Université Paris Sud
Licence d'informatique/IUP-Miage2
Année 2004-2005

auteur : Frédéric Vernier
web: http://vernier.frederic.free.fr/indexpac.html
semaine : 18-23 octobre 2004
docs: http://java.sun.com/j2se/1.4.2/docs/api/index.html

TD/TP PAC - Programmation n° 4
Classes abstraite, Interfaces

Exercice 1

Piece.java

```
public interface Piece {
  /**
   * retourne si oui ou non cette piece est noire
  **/
  public boolean isNoire();
  /**
   * se compare soi-meme a une autre piece
  **/
  public boolean compareTo(Piece Piece_Arg);
  /**
   * se transforme en Dame
  **/
  public void transform();
  /**
   * @return une lettre representant la piece (R=roi, D=dame, ...)
  **/
  public char getLetter();

}
```

Plateau.java

```
public class Plateau  {
  private Piece[][] Cases;
  // cree un plateau vide de la dimension choisie
  public Plateau (int NbCasesCote_Arg) {
    Cases = new Piece[NbCasesCote_Arg][NbCasesCote_Arg];
    for (int i=0; i<NbCasesCote_Arg; i++)
      for (int j=0; j<NbCasesCote_Arg; j++)
        Cases[i][j] = null;
  }

  public String toString() {
    String String1 = "";
    for (int i=Cases.length-1; i>=0; i--) {
      for (int j=0; j<Cases[0].length; j++)
        if (Cases[i][j]==null) String1 = String1+"[ ]";
        else if (Cases[i][j].isNoire())
          String1= String1+ "("+Cases[i][j].getLetter()+")";
        else String1= String1+"{"+Cases[i][j].getLetter()+"}";
      String1= String1+"\n";
    }
    return String1;
  }
}
```

Ecrivez la classe PieceDamier qui décrit {pion, dame}

1. Ecrivez la classe PieceEchiquier qui décrit {pion, tour, cavalier, fou, dame, roi}
   attention: au echec seul un pion peut se transformer en dame !
2. Ajouter les méthodes creerEchiquier() et creerDamier() dans Plateau

