Université Hassan II
Faculté des Sciences Ain Chock – Casablanca
Département Mathématique et Informatique

Série de TD N° 2 **Déclaration des classes et instanciation des objets**	**Licence Professionnelle** Développement Informatique POO en Java

Objectif

Apprendre à créer des classes, définir des attributs et des méthodes, et instancier des objets en Java.

Exercice 1 :

1. Déclarer une classe **Voiture** ayant comme attributs : marque(String), modele(String) et annee(int).
2. Ajouter un constructeur « public Voiture(String marque, String modele, int annee) »avec trois paramètres pour initialiser les attributs.
3. Ajouter à cette classe la méthode « public void infos() » qui affiche les informations sur la voiture.
4. Ajouter à cette classe la méthode «public void demarrer() » qui permet d'afficher le message « Voiture démarrée).
5. Ajouter à cette classe la méthode « public void arreter() » qui permet d'afficher le message « Voiture arrêtée).
6. Dans une classe Principale ajouter la fonction principale main et ajouter le code pour créer deux voitures : v1(Ferrari, Portofina, 2023) et v2(Dacia, Logan, 2010). Afficher les informations des deux voitures puis démarrer et arrêter la Ferrari.
7. Modifier la classe Voiture en ajoutant un attribut booléen « **started** » initialisé par false.
8. Modifier les méthodes « demarrer() » et « arreter() » de tel sort qu'on ne peut pas démarrer une voiture déjà démarrée ou arrêter une voiture non pas encore démarrée.

Correction :

La classe Voiture :

```
public class Voiture {

        private String marque;
        private String modele;
        private int annee;
        private boolean started = false;

        public Voiture(String marque, String modele, int annee) {
```

```
                this.marque = marque;
                this.modele = modele;
                this.annee = annee;
        }

        public void infos() {
                System.out.println("Marque :"+marque+" ; Modèle:"+modele+ " ;
Année:"+annee);
        }

        public void demarrer() {
                if(started) {
                        System.out.println("Voiture déja démarrée!");
                } else {
                        started = true;
                        System.out.println("Voiture démmarée.");
                }
        }

        public void arreter() {
                if(!started) {
                        System.out.println("Voiture déja arrêtée!");
                } else {
                        started = false;
                        System.out.println("Voiture arrêtée.");
                }
        }
}
```

La classe pour tester :

```
public class App {
        public static void main(String[] args) {
                Voiture v1= new Voiture("Ferrari", "Portofina", 2023);
                Voiture v2= new Voiture("Diacia", "Logan", 2010);

                v1.infos();
                v2.infos();
                v1.demarrer();
                v1.arreter();
        }
}
```

Exercice 2 :

1. Déclarez une classe **Point** avec deux attributs : x et y, de type double.
2. Ajoutez un constructeur par défaut à la classe **Point** pour initialiser x et y à zéro.

3. Ajoutez un constructeur paramétré qui permet d'initialiser x et y avec des valeurs fournies.
4. Implémentez des méthodes d'accès (getters) et de modification (setters) pour les attributs x et y.
5. Ajoutez une méthode **distance(Point p)** qui calcule et retourne la distance euclidienne entre le point courant (this) et un autre point p passé en paramètre.
6. Ajoutez une méthode **deplacer(double dx, double dy)** pour déplacer le point en ajoutant les valeurs dx et dy respectivement à x et y.
7. Déclarez une classe **Segment**, représentant un segment défini par deux points a et b, correspondant aux extrémités du segment.
8. Ajoutez un constructeur à la classe **Segment** pour initialiser le segment avec les deux points a et b.
9. Implémentez une méthode **longueur()** dans la classe Segment, qui renvoie la longueur du segment.

