EPF – Lausanne *COURS DE MATHEMATIQUES SPECIALES*

17 décembre 2008

Contrôle d'informatique no 2

Durée : 1 heure 45'

Nom : …… Groupe : ☐

Prénom : ……

Barème sur 100 points

No	1	2	3
Total points	12 points	44 points	44 points

Remarque générale : toutes les questions qui suivent se réfèrent au langage de programmation Java (à partir du JDK 5.0) et les réponses doivent être rédigées à l'encre et d'une manière propre sur ces feuilles agrafées.

Sujet no 1.

Indiquer si les affirmations suivantes sont vraies ou fausses (en écrivant, selon le cas, VRAI ou FAUX à l'endroit prévu à cet effet) :

a) A l'exécution d'un programme (au **RunTime**), la taille d'un objet tableau peut être modifiée (à l'aide du champ **length**).

……

b) Dans la définition d'une méthode statique, on ne peut pas utiliser le mot clé **this**.

……

c) L'adresse d'un objet qui est passée comme argument (effectif) dans l'appel d'une méthode ne peut pas être modifiée par la méthode appelée.

……

d) Toute classe appartenant à un projet interactif exécutable en mode console doit contenir une méthode publique et statique **main**.

Sujet no 2.

On présente ci-dessous les canevas de deux classes appartenant au même package ***cms_ctr2***, à savoir :

- la classe ***Carre*** qui sert à créer et à travailler avec des objets correspondant à des carrés caractérisés par la longueur de leur côté ;
- la classe "principale" publique ***CP_Ctr2_2*** qui contient la méthode ***main*** et utilise la classe ***Carre***.

i. Il s'agit d'abord de définir la classe ***Carre*** munie de :

- la déclaration d'un champ privé de type numérique réel appelé ***cote*** ;
- la définition d'une méthode publique d'instance appelée ***getCote*** qui :
  - n'a pas d'argument ;
  - retourne la valeur du champ ***cote*** (de l'objet qui appelle la méthode) ;
- la définition d'une méthode publique d'instance appelée ***setCote*** qui :
  - a un seul argument de type numérique réel ;
  - ne retourne aucun résultat ;
  - si la valeur de son argument est négative ou nulle, stocke la valeur ***0*** dans le champ ***cote*** (de l'objet qui appelle la méthode) ;
  - autrement (c'est-à-dire si la valeur de son argument est strictement positive), copie la valeur de son argument dans le champ ***cote*** (de l'objet qui appelle la méthode) ;
- la définition d'un constructeur public qui n'a pas d'argument et qui stocke dans le champ ***cote*** (du nouvel objet créé) la valeur réelle ***1***;
- la définition d'un constructeur public (surchargé) qui a un seul argument de type numérique réel et qui appelle la méthode ***setCote*** en lui passant comme argument effectif la valeur de son propre argument ;
- la définition d'une méthode publique et statique appelée ***calculerAire*** qui :
  - a un seul argument de type **Carre** ;
  - retourne une valeur réelle : l'aire du carré correspondant à l'objet argument de la méthode ;

- la définition d'une méthode d'instance appelée ***calculerPerimetre*** qui :
  - n'a pas d'argument ;
  - retourne une valeur réelle : le périmètre du carré correspondant à l'objet appelant la méthode ;

- la définition d'une méthode d'instance ***deformer*** qui :
  - a un seul argument de type numérique entier ;
  - retourne la référence d'un nouvel objet ***Carre*** qui est créé par cette méthode et qui correspond à un carre dont le côté est:
    - calculé en multipliant le côté du carré correspondant à l'objet appelant la méthode par la valeur de l'argument de la méthode, si l'argument de la méthode est strictement positif ;
    - **0** dans les autres cas (c'est-à-dire si l'argument de la méthode est négatif ou nul) ;