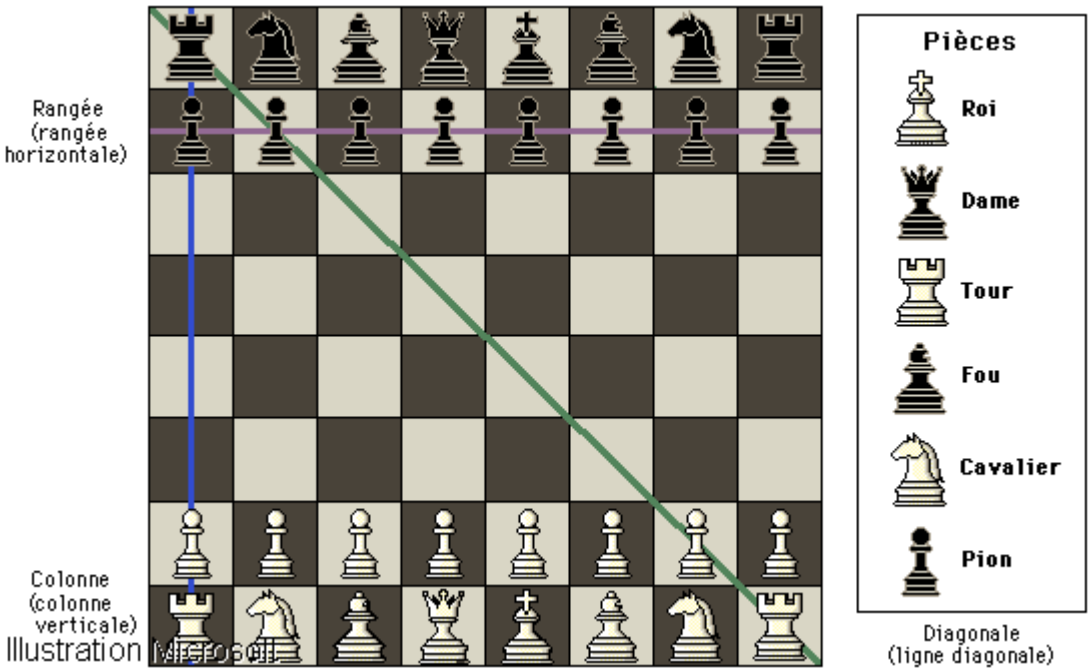


Illustration 1 : *Un échiquier au début du jeu*

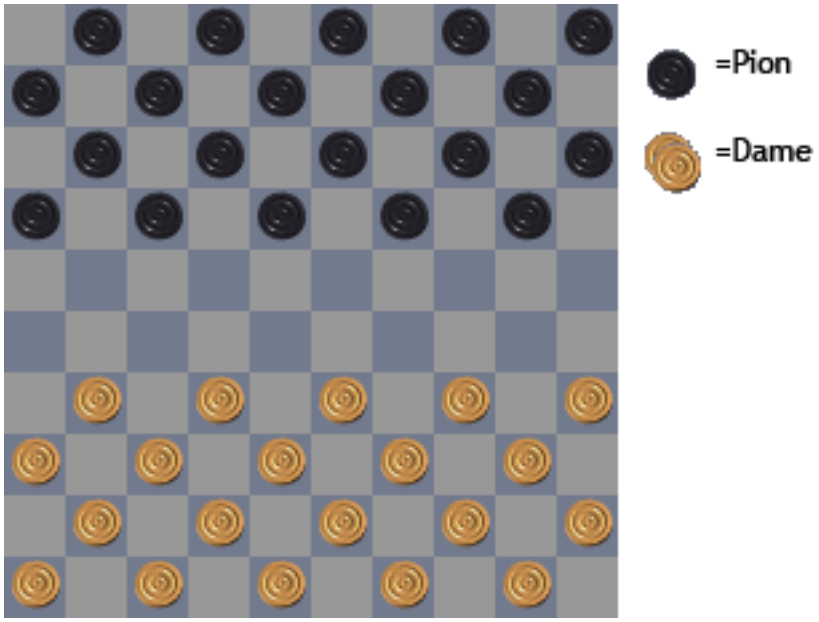
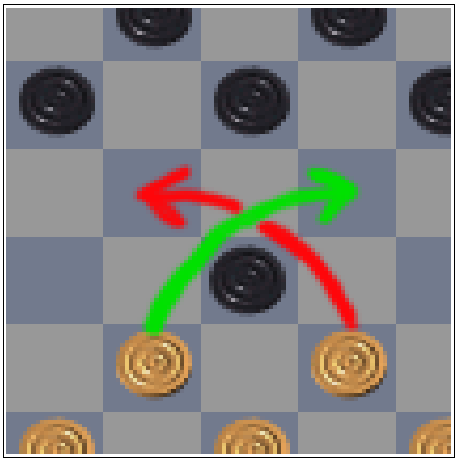


Illustration 2 : *Un Damier au début du jeu (sans Dame)*

3. Créer l'interface JoueurDame qui possède
- une méthode pour connaître le plateau
- une méthode qui retourne vrai si il abandonne, faux sinon
- une méthode pour jouer = déplacer une pièce

5. Ajouter une méthode à l'interface Piece pour que chaque pièce connaisse le joueur auquel elle appartient
6. Créer une classe JoueurDameMauvaisJoueur qui abandonne dès qu'il a 2 dames de moins que son adversaire et/ou 4 pièces de moins. Implémenter aussi la méthode de récupération du plateau mais pas celle de jeu. Comment doit être déclaré cette classe qui n'implémente pas toutes les fonctions de l'interface ?
7. Ajouter une méthode de Plateau qui déplace une pièce d'une case a l'autre (les cases de départ et d'arrivée seront données en argument) si et seulement si la case d'arrivée de contient rien. La case de départ sera mise à null. La case d'arrivée pointera vers la pièce déplacée. Ajouter une méthode pour retourner le plateau. Retourner revient a faire une rotation de 180 degrés ... ce qui revient a faire une symétrie horizontale puis une symétrie verticale ! (x->Max-x puis y->Max-y)
8. Surcharger cette méthode dans Damier pour qu'un déplacement valide vérifie EN PLUS que la case de départ et d'arrivée sont sur la même diagonale. (y1-x1==y2-x2 OU y1+x1==y2+x2)
9. Créer une classe JoueurDameAgressif qui fait l'hypothèse qu'il est « en bas » du damier. Il n'abandonne que lorsqu'il n'a plus de pièce. Pour jouer il cherche parmi ses pièces si il peut prendre (cad il y a un pion adverse devant et une case vide derrière comme sur le figure de droite). Si il n'y a aucune pièce a prendre, le joueur avance le pion le plus en recul (celui le plus en bas possible) ... Toujours en diagonale et de 1 case. On ne tient pas compte des dames.



10. Créer une interface GameListener et un objet GameEvent. Rajouter la méthode addGameListener(GameListener GameListener_Arg) dans Plateau. Cette méthode stocke les listener dans un vecteur. Chaque fois qu'une pièce est déplacée sur le plateau on appelera une fonction fireGameAction(new GameEvent(...)). Cette fonction fireGameAction appelle la ou les fonctions de chacun des GameListener enregistrés.

Exercice 2

L'objectif de cet exercice est de coller ensemble les exercices du TD3 (sur les villes, les personnes et les banques) de rajouter l'interface Contribuable et Citoyen :

- Un contribuable doit pouvoir répondre à la question public float getRevenusAnnuel(); et public void payerImpots(float Montant_Arg, Compte TresorPublic);
- Un Citoyen doit répondre une interface comprenant
  public void setCirconscription(Ville Circonscription);
  public Personne voterParmi(Vector Candidats);

Toutes les classes correspondant aux banques seront mises dans un package banque. De même on créera un package entreprise et personne.

En plus de mettre ensemble les différentes pièces du puzzle (problème des packages, des accès, etc.), vous devrez faire en sorte que les personnes soient des contribuables et des citoyens. Les villes qui posséderont une liste de citoyens devront pouvoir aléatoirement élire l'un d'entre eux et les pays devront au travers de la liste des villes, ponctionner l'impôt chez tous les contribuables.

Il vous revient de trouver un scénario à tester afin de mettre au point le programme.

Enfin créer un fichier jar avec tous les fichier et un manifest pour exécuter le programme.

NB: Pour ceux qui liront l'énoncé jusqu'au bout avant de se lancer, vous trouverez les fichiers JAVA corrigés de la semaine dernière à cette adresse :
http://vernier.frederic.free.fr/PAC/TD3/src_corrigees/