

13 janvier 2016

**Contrôle d'informatique no 2**

Durée : 1 heure 45'

Nom : .....

Groupe : 

Prénom : .....

No	1.1	1.2	2
Total points	73 points	33 points	46 points

*Remarque générale : toutes les questions qui suivent se réfèrent au langage de programmation Java (à partir de Java SE 7) et les réponses doivent être rédigées à l'encre et d'une manière propre sur ces feuilles agrafées.*

**Sujet no 1.**

Le but de cet exercice est de réaliser une application autonome interactive qui aide l'utilisateur à manipuler des objets géométriques correspondant à des ellipses. Cette application doit correspondre à un projet doté d'un package appelé **cms\_ctr2** et qui contient deux classes **publiques**, définies dans deux fichiers distincts et appelées **Ellipse** et, respectivement, **CP\_Ctr2Exo1**. Par la suite, aux points **1.1** et **1.2**, on vous demande d'écrire le code complet de ces deux classes, en respectant les consignes précisées. Vous pouvez éventuellement répondre au point **1.2** en supposant le point **1.1** résolu correctement.

**1.1** Une instance de la classe **Ellipse** est un objet de type **Ellipse** qui regroupe (ou encapsule) des informations concernant une ellipse, à savoir ses deux demi-axes **a** et **b** (où **a** ne doit pas être forcément le grand demi-axe de l'ellipse). De plus, la classe **Ellipse** est munie des méthodes permettant de manipuler les objets de type **Ellipse**.

Écrire le code complet de la classe **publique Ellipse** qui appartient au package **cms\_ctr2** et qui définit :

- a) un champ (d'instance) nommé **a** de type numérique réel, déclaré **privé** et sans valeur initiale explicite et qui sert à stocker la valeur d'un demi-axe de l'ellipse ;
- b) un champ (d'instance) nommé **b** de type numérique réel, déclaré **privé**, avec la valeur initiale explicite **2.5** et qui sert à stocker la valeur de l'autre demi-axe de l'ellipse ;
- c) une méthode **publique** (d'instance) "**setter**" nommée en respectant la convention Java pour le nommage des setters et qui permet la modification de la valeur du champ privé **a** ; cette méthode doit respecter les consignes suivantes :
  - i. si la valeur de son argument noté lui aussi **a** est strictement positive, on copie la valeur de cet argument dans le champ **a** ;
  - ii. autrement, on stocke dans le champ **a** la valeur **10** ;
- d) une méthode **publique** (d'instance) "**setter**" nommée en respectant la convention Java pour le nommage des setters et qui permet la modification de la valeur du champ privé **b** ; cette méthode doit respecter les consignes suivantes :
  - iii. si la valeur de son argument noté **arg** est strictement positive, on copie la valeur de cet argument dans le champ **b** ;
  - iv. autrement, on stocke dans le champ **b** la même valeur que celle du champ **a** ;
- e) une méthode **publique d'instance** appelée **donnerDemiGrandAxe** qui n'a pas d'argument et qui retourne un résultat de type numérique réel; plus précisément, cette méthode détermine et retourne la valeur du grand demi-axe de l'ellipse correspondant à l'objet appelant ;
- f) une méthode **publique et statique** appelée **donnerDemiPetitAxe** qui a un seul argument de type **Ellipse** et qui retourne un résultat de type numérique réel; plus précisément, cette méthode détermine et retourne la valeur du petit demi-axe de l'ellipse correspondant à son argument ;

- g) un constructeur **public** (surchargé) avec deux arguments de type numérique réel : le premier argument nommé **a** (comme le champ **a**) et le deuxième argument nommé **b** (comme le champ **b**) ; ce constructeur permet de créer un objet correspondant à une nouvelle ellipse (en proposant comme premier argument la valeur d'un demi-axe et comme deuxième argument la valeur de l'autre demi-axe de la nouvelle ellipse) et doit respecter les consignes suivantes :
- i. la valeur de son premier argument **a** est "stockée" dans le champ **a** par un appel à la méthode setter correspondante ;
  - ii. la valeur de son deuxième argument **b** est "stockée" dans le champ **b** par un appel à la méthode setter correspondante ;
- h) un constructeur **public** (surchargé) sans argument et qui sert à créer un nouvel objet qui correspond à une ellipse dont un demi-axe vaut **1** et l'autre demi-axe vaut **2** ; plus précisément, il appelle le constructeur avec deux arguments en lui passant comme premier argument (effectif) la valeur numérique réelle **1** et comme deuxième argument (effectif) la valeur numérique réelle **2** ;
- i) un constructeur **public** (surchargé) avec un seul argument de type **tableau** (monodimensionnel) **de valeurs numériques réelles** ; ce constructeur doit respecter les consignes suivantes :
- i. si la valeur de son argument est la valeur spéciale **null** (c'est-à-dire si l'adresse correspondant à son argument n'est pas définie) ou si la taille de l'objet tableau correspondant à son argument est différente de **2**, on stocke dans le champ **a** (de l'objet créé) la valeur **10** et dans le champ **b** (de l'objet créé) la valeur **5** ;
  - ii. autrement, la valeur du premier élément du tableau argument est "stockée" dans le champ **a** (de l'objet créé) par un appel à la méthode setter correspondante et la valeur du deuxième élément du tableau argument est "stockée" dans le champ **b** (de l'objet créé) par un appel à la méthode setter correspondante ;
- j) une méthode **privée d'instance** nommée **calculerAire** qui permet de calculer l'aire d'une ellipse (selon la formule  $aire = \pi * a * b$  où **a** et **b** sont les demi-axes de l'ellipse) ; plus précisément, cette méthode n'a pas d'argument et retourne une valeur numérique réelle qui représente l'aire de l'ellipse correspondant à l'objet appelant ; en Java, la valeur du nombre  $\pi$  est disponible sous la forme du champ statique et final **PI** de la classe prédéfinie **Math** (du package **java.lang**) ;

