CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE REGIONAL MIDI - PYRENEES

NFA 031 – Travaux pratiques

IPST-CNAM Licence

PROJET

Années 2017-2018
TOULOUSE
J. LAFORGUE

1. LE BESOIN

On se propose de réaliser un programme qui permet de gérer une mini plate-forme de jeu (à 2 joueurs).

La plate-forme de jeu gère une liste de joueurs et une liste des parties jouées par les joueurs.
Un joueur s'identifie, choisit un adversaire puis joue contre lui.
Chaque joueur obtient un score à chaque partie jouée.
Dans le cadre de NFA031, nous proposons de jouer à un jeu : le jeu de *othello*.

A terme le programme devra être capable de :

- [FUNC 1] lire les deux fichiers texte (fournis)
  - le fichier des inscrits contenant la liste des joueurs
  - les parties jouées par chacun des joueurs
- [FUNC 2] afficher tous les joueurs inscrits et afficher toutes les parties jouées
- [FUNC 3] ajouter un nouveau joueur (nouvelle inscription)
- [FUNC 4] un joueur s'identifie par un mot de passe. On affiche le résultat de l'identification.
- [FUNC 5] afficher les parties jouées du joueur
- [FUNC 6] choisir un adversaire pour jouer
- [FUNC 7] démarrer et arrêter une partie de othello, et mémoriser la partie jouée par les deux joueurs
- [FUNC 8] implémenter les règles de jeu pour jouer au jeu d'othello.

Bien sur, Il n'est pas envisagé de tout développer en une seule fois. Nous allons réaliser ce programme en 3 étapes.

2. ETAPE 1

Dans cette étape on se propose de réaliser les fonctionnalités :
[FUNC 1] [FUNC 2] [FUNC 3].

Les classes que vous devez créer et/ou modifier sont :

- **PFJeu**
  Cette classe est la plate-forme de jeu qui contient:
  - la liste des joueurs (ArrayList<Joueur>)
  - la liste des parties (ArrayList<Partie>)

- **Joueur**
  Cette classe est la description d'un joueur. Un joueur est caractérisé par :
  - son identification (String) (unique) (ex: nom+prénom, pseudo)
  - son mot de passe (String)

- **Partie**
  Cette classe est la description d'une partie jouée. Une partie jouée est caractérisée par :
  - le numéro de la partie (int)
  - l'identification du joueur
  - la date de création (String) au format "JJ/MM/AAAA hh:mm:ss"
  - le nom du jeu (= "othello")
  - partie terminée ou non (1 ou 0)
  - le score obtenu par le joueur (int)
  - le score obtenu par l'adversaire (int)

Pour démarrer votre programme vous partez du code fourni (projet.zip) dans lequel vous trouvez notamment :
une classe **Projet** avec une méthode main qui crée le **PFJeu** et l' **IHMPFJeu**.
L'IHMPFJeu prend en entrée le PFJeu.

Le constructeur de la classe **PFJeu** doit lire les deux fichiers textes :

- le fichier data/Joueurs.txt
  Chaque ligne de ce fichier est un joueur, de la forme :
  identification;mot de passe

  Exemple :

```
jack31;1mdptdap
killer;dougdoug
JeanDupont;$aze_2
```

- le fichier data/Parties.txt
  Chaque ligne de ce fichier est une partie jouée, de la forme :
  numero;ident;date;"othello";terminé;score;score_adversaire

  Exemple :

```
1;jack31;22/12/2017 13:30:45;othello;0;8;3
1;killer;22/12/2017 13:30:45;othello;0;3;8
2;jack31;22/11/2017 13:50:00;othello;1;32;32
2;killer;22/11/2017 13:50:00;othello;1;32;32
```

```
3;killer;23/12/2017 08:21:00;othello;1;50;14
3;JeanDupont;23/12/2017 08:21:00;othello;1;14;50
4;JeanDupont;10/01/2018 22:30:00;othello;0;2;2
4;jack31;10/01/2018 22:30:00;othello;0;2;2
```

(Vous utilisez la méthode String[] Terminal.lireFichierTexte(String nomFichier); pour lire les fichiers)
(La méthode String[] split(String ligne, int n) permet de décoder une ligne du fichier contenant n champs).