- la définition d'une méthode publique et statique ***comparer*** qui :
  - a deux arguments de type **Carré** ;
  - retourne une valeur de type numérique entier, à savoir :
    - ***-1*** si le côté du carré correspondant au premier argument est plus petit que le côté du carré correspondant au deuxième argument ;
    - ***1*** si le côté du carré correspondant au premier argument est plus grand que le côté du carré correspondant au deuxième argument ;
    - ***0*** dans les autres cas (c'est-à-dire si les côtés des carrés correspondant aux deux arguments sont isométriques).

```
package cms_ctr2;

//La classe définie ci-dessous est incomplète et il faudra y ajouter
//les éléments indiqués en commentaires

class Carre
{
//Ajoutez la déclaration du champ privé cote

……
……
……
```

//Ajoutez la définition de la méthode publique **getCote**

……
……
……
……
……
……

//Ajoutez la définition de la méthode publique **setCote**

……
……
……
……
……
……
……
……
……
……

//Ajoutez la définition du **constructeur** public sans argument

……
……
……
……
……
……

//Ajoutez la définition du **constructeur** public avec un argument

……
……
……
……
……
……

//Ajoutez la définition de la méthode publique et statique **calculerAire**

……
……
……
……
……
……

//Ajoutez la définition de la méthode d'instance **calculerPerimetre**

……
……
……
……
……
……

//Ajoutez de la définition de la méthode d'instance **déformer**

……
……
……
……
……
……
……
……
……

//Ajoutez la définition de la méthode publique et statique **comparer**

……
……
……
……
……
……
……
……
……
……
……

}//fin de la classe Carre

ii. Il s'agit ensuite de compléter, en suivant les indications données sous formes de commentaires, la classe principale ***CP_Ctr2_2*** afin de créer et de manipuler des objets de type ***Carre***.

```
package cms_ctr2;
//Dans la classe  définie ci-dessous, la méthode main est incomplète
//et il faudra y ajouter les instructions précisées en commentaires
public class CP_Ctr2_2
{
      public static void main(String args[])
      {
            //Ajoutez une instruction qui crée un nouvel objet de type
            //Carre dont la référence est stockée dans la variable objet
            //carre1 déclarée à cet effet
```

……

```
            //Ajoutez une instruction qui donne au champ cote de l'objet
            //référencé par carre1 la valeur réelle 25
```

……

```
            //Ajoutez une instruction qui déclare une variable locale
            //réelle aire et y stocke l'aire du carré correspondant
            //à l'objet référencé par carre1, cette aire étant calculée
            //à l'aide de la méthode statique calculerAire
            //de la classe Carre
```

……

```
            //Rien à ajouter
            System.out.println("L'aire vaut " + aire + ".");

            //Ajoutez une instruction qui crée un nouvel objet de type
            //Carre correspondant à un carré de côté 2.5
            //dont la référence est stockée dans la variable objet
            //carre2 déclarée à cet effet
```

……

```
            //Ajoutez une instruction qui déclare une variable locale
            //réelle perimetre et y stocke le périmètre du carré
            //correspondant à l'objet référencé par carre2, ce périmètre
            //étant calculé à l'aide de la méthode d'instance
            //calculerPerimetre de la classe Carre
```

……

```
            //Rien à ajouter
            System.out.println("Le périmètre vaut " + perimetre + ".");

            //Ajoutez une instruction qui déclare une variable objet locale
            //carre3 de type Carre et y stocke l'adresse d'un nouvel objet
            //Carre qui est créé par un appel de la méthode d'instance
            //deformer et qui doit correspondre à un carré obtenu par une
            //déformation du carré représenté par l'objet référencé par
            //carre2 avec un coefficient de déformation 10
```

……

```
            //Ajoutez une instruction qui déclare une variable locale
            //de type numérique entier res et y stocke le résultat
            //retourné par l'appel de la méthode statique comparer
            //afin de comparer les côtés des carrés qui correspondent aux
            //objets référencés par carre1 et carre3
```

……

```
            //Rien à ajouter
            System.out.println(res);

      }     //fin de la méthode main

}           //fin de la classe CP_Ctr2_2
```

