POO : Séance 4
Héritage

1èr semestre – Année universitaire 2024/2025

Rappelez-vous

- Une classe est un modèle pour des objets (ensemble d'objets) ayant les mêmes attributs et les mêmes méthodes.

- Un objet est une instance de classe. L'instanciation est la création d'un objet à partir d'une classe.

- Le regroupement des données et des méthodes dans un objet ainsi que le masquage des données est appelé encapsulation.

- Les constructeurs sont des méthodes particulières dont l'objectif est de 'créer les objets' (en fait, effectuer des traitements initiaux).

- Dans une classe, la création de deux ou plusieurs méthodes dotées du même nom mais ayant des paramètres différents est appelée une surcharge de méthode.

- public et private sont des modificateurs d'accès pour les classes, les méthodes et les attributs.

La classe Point

Un point est caractérisé par :

Une position (abscisse, ordonnée)

On peut:

Initialiser l'abscisse et l'ordonné d'un point

Modifier l'abscisse ou l'ordonné d'un point

Déplacer un point

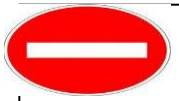
Calculer la distance d'un point par rapport à l'origine

La classe Point

```
Public class Point
{
        //attributs
          private int x;//abscisse du point
          private int y;//ordonnée du point
        //méthodes
        public Point (int abs,int ord)
        {
                x = abs;
                y = ord;
        }
        public void Set_abscisse(int abs)
        {
                x = abs;
        }
        public void Set_ordonne(int ord)
        {
                y =ord;
        }
        public void translate (int a, int b)
        {
                x = x + a;
                y = y + b;
        }
        public double distance ()
        {
                Return Math.sqrt(x*x + y*y);
        }

}
```

Utilisation

```
public class TestPoint
{
    //Méthode
    public static void main (String [] args)
    {
        Point P = new Point (4,8);
        P.x = 4; P.y = 2;
        P. Set_abscisse(5);
        P. Set_ordonne(9);
        P.translate(4,-2);
        System.out.println(" La distance séparant P de l'origine est " + P.distance());
    }
}
```



Déclaration + Instanciation d'un Point

x et y sont des attributs privés

L'abscisse de P est 5

L'ordonné de P est 9

L'abscisse de P est 9 et son ordonné est 7

La distance séparant P de l'origine est 11.4

La classe PointGraphique

Définir la classe PointGraphique.

Un PointGraphique est un Point caractérisé par :

- Abscisse
- Ordonnée
- Couleur

On peut:

- Initialiser l'abscisse et l'ordonné d'un PointGraphique
- Modifier l'abscisse ou l'ordonné d'un PointGraphique
- Déplacer un PointGraphique
- Calculer la distance d'un PointGraphique par rapport à l'origine
- Initialiser sa couleur
- Afficher sa couleur

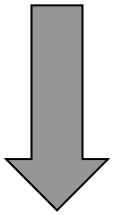
La classe PointGraphique

```
public class PointGraphique
{
    //attributs
     private int x;
     private int y;
     private String couleur;

    //méthodes
    public PointGraphique (int abs,int ord, String coul)
    {
            x = abs;
            y = ord;
            couleur = coul;
    }
    public void Set_abscisse(int abs)
    {
            x = abs;
    }
    public void Set_ordonne(int ord)
    {
            y =ord;
    }
```

```
public void translate (int a, int b)
{
        x = x + a;
        y = y + b;
}

public double distance ()
{
        Return Math.sqrt(x*x + y*y);
}

public void Set_couleur(String coul)
{
        couleur = coul;
}

public void afficher_Couleur()
{
        System.out.println(couleur);
}
```

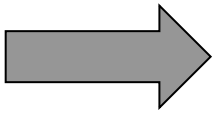
Héritage et extension

Pourquoi ne pas profiter de ce qu'offre la classe Point?