Pour traiter le fichier des joueurs:

Pour chaque ligne lue, on crée un joueur et on l'ajoute à la liste des joueurs de la plate-forme de jeu

Pour traiter le fichier des parties jouées :

Pour chaque ligne lue, on crée la partie que l'on ajoute à la liste des parties de la plate-forme de jeu.

La classe **IHMPFJeu** crée un formulaire dans lequel :

- deux boutons permettent d'afficher tous les joueurs et toutes les parties (chaque joueur ou partie est affiché sous la forme d'une chaîne de caractère)
- deux textes de saisie et un bouton permettant d'ajouter un nouveau joueur. Vous devez tester si l'identifiant saisi n'est pas vide, si contient un caractère blanc et si le joueur n'existe pas déjà,. Vous devez tester si le mot de passe n'est pas vide, si contient un caractère blanc et ne doit pas être plus long de 15 caractères.

Pour réaliser ces trois actions, codez dans la classe Site les trois méthodes suivantes :

public String listerTousJoueurs()

Cette méthode retourne sous la forme d'une chaine de caractère, tous les joueurs (une boucle qui appelle la méthode toString de chaque produit)

public String listerToutesParties()

Cette méthode retourne sous la forme d'une chaine de caractère, toutes les parties (une boucle qui appelle la méthode toString de chaque partie)

public String inscrire(String ident, String mdp)

Cette méthode teste si le joueur n'existe pas déjà, ajoute ou non le nouveau joueur, et retourne sous la forme d'une chaine de caractère le résultat de l'inscription.

Ces trois méthodes sont appelées dans la méthode **submit** de la classe IHMPFjeu.

Nous supposons que les fichiers texte ne contiennent pas d'erreur de syntaxe: pas de "point-virgule" en moins, toutes les informations sont correctement écrites.

La méthode listerTousJoueurs affiche, par exemple, le résultat ainsi :