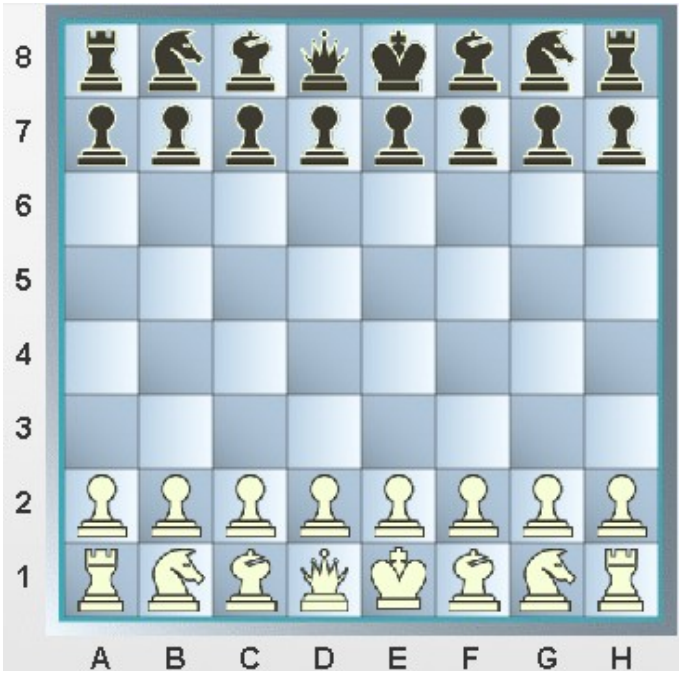
iii. Finalement, on présente ci-dessous les messages obtenus suite à l'exécution du projet contenant les classes ***Carre*** et ***CP_Ctr2_2*** correctement complétées.

```
L'aire vaut 625.0.
Le périmètre vaut 10.0.
0
```

Sujet no 3.

Le but de cet exercice est de réaliser une application autonome interactive qui fournit des informations concernant une case dont la position sur un échiquier est indiquée par l'utilisateur.

Dans la figure ci-dessous, on présente une table d'échec en position verticale et vue de face. On remarque que chacune des 64 cases (ou cellules) peut être localisée de manière unique à l'intersection d'une certaine colonne et d'une certaine ligne. De plus, les colonnes sont identifiées par des lettres allant de ***A*** et ***H*** tandis que les lignes sont numérotées de ***1*** et ***8***.



On vous demande d'écrire le contenu d'un fichier source **CP_Ctr2_3.java** qui définit la classe "principale" publique appartenant au package ***cms_ctr2*** et contenant la méthode ***main***.

Plus précisément, à l'exécution de l'application :

- **a.** on affiche (une seule fois) le message de début : **`Message 1`** (qui remplace des explications plus détaillées du genre "Bonjour ! Ce programme vous demande la position d'une case sur un échiquier et, par conséquent, vous devrez indiquer ci-dessous la colonne et la ligne correspondant à cette case !") ;

Attention : la partie qui suit (du point **b.** au point **p.**) pourra s'exécuter plusieurs fois !

- **b.** on demande (et on stocke) la lettre correspondant à la colonne de la case ;
- **c.** si la lettre indiquée par l'utilisateur n'est pas comprise entre ***A*** et ***H***, on affiche le message **`Colonne non valide !`** et on revient (autant de fois que nécessaire) au point **b.** ;

d. dès que l'indication de la colonne est valide, on demande (et on stocke) la valeur correspondant à la ligne de la case ;

e. si la valeur indiquée par l'utilisateur n'est pas comprise entre ***1*** et ***8***, on affiche le message **`Ligne non valide !`** et on revient (autant de fois que nécessaire) au point **d.** ;

f. dès que l'indication de la ligne est valide, on affiche un message indiquant la couleur (***blanche*** ou ***noire***) de la case correspondant aux informations (valides) données par l'utilisateur ;

g. on affiche une ligne séparatrice formée "d'étoiles" (suite de caractères "astérisque") ;

h. si la cellule indiquée par l'utilisateur se trouve dans la dernière ligne de l'échiquier (c'est-à-dire dans la ligne numéro **8**), on affiche sur une nouvelle ligne le message **`Dernière ligne !`** et on passe au point **j.** ;

i. autrement, on affiche :

