

1 avril 2009

Contrôle d'informatique no 3

Durée : 1 heure 45'

Nom :

Groupe :

Prénom :

| No | 1 | 2 | 3 |
|---------------|-----------|-----------|------------------|
| Nombre points | 35 points | 29 points | (26 + 10) 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.

On considère un projet Java contenant le package *cms_ctr3* muni du fichier *CP_Ctr3Exo1.java*, dont le contenu est présenté par la suite (voir les pages 2 et 3). Dans ce fichier, on définit les classes *Mammifere*, *Cheval* et *Chat* ainsi que la classe "principale" publique *CP_Ctr3Exo1*. On vous demande d'écrire (aux pages 4 et 5) les résultats affichés à l'écran suite à l'exécution de ce projet.

```

package cms_ctr3;

abstract class Mammifere
{
    int age;

    Mammifere( )
    {
        this(3);
        System.out.println("Un mammifère de plus !");
    }

    Mammifere(int age)
    {
        this.age = age;
        System.out.println("Encore un mammifère !");
    }

    abstract void dormir( );

    String sePresenter( )
    {
        System.out.println("Bonjour !");
        return "Je suis un gentil animal !";
    }
} //fin de la classe Mammifere

class Cheval extends Mammifere
{
    double vitesse;

    Cheval( )
    {
        System.out.println("Voici un cheval !");
    }

    Cheval(double vitesse)
    {
        this( );
        this.vitesse = vitesse;
        System.out.println("Un cheval qui court !");
    }

    Cheval(int arg1, double arg2)
    {
        super(arg1);
        vitesse = arg2;
        System.out.println("Un cheval complet !");
    }

    void dormir( )
    {
        System.out.println("Je dors debout !");
    }

    String sePresenter( )
    {
        if(vitesse < 50)
            return "Je suis un cheval lent !";
        else
            return "Je suis un cheval rapide !";
    }
} //fin de la classe Cheval

```

```

class Chat extends Mammifere
{
    boolean estRaye;

    void dormir( )
    {
        System.out.println("Je dors beaucoup !");
    }

    String sePresenter( )
    {
        return super.sePresenter( ) + "\nChat docile !";
    }

    String sePresenter(String nom)
    {
        super.sePresenter( );
        return "Mon nom est : " + nom;
    }
} //fin de la classe Chat

public class CP_Ctr3Exo1
{
    public static void main(String[] args)
    {
        Mammifere tab[] = new Mammifere[4];
        tab[0] = new Cheval(30);
        System.out.println("-----");
        tab[1] = new Cheval( );
        System.out.println("-----");
        tab[2] = new Cheval(5, 60);
        System.out.println("-----");
        tab[3] = new Chat( );
        System.out.println("-----");
        System.out.println("*****");
        for(int i=0; i<tab.length; i++)
        {
            tab[i].dormir( );
            System.out.println(tab[i].sePresenter( ));
            System.out.println("Age : " + tab[i].age + ".");
            if(tab[i] instanceof Cheval)
            {
                System.out.println("Vitesse: " +
                    ((Cheval)tab[i]).vitesse + ".");
                System.out.println
                    (((Cheval)tab[i]).sePresenter());
            } else
            {
                ((Chat)tab[i]).sePresenter("Minou");
                if(((Chat)tab[i]).estRaye)
                    System.out.println("Chat rayé !");
                else
                    System.out.println("Chat non rayé !");
            }
            System.out.println("-----");
        } //fin de la boucle for
        System.out.println("*****");
    } //fin de la méthode main
} //fin de la classe principale CP_Ctr3Exo1

```

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.

[illegible]

Sujet no 2.

On considère un projet Java contenant le package *cms_ctr3* muni du fichier *CP_Ctr3Exo2.java*, dont le contenu est présenté ci-dessous. Dans ce fichier, on définit les classes *LongueurException* et *PointException* ainsi que la classe "principale" publique *CP_Ctr3Exo2*. On vous demande d'écrire (aux pages 8 et 9) les résultats affichés à l'écran suite à l'exécution de ce projet.

```
package cms_ctr3;

class LongueurException extends RuntimeException
{
    String strExp;
    LongueurException(String arg)
    {
        super("Longueur non valide !");
        strExp = arg;
        System.out.println("Il y a un problème !");
    }
} //fin de la classe LongueurException

class PointException extends Exception
{
    String strExp;
    PointException(String arg)
    {
        super("Point mal placé !");
        strExp = arg;
        System.out.println("On a des soucis !");
    }
} //fin de la classe PointException

public class CP_Ctr3Exo2
{
    public static String controler(String arg)
    {
        try
        {
            if(arg.length() != 11)
                throw new LongueurException(arg);
            if(arg.charAt(3) != '.')
                throw new PointException(arg);
        } catch (PointException e)
        {
            System.out.println(e.getMessage());
            controler(e.strExp.substring(0,3) + '.'
                        + arg.substring(4));
        } catch (LongueurException ex)
        {
            System.out.println(ex.getMessage());
            if(ex.strExp.length() == 0)
                return "021.6931111";
            else
                throw ex;
        } finally
        {
            System.out.println("Numéro contrôlé !");
        }
        return arg.substring(4);
    } //fin de la méthode controler
}
```

```

public static void main(String[] args)
{
    String tab[ ] = {"021.6939999",
                     "31.6318111",
                     "044.12345678",
                     "02298765432",
                     ""};

    for(int i=0; i<tab.length; i++)
    {
        try
        {
            tab[i] = controler(tab[i]);
        } catch(LongueurException ex)
        {
            System.out.println("A nouveau : "
                               + ex.getMessage());

            if(ex.strExp.length( ) > 11)
            {
                tab[i] =
                    controler(tab[i].substring(0,11));
            } else
            {
                tab[i] = "021.6932295";
            }
        }

        System.out.println("-----");
    } //fin de la première boucle for

    for(int i=0; i<tab.length; i++)
    {
        System.out.println("-----");
        System.out.println(tab[i]);
    } //fin de la deuxième boucle for
    } //fin de la méthode main
} //fin de la classe principale CP_Ctr3Exo2