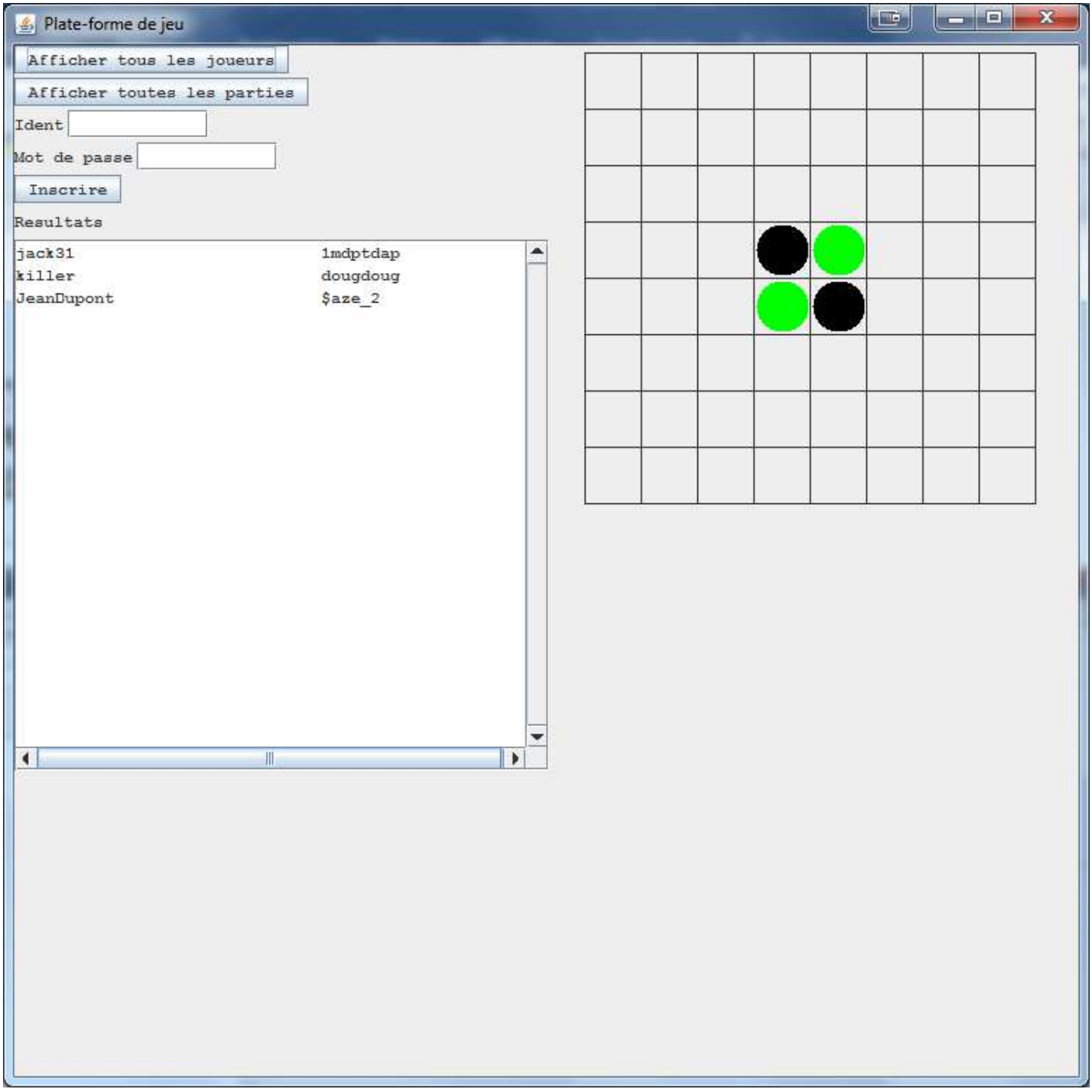
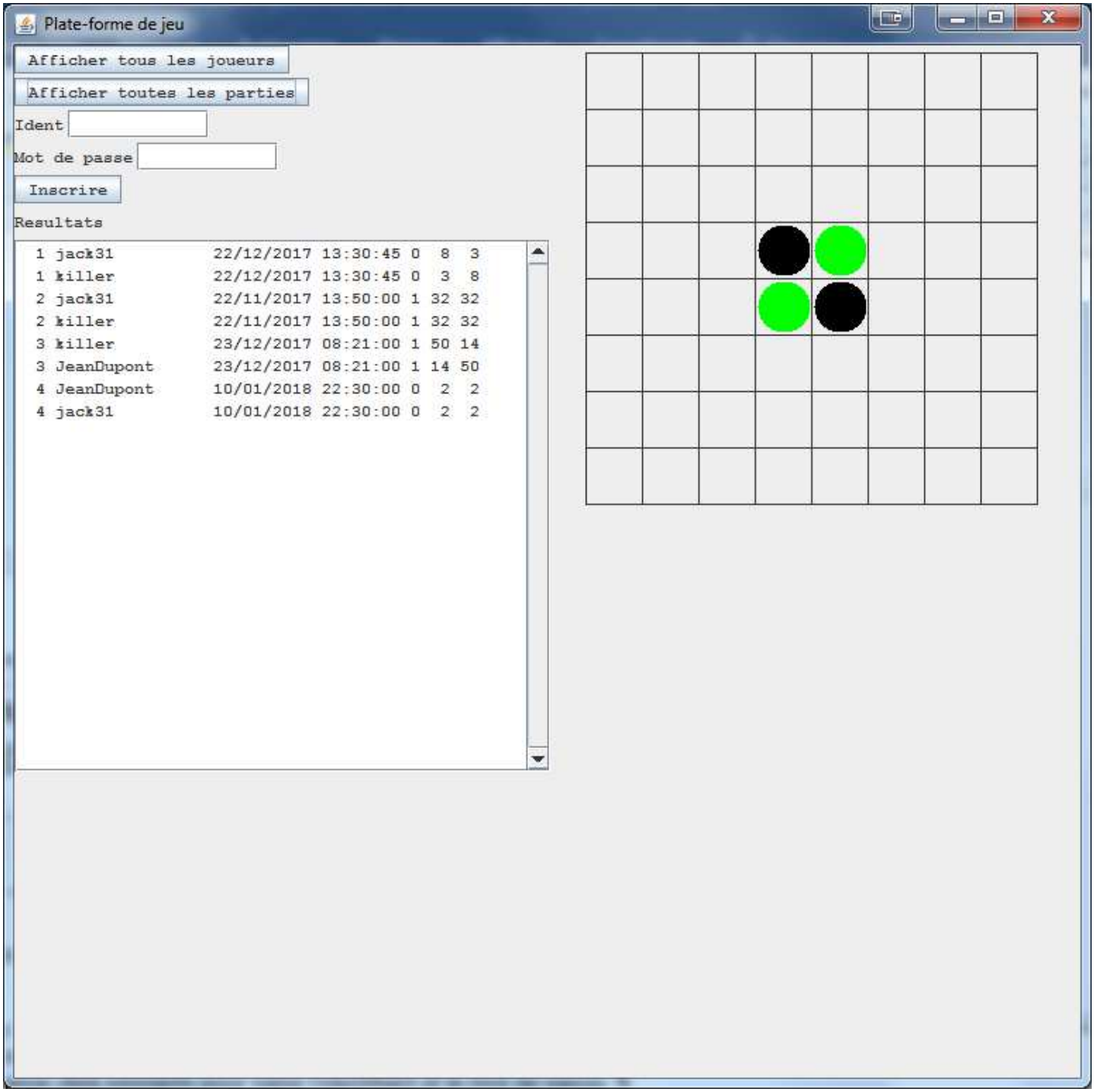


