DS S2 Programmation Orientée Objet

Jean-Claude MARTIN 17 Juin 2016 Durée : 2h. Documents autorisés : tout sauf livres.

* Lire tout l’ énoncé avant de commencer.
* Vous donnerez les instructions package mais pas les instructions import.
* On ne vous demande pas les méthodes getXXX et setXXX.
* C’est à vous de décider la manière la plus pertinente de programmer ce qui est demandé.
* Vous devez faire des choix que vous devez expliquer.
* Le bareme est donné à titre indicatif.

But de votre programme : Vous devez programmer un jeu de plateau de morpion à 3 pions qui consiste pour l’utilisateur à aligner 3 pions horizontalement, verticalement ou en diagonale. L’utilisateur commence par jouer. Ensuite le système joue. Et ainsi de suite jusqu’à ce que la partie soit finie.

1. Modèle de Pion (2 pts)

Définir la notion de Pion. Un pion sera représenté par les attributs suivants :

* un type (de type int)
* un numéro de ligne
* un numéro de colonne

On considère que la notion de Pion est trop générale pour permettre de créer une instance de Pion. Donner le code source de Pion.

Vous choisirez si c’est une interface, une classe abstraite ou une classe concrète.

Si c’est pertinent, vous donnerez, en plus de l’entête, les déclarations d’attributs (s’il y en a, on autorisera des sous-classes à y accéder).   
On ne vous demande rien d’autre.

1. Modèle de Pion de Morpion (1.5 pts)

On veut maintenant programmer la notion de pion spécifique au jeu de morpion. Par rapport à la notion générale de Pion, un pion de jeu de morpion inclura en plus :

* deux constantes de type int PION\_UTILISATEUR et PION\_SYSTEME qui seront des valeurs possibles pour l’attribut type de ce pion
* Deux méthodes
  + public static PionMorpion creerPionUtilisateur (int ligne, int colonne)
  + public static PionMorpion creerPionSysteme (int ligne, int colonne)

qui créent une instance de PionMorpion avec la bonne valeur pour l’attribut type et retournent une référénce vers cette instance. On ne vous demande pas de constructeur.

Donnez le code source de PionMorpion.

3) Point de vue critique (1 pt)

Les instructions que l’on vous a données aux questions 1) et 2) ne sont pas vraiment orientées objet. Expliquez pourquoi.

4) Amélioration (1.5 pts)

Afin de rendre plus orienté objet les classes développées aux questions 1 et 2, on suppose que vous supprimez maintenant la classe PionMorpion et que vous la remplacez par 2 classes : PionUtilisateur et PionSysteme.

Donnez la définition de ces classes (elles comportent seulement l’entête, un constructeur et si besoin des déclarations d’attributs).

Expliquez juste ce que vous modifiez dans Pion si besoin.

Dans les questions suivantes, vous utiliserez PionUtilisateur, PionSysteme et Pion.

5) Modele de Jeu de plateau (3.5 pts)

On vous donne en annexe le code de la classe Plateau.java. Lisez le.

1. Est-ce que Plateau est une classe générique ? Justifiez votre réponse.
2. Pour quelle raison le programmeur a-t-il indiqué (T[][]) dans l’instruction suivante :  
   plateau = (T[][]) new Object [n][n];
3. On vous donne le code ci-dessous :

package morpion.modeles ;

import java.util.ArrayList;

public interface Jouable<T> {

Plateau<T> getPlateau();

ArrayList<T> listeDesCoups();

void ajouter (T p);

void faireJouerSysteme();

boolean userWins();

boolean systemWins();

}

Pour quelle raisons le programmeur de cette interface a-t-il indiqué “public” devant l’interface ?

d) Est-ce que l’on peut écrire “new Jouable()” ?

e) On considère une classe ModeleJeuMorpion qui implémente l’interface Jouable.   
Donnez juste :

* l’entête de la classe de la classe ModeleJeuMorpion
* la déclaration des attributs plateau et listeCoups de la classe ModeleJeuMorpion.

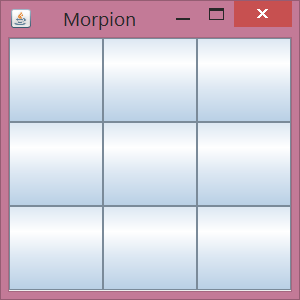
Vous n’avez pas à donner le reste du code de cette classe.