   1) sur une même nouvelle ligne, les coordonnées des cases qui se trouvent dans la même colonne que la case indiquée par l'utilisateur mais au dessus de celle-ci (voir l'exemple d'affichage présenté plus loin) ;

   2) sur une autre ligne, le nombre de casses qui viennent d'être affichées ;

j. on affiche une ligne séparatrice formée "d'étoiles" (suite de caractères "astérisque") ;

k. si la cellule indiquée par l'utilisateur se trouve dans la première colonne de l'échiquier (c'est-à-dire dans la colonne ***A***), on affiche sur une nouvelle ligne le message **`Première colonne !`** et on passe au point **m.** ;

l. autrement, on affiche :

   1) sur une même nouvelle ligne, les coordonnées des cases qui se trouvent dans la même ligne que la case indiquée par l'utilisateur mais à gauche de celle-ci (voir l'exemple d'affichage présenté plus loin) ;

   2) sur une autre ligne, le nombre de cases qui viennent d'être affichées ;

m. on affiche une ligne séparatrice formée "d'étoiles" (suite de caractères "astérisque") ;

n. on demande à l'utilisateur s'il veut continuer l'exécution, en affichant le message **`Voulez-vous continuer ? (O/N)`** ;

o. on stocke la réponse de l'utilisateur ;

p. si l'utilisateur a répondu par la lettre **O** ou **o**, l'exécution revient au point **b.** ;

q. autrement, le programme affiche (une seule fois) le message **`Au revoir !`** et s'arrête.

Remarques :

- *d'une manière générale, on suppose que les données introduites par l'utilisateur respectent les types Java attendus dans l'application interactive ;*
- *le code Unicode du caractère **A** est 65 ;*
- *pour les point **i. 1)** et **l. 1)**, chaque couple colonne – ligne doit être entouré d'une paire de parenthèses et séparé du couple suivant par une tabulation (voir l'exemple d'affichage présenté ci-dessous) ;*
- *utilisez de manière appropriée les deux méthodes permettant d'afficher des messages à l'écran, à savoir **println** et **print** ;*
- *dans les chaînes de caractères à afficher, indiquez chaque espace par un signe spécifique, par exemple ␣ (ce qui donne pour le dernier message à afficher : **Au␣revoir␣!**).*

A l'exécution, la sortie du programme doit respecter la mise en page donnée dans l'exemple ci-dessous.

```
Message 1
Introduire la lettre correspondant à la colonne :
w
Colonne non valide !
Introduire la lettre correspondant à la colonne :
D
Introduire le numéro correspondant à la ligne :
10
Ligne non valide !
Introduire le numéro correspondant à la ligne :
-2
Ligne non valide !
Introduire le numéro correspondant à la ligne :
6
La case est noire !
****************************************
Les positions au-dessus : (D,7)   (D,8)
Il y a 2 case(s) au-dessus.
****************************************
Les positions à gauche : (C,6)    (B,6)   (A,6)
Il y a 3 case(s) à gauche.
****************************************
Voulez vous continuer ? (O/N)
```

```
O
Introduire la lettre correspondant à la colonne :
C
Introduire le numéro correspondant à la ligne :
8
La case est blanche !
****************************************
Dernière ligne !
****************************************
Les positions à gauche : (B,8)    (A,8)
Il y a 2 case(s) à gauche.
****************************************
Voulez vous continuer ? (O/N)
O
Introduire la lettre correspondant à la colonne :
A
Introduire le numéro correspondant à la ligne :
5
La case est noire !
****************************************
Les positions au-dessus : (A,6)   (A,7)   (A,8)
Il y a 3 case(s) au-dessus.
****************************************
Première colonne !
****************************************
Voulez vous continuer ? (O/N)
o
Introduire la lettre correspondant à la colonne :
A
Introduire le numéro correspondant à la ligne :
8
La case est blanche !
****************************************
Dernière ligne !
****************************************
Première colonne !
****************************************
Voulez vous continuer ? (O/N)
Z
Au revoir !
```

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....