Plate-forme de jeu
Afficher tous les joueurs
Afficher toutes les parties
Ident
Mot de passe
Inscrire
Resultats
jack31 1mdptdap
killer dougdoug
JeanDupont $aze_2

La méthode listerToutesParties affiche, par exemple, le résultat ainsi :



3. ETAPE 2

Dans cette étape on se propose de réaliser la fonctionnalité [FUNC 4], [FUNC 5] et [FUNC 6].

Pour la [FUNC 4], vous créez dans la classe PFSite la méthode qui réalise l'identification du joueur courant, et vous ajoutez dans l'IHM un bouton permettant d'exécuter cette méthode. On utilise les deux textes de saisie déjà existants pour saisir l'identifiant et le mot de passe.
On affiche, dans un texte non éditable de l'ihm, le nom du joueur qui s'est identifié.

Pour la [FUNC 5], vous créer un nouveau bouton pour afficher les parties du joueur courant.

Pour la {FUNC 6], dans l'IHM on crée une liste de valeurs (addListScroll). Puis, après l'initailisation de la plate-forme on utilise la méthode setListData pour initialiser la liste de valeurs avec la liste de tous les joueurs. On ajoute un bouton pour valider la sélection de l'adversaire.
On affiche, dans un texte non éditable de l'ihm, le nom de l'adversaire que l'on a choisi.

Exemple d'une IHM :

