `M1_<prenom><nom>`)
- Tous le reste des travaux se fera dans cette branche
- Ne pas oublier d'ajouter les fichiers à l'Index et de valider le HEAD pas de push pour le moment
- Les versions stables devront être étiquetées.
- Créer la branche `M1_<prenom><nom>` sur le serveur distant
- Rattacher votre branche locale avec votre branche distante

Jeu de la Vie Implémentation

- Soit la classe Grid2d qui implémente la gestion mémoire
- Outre les méthodes usuelles (constructeurs, …) cette classe présente les méthodes suivantes :

`get_dim()`
  - Retourne les dimensions de la grille

`get_value_at(i,j)`
  - retourne la valeur à l'emplacement `i,j` ou –1 si l'emplacement est inexistant

`set_value_at(i,j,value)`
  - affecte la valeur `value` à l'emplacement `i,j` et retourne 0 si l'affectation réussie et -1 sinon

`get_neighs_value(i,j)`
  - qui retourne la liste des valeurs des voisins de `i,j` dans la grille

Jeu de la vie Implémentation

On souhaite développer la classe `ComputeKernel` du moteur de calcul

Cette classe aura pour attribut deux objets Grid2d : `current_grid, next_grid`

Les principales méthodes à développer seront les suivantes :

- `compute_rules(neighs_value,local_value)`
  - Entrées :
    - `neighs_value` la liste des valeurs des voisins
    - `local_value` la valeur de la cellule
  - Retourne la valeur du prochain état de la cellule
- `apply_rules(i,j)`
  - Affecte la valeur de la celle `(i,j)` de `next_grid` en appliquant les règles du jeu de la vie à partir de la grille `current_grid`
- `compute_gol()`
  - Affecte à `next_grid` l'application d'une itération du jeu de la vie à `current_grid`
  - Donne à `current_grid` la valeur de `next_grid`

Exercice 2

- Créer le fichier `ComputeKernel.py`
- Développement de la méthode `compute_rules`
  1. Proposer et coder des tests unitaires pour cette méthode
  2. Développer la méthode en question
  3. Jouer les tests unitaires
- *Facultatif* Développer la méthode `apply_rules`

Exercice 3

- Faire la couverture de code sur vos tests unitaires de la méthode `compute_rules`
- Faire l'analyse de la couverture de code et identifier les éventuels tests manquants

Exercice 4

- Développer la méthode `compute_gol()` qui
  - Affecte à `next_grid` l'application d'une itération du jeu de la vie à `current_grid`
  - Donne à `current_grid` la valeur de `next_grid`
- Réaliser le profilage de la méthode `compute_gol()` pour des grilles générées aléatoirement (méthode `generate()` de `Grid2d`) de tailles 10x10, 100x100, 1000x1000