Définir la classe PointGraphique à partir de la classe Point

Héritage



On n'est pas obligé de réécrire les parties héritées

La classe PointGraphique peut aussi disposer de ses propres attributs et méthodes :

Extension

Point / PointGraphique

```
public class PointGraphique extends Point
{
    private String couleur; //couleur du point

    public PointGraphique (int abs,int ord, String coul)
    {
        x = abs;
        y = ord;
        couleur = coul;
    }
     public void Set_couleur(String coul)
    {
        couleur = coul;
    }
     public void afficher_Couleur()
    {
        System.out.println(couleur);
    }

}
```

La classe PointGraphique hérite de la classe Point

La classe PointGraphique Définit un nouvel attribut

La classe PointGraphique définit deux nouvelles méthodes

Héritage (1)

- Héritage : mécanisme permettant le partage et la réutilisation de propriétés entre les objets

- Une classe F hérite d'une classe M:

  La classe M est une classe mère (super-classe)

  La classe F est une classe fille (sous-classe)

  Les objets de F ont toutes les caractéristiques de M ainsi que celles définies dans F

Héritage (2)

- La classe PointGraphique possède tous les attributs (private et public) et toutes les méthodes (private et public) de la classe Point.

- Cependant, un objet de la classe PointGraphique n'a le droit d'utiliser directement que les attributs et les méthodes publiques de la classe Point

L'accès au attributs privés hérités ne se fera que via des méthodes publiques

Visibilité des membres

- Visibilité des membres de la classe mère dans la classe fille.

	Public	Private
Attribut	Oui	Non
Méthode	Oui	Non

Utilisation

```
public class TestPointGraphique
{
    //Méthode
    public static void main (String [] args)
    {
        PointGraphique G = new PointGraphique (5,10,'rouge');

        G. Set_abscisse(12);

        G. Set_ordonne(8);

        G.Set_couleur('vert');

        G.translate(0,-4);


        System.out.println(" La distance séparant G de l'origine est " + G.distance()
        + "  et la couleur de G est "  + G. afficher_Couleur());

    }
}
```

Déclaration + Instanciation d'un Point Graphique

L'abscisse de G est 12

L'ordonné de G est 8

La couleur de G est verte

G devient de coordonnées (12,4)

La distance séparant G de l'origine est 12.64 et la couleur de G est verte

Classe mère / Classe fille

- Une classe ne peut hériter (extends) que d'une seule classe.

Héritage simple

- Une classe peut être la mère de plusieurs classe dérivées.

La classe animal est la classe mère de:

La classe Mammifere La classe Oiseau La classe Reptile

La classe MoyenTransport est la classe mère de:

La classe Voiture La classe Bus
La classe Metro La classe Taxi

…

Niveaux d'héritage

- Pas de limitation dans le nombre des niveaux dans la hiérarchie d'héritage (A hérite de B, B hérite de C , …).

La classe Mammifere hérite de la classe Animal

public class Mammifere extends Animal

La classe Chien hérite de la classe Mammifere

public class Chien extends Mammifere

La classe EtreHumain

```
public class EtreHumain
{
	//Déclaration des attributs
	private String nom;
	private int age;
	private String profession;

	//Déclaration des méthodes
	public EtreHumain (String leNom, int lAge, String laProfession)
	{
		nom = leNom;
		age = lAge;
		profession = laProfession;
	}

	public void sePresenter ()
	{
	System.out.println("  Mon nom est "  + nom );
	System.out.println("  \n Mon age est "  + age);
	System.out.println("  \n Ma profession est "  + profession);
	}
}
```

La classe Etudiant

```
public class Etudiant extends EtreHumain
{

    private String filiere;

    public Etudiant(String leNom, int lAge, String laFiliere)
    {
        nom = leNom;
        age = lAge;
        profession = " Etudiant ";
        Filiere = laFiliere;
    }
    public Etudiant(String leNom, int lAge, String laFiliere)
    {
        super(leNom, lAge, " Etudiant ");
        filiere = laFiliere;
    }
    public String quelleFiliere()
    {
        return filiere;
    }

}
```