La classe Point :

```
public class Point {

	private double x,y;

	public Point() {
		this.x = 0;
		this.y = 0;
	}
	public Point(double x, double y) {
		this.x = x;
		this.y = y;
	}
	public double getX() {
		return x;
	}
	public double getY() {
		return y;
	}
	public void setX(double x) {
		this.x = x;
	}
	public void setY(double y) {
		this.y = y;
	}
	public void deplacer(double dx, double dy) {
		this.x += dx;
		this.y += dy;
	}

	public double distance(Point p) {
```

```
            return Math.sqrt(Math.pow(p.getX()-x,2)+Math.pow(p.getY()-y,2));
        }

}
```

La classe Segment :

```
public class Segment {

        private Point a,b;

        public Segment(Point x, Point y) {
            this.a = x;
            this.b = y;
        }
        public double longueur() {
            return a.distance(b);
        }

}
```

La classe pour tester :

```
public class App {

        public static void main(String[] args) {

            Point p1, p2;

            p1 = new Point(2, 1);
            p2 = new Point(3, 1);

            Segment seg = new Segment(p1, p2);

            System.out.println(seg.longueur());

        }

}
```

Exercice 3 :

Imaginez un jardin virtuel dans lequel l'utilisateur doit s'occuper de plusieurs plantes virtuelles pour les maintenir en bonne santé. Une plante peut grandir, a besoin d'eau pour survivre, et dépérit si elle n'est pas correctement arrosée.

L'objectif est de coder un programme simulant une petite colonie de plantes, chacune avec son propre niveau de croissance et d'hydratation. L'utilisateur pourra interagir avec les plantes pour les arroser et suivre leur état de croissance.

1. Déclarez une classe Plante avec les attributs privés suivants :

   nom de type String : le nom de la plante.
   croissanceMax de type int: représente le niveau de croissance maximale.
   croissance de type int : indique le niveau de croissance actuel de la plante.
   hydratation de type int : indique le niveau d'hydratation de la plante.

2. Le constructeur prend en paramètre une chaîne de caractères pour le nom de la plante et initialise les autres attributs de manière aléatoire :

   croissanceMax entre 8 et 12.
   croissance entre 1 et 3.
   hydratation entre 3 et 5.

3. Écrivez une méthode de signature **public void statut()** qui affiche le nom de la plante, son niveau de croissance actuel et son état :

   "Bien hydratée" si l'attribut hydratation est supérieur ou égal à 5.
   "A soif" si hydratation est inférieur à 5.

4. Écrivez une méthode **public void arroser()** qui augmente le niveau d'hydratation d'une valeur aléatoire entre 1 et 2. Affichez ensuite un message de satisfaction. Si la plante est déjà bien hydratée (niveau hydratation >= 5), affichez un message indiquant qu'elle n'a pas soif.

5. Écrivez une méthode **public void grandir()** qui augmente le niveau de croissance de la plante de 1 seulement si elle est bien hydratée. Si le niveau de croissance atteint croissanceMax, affichez un message indiquant que la plante a atteint sa taille maximale.

6. Tester la classe dans une fonction main.

Correction :

```
import java.util.Random;

public class Plante {

    private String nom;
    private int croissanceMax;
    private int croissance;
    private int hydratation;
    private Random rand = new Random();

    public Plante(String nom) {
```

```
        this.nom = nom;
        this.croissanceMax = rand.nextInt(5) + 8;
        this.croissance = rand.nextInt(3) + 1;
        this.hydratation = rand.nextInt(3) + 3;
    }

    public void statut() {
        System.out.println("Nom: " + nom);
        System.out.println("Croissance actuelle: " + croissance + "/" + croissanceMax);
        if (hydratation >= 5) {
            System.out.println("État: Bien hydratée");
        } else {
            System.out.println("État: A soif");
        }
        System.out.println("--------------------------");
    }

    public void arroser() {
        if (hydratation >= 5) {
            System.out.println(nom + " n'a pas soif");
        } else {
            int ajoutHydratation = rand.nextInt(2) + 1;
            hydratation += ajoutHydratation;
            System.out.println(nom + " est arrosée et satisfaite !");
        }
        System.out.println("--------------------------");
    }

    // Méthode pour faire grandir la plante
    public void grandir() {
        if (hydratation >= 5) {
            if (croissance < croissanceMax) {
                croissance++;
                hydratation--; // La plante utilise de l'hydratation pour grandir
                System.out.println(nom + " a grandi ! Niveau de croissance: " +
croissance);
            } else {
                System.out.println(nom + " a atteint sa croissance maximale.");
            }
        } else {
            System.out.println(nom + " a soif et ne peut pas grandir.");
        }
        System.out.println("--------------------------");
    }

}
```