- k)** une méthode **publique** et **statique** nommée **estCercle** qui permet de savoir si une ellipse est en fait un cercle ; plus précisément, cette méthode a un (seul) argument qui doit correspondre à une ellipse et retourne comme résultat une valeur de type logique ; plus précisément :
- i.** si les valeurs des deux demi-axes de l'ellipse correspondant à l'objet argument sont égales, la méthode retourne la valeur logique correspondant à **VRAI** ;
  - ii.** autrement, la méthode retourne la valeur logique correspondant à **FAUX** ;
- l)** une méthode **publique d'instance** nommée **encercler** qui permet de transformer une ellipse en un cercle correspondant à son grand demi-axe ; plus précisément, cette méthode n'a pas d'argument et ne retourne aucun résultat ; par contre, suite à l'appel de cette méthode, les valeurs des deux demi-axes de l'ellipse correspondant à l'objet appelant deviennent égales à son (ancien) grand demi-axe ;
- m)** une méthode **publique d'instance** nommée **comparer** qui permet de comparer les aires de deux ellipses ; plus précisément, cette méthode a un seul argument qui doit correspondre à une ellipse et retourne un résultat de type numérique entier ; par conséquent, la méthode **comparer** doit respecter les consignes suivantes :
- i.** si l'aire de l'ellipse correspondant à l'objet appelant est strictement plus grande que l'aire de l'ellipse correspondant à l'objet argument, la méthode retourne la valeur entière **-1** ;
  - ii.** (autrement) si l'aire de l'ellipse correspondant à l'objet appelant est égale à l'aire de l'ellipse correspondant à l'objet argument, la méthode retourne la valeur entière **0** ;
  - iii.** (autrement, c'est-à-dire si l'aire de l'ellipse correspondant à l'objet appelant est strictement plus petite que l'aire de l'ellipse correspondant à l'objet argument) la méthode retourne la valeur entière **1** ;
- n)** une méthode **publique** et **statique** nommée **informer** qui a un (seul) argument correspondant à une ellipse et qui retourne une valeur de type chaîne de caractères qui correspond au message suivant : *Une ellipse d'axes a=AAA et b=BBB !* où **AAA** est la valeur du champ **a** de l'objet argument et **BBB** est la valeur du champ **b** de l'objet argument.

[illegible]

[illegible]

[illegible]

This image shows a full page of a document template designed for handwriting practice or general note-taking. It consists of approximately 28 evenly spaced horizontal dotted lines across the entire width of the page. The background is plain white, and there are no margins, headers, footers, or other markings present.



**1.2** Ecrire le *code complet* de la classe publique **CP\_Ctr2Exo1** qui appartient au package **cms\_ctr2** et qui contient la méthode **main**. Cette classe réalise la partie interactive du projet (voir aussi l'exemple d'affichage à l'exécution présenté plus loin) et utilise la classe **Ellipse**.

Plus précisément, dans la méthode **main** :

- a) on déclare (et, le cas échéant, on initialise) les variables (locales) dont on a besoin pour réaliser le dialogue avec l'utilisateur ;
- b) on déclare deux variables locales de type numérique réel nommées *x* et *y* ;
- c) on affiche à l'écran un message qui demande à l'utilisateur d'introduire la valeur d'un demi-axe de l'ellipse ;
- d) on récupère la valeur introduite par l'utilisateur et on la stocke dans la variable *x* ;
- e) on affiche à l'écran un message qui demande à l'utilisateur d'introduire la valeur de l'autre demi-axe de l'ellipse ;
- f) on récupère la valeur introduite par l'utilisateur et on la stocke dans la variable *y* ;
- g) on déclare 3 variables (objets) locales de type **Ellipse** nommées *e1*, *e2* et *e3* ;
- h) on crée un nouvel objet correspondant à une ellipse dont les deux demi-axes sont égaux aux valeurs des variables *x* et de *y* et on stocke son adresse dans la variable *e1* ;

