L3 SRI
Programmation Orientée Objet (session 1, 2018-19)

Durée = 1h30

Documents autorisés. Les réponses non justifiées ne seront pas comptabilisées.

Exercice 1 : Polymorphisme (7 points)

On dispose des 4 classes suivantes :

```
public class A {
    protected char att ;
    public A () {
        att = 'a' ;
    }
    public void affiche() {
        System.out.println("Dans A : "+att) ;
    }
}

public class B extends A {
    public B () {
        att = 'b' ;
    }
    public void affiche() {
        System.out.println("Dans B : "+att) ;
    }
}

public class C extends B {
    public C () {
        att = 'c' ;
    }
    public void affiche() {
        System.out.println("Dans C : "+att) ;
    }
}
```

```
public class Main {

    public static void main (String arg[]) {
        A a = new A() ;
        B b = new B() ;
        C c = new C() ;
        A abis = new B() ;
        A ater = new C() ;
        B bbis = new C() ;

        a.affiche() ;
        b.affiche() ;
        c.affiche() ;
        abis.affiche() ;
        ater.affiche() ;
        bbis.affiche() ;
        ((A)abis).affiche() ;
        ((B)ater).affiche() ;
    }
}
```

1. Donner le résultat obtenu lors de l'exécution de la classe `Main`.
2. Expliquer le(s) différent(s) cas de polymorphisme rencontré(s).
3. Donner un exemple des autres cas de polymorphisme que vous connaissez.

Exercice 2 : Problème (13 points)

On veut implémenter le jeu de taquin. Le taquin est un jeu solitaire en forme de damier carré de dimension n sur n. Il est composé de $(n \times n) - 1$ petits carreaux tous différents (on considèrera ici qu'ils sont identifiables par une lettre inscrite sur le carreau). Le jeu consiste à partir d'une configuration initiale quelconque des carreaux et à arriver dans une configuration donnée en faisant coulisser les carreaux un à un.

Par exemple, passer de

A	B	C
D	E	F
G	H	

à

	H	G
F	E	D
C	B	A

On ne s'intéressera pas ici à l'algorithme permettant de résoudre le jeu mais plutôt à l'ensemble des classes Java permettant à un utilisateur de jouer. On va donc implémenter 3 classes :

— la classe `Case` qui permet de représenter une des cases du damier du jeu ; cette case contiendra soit un carreau (donc une lettre pour simplifier), soit rien (case dite "vide") ;

— la classe `Plateau` qui permet de représenter le damier du jeu ; ce sera donc une matrice de cases dans laquelle on aura une et une seule case vide et on pourra échanger une case non vide avec la case vide si elles sont adjacentes (ce qui correspondra au mouvement des carreaux sur le damier) ;

— la classe `Jeu` qui permet de représenter le jeu ; elle contiendra un plateau et permettra de l'initialiser dans une configuration donnée puis de jouer en demandant à l'utilisateur la lettre qu'il veut bouger.

1. Donner la définition des 3 classes avec leurs attributs, leurs constructeurs et leurs méthodes mais sans écrire le détail des corps des méthodes.
2. Ecrire la méthode `void affiche()` de la classe `Jeu` qui permet d'afficher le damier du jeu. Ecrire aussi les méthodes utilisées par `affiche`.
3. Ecrire la méthode `void bougerCase(char c)` de la classe `Plateau` permettant faire coulisser sur le damier le carreau correspondant à la lettre donnée en paramètre. Bien sûr si cette lettre n'est pas à côté de la case vide ou n'existe pas sur le plateau, il faudra le détecter et le traiter proprement (indice : utilisation des exceptions).
4. Ecrire la méthode `boolean jouerUnCoup()` de la classe `Jeu` qui permet d'interroger l'utilisateur, de récupérer sa réponse et de faire le déplacement demandé s'il est possible. On utilisera la classe `Scanner` pour récupérer la réponse de l'utilisateur.
5. Ecrire la méthode `void faireUnePartie()` de la classe `Jeu` qui permet à l'utilisateur d'enchaîner plusieurs coups.

Annexe

Pour la classe `Exception`	
`Exception(String s)`	crée une exception (s étant le message associé)
`String getMessage()`	récupère le message associé à l'exception
Pour la classe `Scanner`	
`Scanner(System.in)`	crée un scanner sur le flot d'entrée
`String nextLine()`	récupère la chaine de caractères correspondant à la saisie de l'utilisateur
Pour la classe `String`	
`char charAt(int p)`	renvoie le caractère qui est en position p dans la string courante