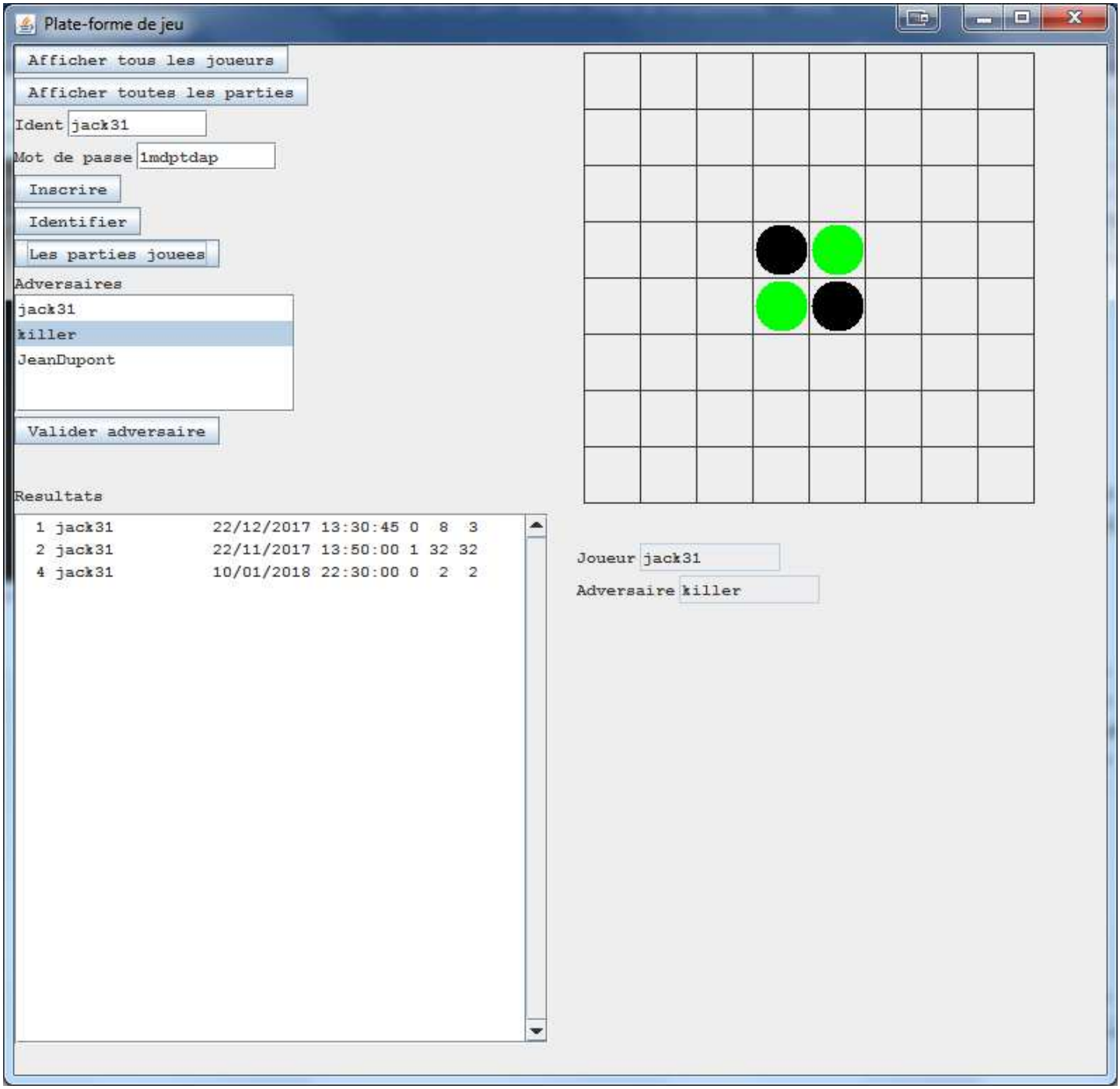


Plate-forme de jeu
Afficher tous les joueurs
Afficher toutes les parties
Ident jack31
Mot de passe 1mdptdap
Inscrire
Identifier
Les parties jouees
Adversaires
jack31
killer
JeanDupont
Valider adversaire
Resultats
1 jack31 22/12/2017 13:30:45 0 8 3
2 jack31 22/11/2017 13:50:00 1 32 32
4 jack31 10/01/2018 22:30:00 0 2 2
Joueur jack31
Adversaire killer

4. ETAPE 3

Dans cette étape on se propose de réaliser la fonctionnalité [FUNC 7] qui consiste à démarrer une partie puis l'arrêter.
Si les deux joueurs ne sont pas tous les deux identifiés alors la partie ne peut pas démarrer.