6) VuePlateauMorpion (6 pts)

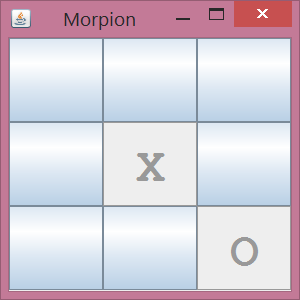
L’utilisateur pourra cliquer sur une case pour indiquer qu’il veut jouer sur cette case.

Vous pourrez utiliser comme Ecouteur la classe ControleurMorpion que vous allez écrire à la question suivante.

Lorsqu’on exécute le main() de la classe VuePlateauMorpion, la vue affichée est la suivante :



Si l’utilisateur clique sur la case du milieu, la vue affiche une “X” puis fait jouer le système qui choisit une place et affiche un “O” :



On vous donne en annexe une partie incomplète du code de la classe VuePlateauMorpion.

Ecrivez sur votre copie (pas sur le sujet : il n’y a pas la place : chaque “...” peut représenter plusieurs lignes de code ! Vous n’êtes pas obligé de recopier les commentaires, ils sont juste là pour vous aider) :

1. les déclarations des attributs et constantes de la classe VuePlateauMorpion
2. le code du constructeur de VuePlateauMorpion
3. le code de la méthode miseAJour () de VuePlateauMorpion

7) classe ControleurMorpion (3 pts)

Définir une classe ControleurMorpion qui écoute un bouton de la grille et qui mets à jour le modèle et la vue :

1. entête de ControleurMorpion
2. déclaration des attributs de ControleurMorpion
3. méthode actionPerformed de ControleurMorpion

8) Mémorisation d’une partie (1.5 pts)

On souhaite mémoriser la suite des coups de la partie sur disque.

Définissez la méthode qui permet de le faire et indiquez où vous l’appelez.

Annexes

**// Plateau.java**

//-----------------------------------------

package morpion.modeles ;

//-----------------------------------------

public class Plateau<T> {

int n ;

T plateau [][];

//-----------------------------------------

Plateau (int n) {

this.n = n ;

plateau = (T[][]) new Object [n][n];

}

//-----------------------------------------

void placer (int ligne, int colonne, T p) {

plateau [ligne][colonne] = p ;

}

//-----------------------------------------

int getNb () {

int nb = 0;

for (int l=0; l < n; l++)

for (int c=0; c < n; c++)

if (plateau [l][c] != null)

nb ++;

return nb ;

}

//-----------------------------------------

boolean isFull () {

return getNb() == n \* n ;

}

//-----------------------------------------

public T get(int l, int c) {

return plateau [l][c];

}

}

**// VuePlateauMorpion.java**

//-------------------------------------------------------

package morpion.vues;

import java.awt.BorderLayout;

import java.awt.Container;

import java.awt.Font;

import java.awt.GridLayout;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.UIManager;

import morpion.controleurs.\*;

import morpion.modeles.\*;

//-------------------------------------------------------

public class VuePlateauMorpion extends JPanel {

//-------------------------------------------------------

// ATTRIBUTS

...

//-------------------------------------------------------

// CONSTRUCTEUR

public VuePlateauMorpion (ModeleJeuMorpion m) {

...

}

// METHODES

//-----------------------------------------

public void miseAJour() {

Plateau<Pion> plateau = modele.getPlateau();

// Pour chaque bouton de la grille

...

// Afficher sur le bouton le symbole qui correspond au pion

...

// Désactiver ce bouton

b.setEnabled(false);

...

}

//-----------------------------------------

public static void main (String args []) {

int taille = 3 ;

// Créer un modèle du jeu

ModeleJeuMorpion m = new ModeleJeuMorpion(taille);

// Créer une vue sur ce modèle

VuePlateauMorpion v = new VuePlateauMorpion (m);

// Creer une fenetre

JFrame f = new JFrame ("Morpion");

Container c = f.getContentPane();

// Ajouter a la fenetre la vue

c.add(v);

// Afficher la fenetre

f.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

f.setSize(300,300);

f.setLocationRelativeTo(null);

f.setVisible(true);

}

//-------------------------------------------------------

public void setModele(ModeleJeuMorpion m) {

this.modele = m ;

}

}