UNIVERSITE CHEIKH ANTA DIOP DE DAKAR

ECOLE SUPERIEURE POLYTECHNIQUE

DEPARTEMENT INFORMATIQUE



M. FALL && M. TOURE

**DIC1**
**Java**
**TP N° 2 : Récursion et POO en Java**

# PARTIE A : Récursion

## Exercice 1

Créer une classe ***Recursion*** qui contient les méthodes *statiques* (donc pas besoin d'instances pour les invoquer) et *récursives* ci-dessous :

- ***public static double factorial (double num)***
  // return the factorial of num
- ***public static double average (double [] inArray, int size);***
  // return the average of size numbers in inArray
- ***public static double findlargest (double [] inArray, int size);***
  // return largest int in inArray
- ***public static int binarysearch (double [] inArray, double val, int low, int high);***
  // searches for index of val in inArray between low and high
- ***public static void permute (char [] inArray, int size)***
  // compute and display all permutations of an array of characters
- Ecrire une fonction ***main*** pour tester toutes ces fonctions (voir ***figure***)

```
115  public static void main(String[] args) {
116      //double fact = 6.0;
117      System.out.println("Factorial (6) is : "+factorial(6));
118      double [] array = {1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0};
119      char [] inarray = {'S', 'A', 'T'};
120      System.out.println("Average of numbers is : "+average(array, array.length));
121      System.out.println("Largest of numbers is : "+findlargest(array, array.length));
122      System.out.println("Binary Search of the index of a given number is : "+binsearc
123      System.out.println("All permutations for an array of char: ");
124      permute(inarray,inarray.length);
125
126
```

Problems @ Javadoc Declaration Console

```
<terminated> Recursion [Java Application] C:\Program Files\Java\jre1.8.0_91\bin\javaw.exe (26 mai 2017 à 13:51:43)
Factorial (6) is : 720.0
Average of numbers is : 4.5
Largest of numbers is : 8.0
Binary Search of the index of a given number is : 0
All permutations for an array of char:
SAT
AST
STA
TSA
TAS
ATS
```

# PARTIE B : POO

## Exercice 1

Ecrivez une classe Point avec les attributs (privés) suivants :

- $\boldsymbol{x}$: abscisse du point,
- $\boldsymbol{y}$: ordonnée du point.

La classe Point doit disposer des constructeurs suivants :

- *Point();*
- *Point(double, double);*
- *Point(Point);*

La classe Point doit contenir ces méthodes ci-dessous :

- *public void setX (double) ;*
- *public void setY (double) ;*
- *public double x () ;*
- *public double y () ;*
- *public double distanceFromPoint(Point);*
- *private void updateDistance () ; Recalcule la distance d'un point à l'origine à chaque fois les coordonnées sont modifiées.*
- *public static double distanceTwoPoint(double, double, double, double)*
- *public void translater(double, double);*
- *public void rotate(double ⊖); utiliser les coordonnées polaires comme suit :*

$$\boldsymbol{x_2 = x_1 cos \ominus - y_1 sin \ominus}$$
$$\boldsymbol{y_2 = x_1 cos \ominus + y_1 sin \ominus}$$

- *public String toString(); Affiche :* $\boldsymbol{p_1 = (x, y)}$
- Ecrivez aussi une classe *TestPoint* afin de tester la classe *Point*.

## Exercice 2

Ecrire une classe *Droite.*

***RAPPEL***

Une droite est un ensemble infini de points. Si on dispose de deux points

$p_1(x_1, y_1)$ *et* $p_2(x_2, y_2)$, alors, il n'existe qu'une et une seule droite passant par ces deux points.

$$m = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$$

La pente *m*, mesure l'inclinaison de la droite. Elle correspond à la variation de la valeur de y lorsque x augmente d'une unité. Graphiquement, elle exprime la variation verticale de la droite pour un déplacement horizontal d'une unité positive. Une droite (D) est aussi représentée par l'équation : $y = mx + b$

L'ordonnée à l'origine, qui est représentée par la lettre b, est la valeur de y lorsque x est zéro. Il s'agit donc de la position de la droite lorsque celle-ci croise l'axe des y. si la droite est parallèle à l'axe des ordonnées, on dit parfois que sa pente est infinie, ou plus rigoureusement, que sa pente n'est pas définie.

On veut créer la classe nommée Droite avec comme attributs :

- $\boldsymbol{p_0}$ : un point appartenant à la droite ;
- ***m*** : la pente de la droite (***slope*** en anglais);

Il est demandé d'ajouter les constructeurs suivants à la classe Droite :

- *Droite (Point, double) ;*
- *Droite (Point, Point) ;*
- *Droite (double a, double b) ; m sera calculée avec les points* $p_1(a,0)\ et\ \ p_2(0,b)$.

Rajoutez les méthodes suivantes :

- *public double slope () ;*
- *public double xIntercept() ;*
- *public double yIntercept() ;*
- *public boolean equals(Droite);*
- *public boolean isHorizontal() ;*
- *public boolean isVertical() ;*
- *private void normalize() {*

  *if (isHorizontal())P0 = new Point(0,yIntercept());*

  *else if (isvertical()) P0 = new Point(xIntercept(),0);*

  *else.if (yIntercept0 == 0) P0 = new Point(1,m);*

  *else P0 = new Point(0,yIntercept());*

  *}. Cette méthode est invoquée dans toutes les méthodes qui modifient l'objet.*
- *public boolean isParallelTo(Droite);*
- *public boolean isPerpendicularTo(Droite);*

- *public void translate(double, double) ; Déplace tous les points de la droite ;*
- *public String tostring() ; D = ax+b, prenant compte le cas où a = 0 ou b = 0*
- Ecrivez aussi une classe ***TestDroite*** afin de tester la classe ***Droite***.

Exercice 3

Ecrire la classe Fraction, ainsi définie :

Champs :

- *private long numerator;*
- *private long denominator;*

Constructeurs :

- *public Fraction () ;*
- *public Fraction (long, long);*
- *public Fraction (long);*
- *public Fraction (String) ;*

Méthodes :

- *public void setNumerator (long num);*
- *public void setDenominator (long denom);*
- *public long numerator ();*
- *public long denominator ();*
- *public Fraction add (Fraction f);*
- *public Fraction subtract (Fraction f) ;*
- *public Fraction multiply (Fraction f) ;*
- *public Fraction divide (Fraction f) ;*
- *public boolean equals (Fraction f); Remarque:* $\frac{2}{-3} = \frac{-2}{3} = \frac{-4}{6}$

- ***private void stringToFraction (String fString);***

  *Appelée dans public Fraction (String fString) ;*

- ***private Fraction simplify () ;***
- ***public int intValue () ;***
- ***public long longValue () ;***
- ***public float floatValue () ;***
- ***public double doubleValue () ;***
- ***public String toString () ;***
- ***public int hashCode () ;***
- *Ecrivez aussi une classe* ***TestFraction*** *afin de tester la classe* ***Fraction****.*