On ajoute deux nouveaux boutons : un pour démarrer la partie et un autre pour arrêter la partie.
Quand la partie s'arrête, on ajoute cette partie dans les parties jouées de la plate-forme de jeu pour chacun des joueurs (*pas dans le fichier !!*).

Pour réaliser la [FUNC 8], si un joueur a gagné ou si aucun ne peut jouer alors la partie s'arrête automatiquement. On ajoute cette partie dans les parties jouées de la plate-forme de jeu pour chacun des joueurs.

Il faut aussi coder les règles du jeu d'othello suivantes (vous les codez une après l'autre) :

Règle 1 : Le premier joueur qui commence, joue avec la couleur noire.

Règle 2 : Chaque joueur joue à tour de rôle

Règle 3 : Un joueur joue toujours sur une case vide

Règle 4 : Pour jouer, un joueur doit retourner des pions de l'adversaire

Règle 5 : S'il existe au moins 1 case jouable, le joueur doit jouer

Règle 6 : Si un joueur ne peut pas jouer, il passe son tour

Règle 7 : La partie s'arrête quand aucun des joueurs ne peuvent jouer.

Règle 8: Le gagnant est celui qui a le plus de pion de sa couleur

5. QUELQUES PRECISIONS

Les règles de codage que vous devez suivre sont :

- sauf pour les constantes, les attributs de toutes les classes doivent être privés
- toutes les classes commencent par une majuscule
- tous les attributs, méthodes, paramètres et variables commencent par une minuscule
- les noms de tous les répertoires de package sont en minuscule
- les packages créés commencent tous par "fr.cnam"
- une indentation stricte est respectée dans tout le code
- un minimum de commentaire est nécessaire afin de comprendre ce que fait le code. Surtout quand ce que fait le code n'est pas décrit précisément dans le sujet.

Conseils :

- Ecrivez peu de code à chaque fois afin d'avoir toujours un code qui se compile correctement
- Après chaque étape ou sous-étape, le programme s'exécutant correctement, faites une sauvegarde des sources. Cela permet de pouvoir revenir en arrière si besoin et permet aussi de pouvoir donner à l'enseignant une version du programme qui s'exécute correctement et qui remplit correctement une partie des fonctionnalités demandées dans le sujet.
- Dans un premier temps ne faites pas plus que le sujet ne le demande. Une fois que le programme respecte le sujet, vous faites une sauvegarde et vous pouvez apporter des améliorations qui sont toujours appréciées par le correcteur, à condition qu'elles soient bien commentées.

Dans le fichier Projet.java, vous renseigner les noms et prénoms du groupe.

6. ARCHITECTURE DES REPERTOIRES

L'architecture suivante est une proposition de création des répertoires et des fichiers pour réaliser votre projet. Cette architecture doit être respectée.

projetNFA031
- **data**
  - Joueurs.txt
  - Parties.txt
- **src**
  - **fr**
    - **cnam**
      - **projet**
        - Projet.java (*main*)
        - PFJeu.java
        - Joueur.java
        - Partie.java
        - IHMPFJeu.java
        - Othello.java
      - **ihm**
        - Formulaire.java
        - FormulaireException.java
        - FormulaireInt.java
        - ....
      - **util**
        - DateString.java
        - Terminal.java
        - TerminalException.java
        - ...
- **bin**
  - <les .class>

Les classes en rouge sont des classes qui vous sont données.

Le répertoire **data** contient les fichiers de données.

7. PLANNING

Vous avez 3 séances de TP pour réaliser ce projet

Le projet sera ramassé 1/4 heure avant la fin de la dernière séance de TP (le temps de les ramasser tous).
L'enseignant vous demandera de déposer vos projets dans le répertoire "commun" spécifique (**lecteur réseau Q:**) qu'il récupèrera sur une clef USB.

Il est donc **impératif** que, au moins un membre du groupe de TP, soit présent lors de la dernière séance afin de donner son projet.