*Remarque : pour le point i) ci-dessous, il faut utiliser la méthode **estCercle** de la classe **Ellipse**.*

- i) si l'ellipse correspondant à la variable objet *e1* est en fait un cercle, on affiche à l'écran le message **Eureka !** ;
- j) par un appel au constructeur sans argument de la classe **Ellipse**, on crée un nouvel objet de type **Ellipse** et on stocke son adresse dans la variable *e2* ;
- k) par un appel convenable de la méthode **encerler** de la classe **Ellipse**, on transforme l'ellipse correspondant à la variable objet *e2* en cercle ;
- l) à l'aide d'un appel convenable de la méthode **informer** de la classe **Ellipse**, on affiche à l'écran les valeurs des demi-axes de l'ellipse correspondant à la variable objet *e2* ;
- m) on crée un nouvel objet correspondant à une ellipse dont les deux demi-axes valent 5 et, respectivement, 3 et on stocke son adresse dans la variable objet *e3* ;

*Remarque : pour le point n) ci-dessous, il est convenable d'utiliser la méthode **comparer** de la classe **Ellipse**.*

- n) si l'aire de l'ellipse correspondant à la variable objet *e2* est égale à l'aire de l'ellipse correspondant à la variable objet *e3*, on affiche à l'écran le message **Au revoir !** ; autrement, on affiche à l'écran le message **Adieu !**



[illegible]

## Sujet no 2.

Préciser les messages qui seront affichés à l'écran suite à l'exécution du projet contenant les deux classes suivantes (qui appartiennent à un même package).

```
package cms_ctr2;
class Maison
{
    int nbPieces;
    String adresse;
    static int i = -5;

    Maison(Maison m){
        this(m.nbPieces + 1, m.adresse);
        if(i < 0)
            System.out.println("Formare !");
    }

    Maison( ){
        this(5);
        adresse = "Neverland";
        if(i < 0)
            System.out.println("Aedificare !");
    }

    Maison(int n, String s){
        i++;
        if(i < 0)
            System.out.println("Facere !");
        nbPieces = n;
        adresse = s;
    }

    Maison(int n){
        this(n, "Pays imaginaire");
        if(i < 0)
            System.out.println("Construere !");
    }

    void mettreDaccord(Maison m){
        nbPieces = m.nbPieces;
        m.nbPieces = nbPieces;
        m.adresse = adresse;
        adresse = m.adresse;
    }

    static void agrandir(Maison m, int delta){
        m.nbPieces = m.nbPieces + delta;
    }

    void demenager(String s){
        adresse = s;
    }
}
```

```

    Maison chambouler(Maison mm){
        Maison.agrandir(this, mm.nbPieces);
        this.demenager(mm.adresse);
        return this;
    }

    void afficher( ){
        System.out.println("Maison de " + nbPieces +
                           " chambres au " + adresse + "." );
    }
} //fin de la classe Maison

public class CP_Ctr2Exo2 {
    public static void main(String[] args)
    {
        System.out.println("*** Constructions ***");
        Maison m1 = new Maison(3, "Pays de Nulle Part");
        System.out.println("-----");
        Maison m2 = new Maison(7);
        System.out.println("-----");
        Maison m3 = new Maison();
        System.out.println("-----");
        Maison m4 = new Maison(m1);
        System.out.println("-----");
        m1.afficher();
        m2.afficher();
        m3.afficher();
        m4.afficher();

        System.out.println("*** Déménagements ***");
        m1 = m2;
        m2 = m3;
        m4 = m1;
        m1.afficher();
        m2.afficher();
        m3.afficher();
        m4.afficher();

        System.out.println("*** Harmonisations ***");
        Maison m5 = new Maison(5, "Pays d'Enhaut");
        Maison m6 = new Maison(9, "Pays du Feu");
        m5.mettreDaccord(m6);
        m5.afficher();
        m6.afficher();

        System.out.println("*** Chamboulements ***");
        Maison m7 = new Maison(2, "Pays d'En Bas");
        Maison m8 = new Maison(4, "Pays des jouets");
        Maison m9 = new Maison();
        m9 = m7.chambouler(m8);
        m7.afficher();          m8.afficher();          m9.afficher();
    } //fin de la méthode main
} //fin de la classe principale

```

[illegible]

This image shows a full page of a handwriting practice worksheet. It consists of approximately 28 horizontal rows. Each row is defined by two parallel dotted lines, creating a series of uniform gaps for letter height. The entire page is otherwise blank, with no margins, text, or other markings.

A part ce texte, cette page doit rester normalement blanche (mais vous pouvez l'utiliser pour la rédaction de vos réponses si la place prévue à cet effet s'est avérée insuffisante).