```

This image shows a full page of primary-ruled notebook paper. It features multiple sets of horizontal lines designed to guide young learners' handwriting. Each set consists of three lines: a solid top line, a dashed middle line, and a dotted bottom line. These sets are repeated down the entire page, providing ample space for practicing letter formation and alignment. The paper is otherwise blank, with no margins, text, or illustrations.

[illegible]

Sujet no 3.

On considère un projet Java contenant le package *cms_ctr3* muni de fichiers correspondant, respectivement, à une interface et à plusieurs classes, à savoir :

- l'interface *IComparable* qui contient la méthode *comparer* ;
- la classe *PBD* qui permet l'instanciation d'objets correspondant à des points dans le plan précisés par leurs abscisses et leurs ordonnées cartésiennes ;
- la classe abstraite *Polygone* qui sert de classe de base pour la classe *Triangle* ;
- la classe *Triangle* qui est une classe dérivée (classe fille) de la classe *Polygone* et qui permet l'instanciation d'objets correspondant à des triangles précisés par leurs trois sommets de type *PBD* stockés dans un champ propre de type tableau de *PBD* ;
- la classe "principale" *CP_Ctr3Exo3* qui contient la méthode *main*.

On vous présente ci-dessous les codes (complets et sans fautes) de l'interface *IComparable* et des classes *PBD* et *Polygone*.

```
package cms_ctr3;

public interface IComparable
{
    int comparer(Object o);
} //fin de la l'interface IComparable

package cms_ctr3;

public class PBD
{
    double x, y ;

    PBD(double x, double y)
    {
        this.x = x;
        this.y = y;
    }

    public PBD creerClone( )
    {
        return new PBD(x,y);
    }

    public double calcDist(PBD arg)
    {
        return Math.sqrt((x-arg.x)*(x-arg.x)
                           +(y-arg.y)*(y-arg.y));
    }
} //fin de la classe PBD
```

```

package cms_ctr3;

public abstract class Polygone
{
    protected double perimetre;

    public double getPerimetre( )
    {
        return perimetre;
    }

    abstract double calculerPerimetre( );
} //fin de la classe Polygone

```

3.1

On vous donne par la suite le canevas de la classe publique *Triangle* qui doit :

- dériver de la classe de base (classe mère) *Polygone* ;
- implémenter l'interface *Comparable*.

On vous demande de compléter, aux endroits prévus à cet effet et indiqués par des commentaires :

- la déclaration du package *cms_ctr3* ;
- l'en-tête de la classe *Triangle* ;
- l'en-tête et le corps de la méthode redéfinie *calculerPerimetre* ;
- l'en-tête et le corps de la méthode redéfinie *comparer*.

Plus précisément :

- la méthode d'instance *calculerPerimetre* doit retourner la valeur réelle du périmètre du triangle correspondant à son objet appelant ;
- la méthode d'instance *comparer* doit retourner la valeur entière :
 - o **666** si le type de son objet argument n'est pas (exactement) le même que le type de son objet appelant ;
 - o **-1** si le périmètre de son objet appelant est plus petit que le périmètre de son objet argument ;
 - o **0** si le périmètre de son objet appelant est égal au périmètre de son objet argument ;
 - o **1** si le périmètre de son objet appelant est plus grand que le périmètre de son objet argument.

```
//Déclarer le package
```

```
//Préciser l'en-tête de la classe Triangle
```

```
{    //rien à ajouter
    PBD s[] = new PBD[3];

    //rien à ajouter
    public Triangle(PBD s1, PBD s2, PBD s3)
    {
        s[0] = s1;
        s[1] = s2;
        s[2] = s3;
        perimetre = calculerPerimetre( );
    }

    @Override
    //Redéfinir la méthode calculerPerimetre
    //(en-tête et corps)
```

```
@Override
//Redéfinir la méthode comparer
//(en-tête et corps)
```

This image shows a full page of white paper with horizontal dotted lines. The lines are evenly spaced and run across the width of the page, providing a guide for handwriting practice. There are no margins, text, or other markings on the page.

3.2

On vous donne ci-dessous le code de la classe principale du projet *CP_Ctr3Exo3*.

```
package cms_ctr3;

public class CP_Ctr3Exo3
{
    public static void main(String[ ] args)
    {
        PBD a = new PBD(0,0);
        PBD b = new PBD(3,0);
        PBD c = new PBD(0,3);
        PBD e = new PBD(1,1);
        PBD f = new PBD(7,1);
        PBD g = new PBD(1,9);

        Polygone tr1 = new Triangle(a, b, c);
        Polygone tr2 = new Triangle(e, f, g);

        c.y = 4;

        System.out.println(tr1.calculerPerimetre( ));
        System.out.println(tr2.getPerimetre( ));
        System.out.println(((Triangle)tr1).comparer(tr2));
        System.out.println(((Triangle)tr1).comparer("Toto"));
    } //fin de la méthode main
} //fin de la classe principale CP_Ctr3Exo3
```

[illegible]