Appel du constructeur de la classe EtreHumain

La classe Enseignant

```
public class Enseignant extends EtreHumain
{

     private String specialite;

     public Enseignant(String leNom, int lAge, String laSpecialite)
     {
            super(leNom, lAge, "Enseignant");
            specialite = laSpecialite;
     }

    public String quelleSpecialite()
    {
           return specialite;
    }


 }
```

La classe EtudiantSportif

```
public class EtudiantSportif extends Etudiant
{

    private String sport;

    public EtudiantSportif (String leNom, int lAge, String laFiliere, String leSport)
    {
        super(leNom, lAge, laFiliere);
        sport = leSport;

    }
    public String quelSport()
    {
        return sport;
    }

}
```

Utilisation

```
public class test
{
 public static void main (String [] args)
          {

            EtreHumain E1 = new EtreHumain  (" Mohamed ", 37, " Medecin ");
            E1. sePresenter ();
            Etudiant E2 = new Etudiant (" Ali ", 19, " Gestion ");
            E2. sePresenter ();
            System.out.println(" \n  Ma filière est "  + E2.quelleFiliere() );
```

Mon nom est Mohamed
Mon age est 37
Ma profession est Medecin

Mon nom est Ali
Mon age est 19
Ma profession est Etudiant
Ma filière est Gestion

Utilisation (suite)

```
Enseignant E3 = new Enseignant (" Salah ", 46, " Comptabilité ");

E3. sePresenter ();

System.out.println(" \n  Ma spécialité est "  + E3. quelleSpecialite() );

EtudiantSportif E4 = new EtudiantSportif (" Sami ", 21, " Droit ", " Tennis ");

E4. sePresenter ();

System.out.println(" \n  Ma filière est "  + E4. quelleFiliere() );
System.out.println(" \n  Mon sport est "  + E4. quelSport() );
}
```

Mon nom est Salah
Mon age est 46
Ma profession est enseignant
Ma spécialité est comptabilité

Mon nom est Sami
Mon age est 21
Ma profession est Etudiant
Ma filière est Droit
Mon sport est Tennis

Constructeur (s) de la sous-classe

- Comme toutes les classes, si une sous-classe (ou une classe mère) ne définit pas de constructeur, elle en possède un par défaut.

- Le constructeur par défaut de la classe fille invoque le constructeur par défaut de la classe mère.

- Si la classe mère perd son constructeur par défaut, la classe fille perd aussi le sien. En fait, ce dernier continue à invoquer le constructeur par défaut de la classe mère. On est donc obligés de définir un nouveau constructeur pour la classe fille… Et que celui–ci appelle le constructeur de la mère.

Constructeur (s) de la sous-classe

- Pour construire un constructeur de la classe fille, on doit invoquer le constructeur de la classe mère en utilisant

Super (paramètres du constructeur de la super classe)

- Si l'appel de super est présent dans le constructeur de la classe fille, il doit toujours être la première instruction du corps du constructeur.

- Quand un objet est créé, les constructeurs sont invoqués en remontant la classe en classe dans la hiérarchie jusqu'à la classe Object

Questions (1)

❖ L'héritage permet de réutiliser des classes pour construire d'autres classes plus complètes.

Oui

❖ La classe fille posséde tous les attributs et les méthodes publiques et privés de la classe mère

Oui

❖ Un objet de la classe fille a le droit d'utiliser directement les attributs et les méthodes privés de la classe mère.

Non

❖ Une classe peut hériter d'une seule classe.

Oui

Questions (2)

❖ Une classe peut être la mère de plusieurs classe dérivées.

Oui

❖ Le constructeur par défaut de la classe fille invoque le constructeur par défaut de la classe mère.

Oui

❖ Un objet de la classe mère peut faire appel aux méthodes définies dans la classe fille.

Non

❖ L'appel de super doit toujours être la première instruction du corps du constructeur.

Oui

Héritage encore

A suivre…