JAVA-OBJET

Classe et objet

Une classe est un moule (un modèle) qui va servir à créer des exemples. C'est un ensemble de variables et de méthodes. Parmi les variables, on rencontre notamment les attributs dits les variables d'instance. Quant aux méthodes, ce sont des actions. Par exemple, la classe humain pourrait être définie par les attributs âge, poids, taille, date, etc, et par les actions manger, dormir, travailler, etc.

Remarques

- Les attributs d'une classe peuvent être de types de base ou des objets.
- Dans une classe, on peut ne pas avoir d'attributs.
- Dans une classe, on peut ne pas avoir de méthodes.
- Par convention, le nom d'une classe débute par une majuscule.

Un objet est un exemple d'une classe que l'on appelle également une instance d'une classe. Ainsi, chaque objet crée possèdera

- une identité ;
- son propre état, à savoir, des valeurs particulières pour les attributs de la classe auquel il appartient ;
- des méthodes qui vont agir sur son état.

Pour créer un objet obj instance d'une classe Clas, on utilise l'instruction

Clas obj = new Clas() ;

Remarque

new initialise automatiquement les attributs d'un objet à

- ✓ 0 pour les valeurs numériques ;
- ✓ false pour les booléens ;
- ✓ '\0' pour les caractères ;
- ✓ null pour les autres.

L'encapsulation des données ou droits d'accès

Java possède trois catégories d'autorisation d'accès :

public : aucune restriction ; Tout objet de tout autre classe peut y accéder.

private : accès réservé. Tout objet d'une autre classe ne peut y accéder.

protected : accès réservé. Tout objet d'une autre classe du même package peut y accéder, mais tout objet d'une autre classe d'un autre package ne peut y accéder.

Remarque

- ✓ Il est souhaitable que de déclarer les attributs private.
- ✓ Par défaut, les méthodes sont publiques.
- ✓ Si obj est un objet d'une classe C, A un attribut qui n'est pas d'accès private et M une méthode, alors l'instruction obj.A désigne l'attribut A de obj et l'instruction obj.M désigne la méthode M de obj.
- ✓ Si obj est un objet d'une classe C et si A est un attribut d'accès private, il n'est guère possible d'utiliser l'instruction obj.A

Exemple

Supposons que l'on veuille créer une nouvelle classe Personne qui ne contient pour le moment que des attributs. A cette fin,

- Créez un nouveau projet de nom **obj1**;

- Cliquez sur l'onglet **PROJET** puis sur **Ajouter une classe ...**

- Créez une classe de nom **Personne** qui ne comporte pas la méthode main.

Maintenant, complétez la classe Personne comme suit.

```
class Personne
    {
        // Afin d'eviter toute modification des attributs par les objets, il
       // est suhaitable de les definir via le mot clé private

        private String nom;
        private String prenom;
        private int age;
```

La surcharge de méthodes

- ✓ Le prototype d'une méthode est composé de
  - son nom ;
  - le type de chacun des arguments qu'elle reçoit ;
  - son type de retour.
- ✓ La signature d'une méthode est composée de
  - son nom ;
  - le type de chacun des arguments qu'elle reçoit.

- ✓ La surcharge de méthodes consiste à donner le même nom à des méthodes différentes dans une même classe.
  - Le type de retour n'intervient pas dans une méthode surchargée.
  - Les méthodes surchargées sont également dites polymorphes.
  - Il n'y a pas de limites au nombre de surcharges d'une même méthode.

Les constructeurs

Un constructeur est une méthode qui est automatiquement appelée par le compilateur lors de la création d'un objet.

Un constructeur n'est appelé qu'une seule fois dans la vie d'un objet.

Un constructeur sert notamment à initialiser les attributs de l'objet.

Un constructeur est public.

Bien que non indispensable, un constructeur permet la création d'un objet en précisant des valeurs initiales pour ses attributs.

Un constructeur obéit aux règles qui suivent

- son nom est impérativement celui de sa classe ;
- il ne doit point comporter de type de retour (int, void, etc);
- il ne doit pas comporter l'instruction return

Les constructeurs peuvent être surchargés.

Un constructeur sans arguments est dit constructeur par défaut.

Lorsqu'il existe un constructeur avec un ou des arguments, la construction d 'un objet par défaut nécessitera la présence d'un constructeur par défaut.

Lorsqu'une classe n'a aucun constructeur, elle en possède néanmoins un qui est un constructeur par défaut et qui consiste à initialiser les attributs de l'objet avec les valeurs par défaut.

Exemple

```
class Date
{
private int mois;
private int jour;
private int annee;
...
```

// on définit un constructeur

```
public Date(int j, int m, int a)
 {
jour=j; mois=m; annee=a;
}
```

```
// on peut définir d'autres constructeurs

public Date(int j, int m)
{
jour=j; mois=m; annee=2011;
}

public Date(int j)
{
jour=j; mois=10; annee=2011;
}

public Date()
{
jour=17; mois=11; annee=2011;}
}
```

Le mot clé this

Le mot clé this désigne l'objet courant, c.-à-d., l'objet sur lequel la méthode est appelée. On accède aux attributs d'une classe grâce au mot clé **this**.

Exemple

Maintenant, **complétez la classe Personne** en créant deux constructeurs comme suit et ce, après l'instruction `private int age;`.

```
        // On cree un premier constructeur qui va nous permettre d'instancier les attributs
        // on notera qu'un constructeur porte toujours le meme nom que la classe

         public Personne()
         {
             prenom = "Inconnu";
         }

        // On cree un deuxieme constructeur mais avec un nombre de parametres different.
        // On parle alors de constructeur surcharge ou plus generalement de methodes surchargees

         public Personne(String P, String N, int age)
         {
             prenom = P;
             nom = N;
             this.age = age;
         }
```

Les accesseurs et les mutateurs

Un accesseur est une méthode qui permet d'afficher les variables et dont le nom commence par get.

Un mutateur est une méthode qui permet de modifier les variables et dont le nom commence par set.

Les accesseurs sont du même type que la variable qu'ils doivent retourner.

Les mutateurs sont de type void.

Les accesseurs et mutateurs permettent à des objets d'utiliser les attributs d'accès private.

Exemple

```
class Date
{
private int mois;
private int jour;
private int annee;
...

public String getMois()
{
return mois;
}
public void setMois(int m)
{
mois = m;
}
}
```

Exemple

Maintenant, après avoir créer les constructeurs, on va créer les accesseurs et mutateurs. A cette fin, **complétez la classe Personne** comme suit.

```
 // Comme les attributs sont de type private, les objets ne peuvent les utiliser
 // via l'instruction obj.attribut.
// Maintenant, afin de pouvoir acceder aux attributs dans les objets puisqu'ils
//sont private, on Aussi, pour remedier aux probleme,
// on utilise les accesseurs (acces en lecture) et mutateurs (acces en ecriture)

// Maintenant on cree les accesseurs (acces en lecture) qui commencent par le
//mot get suivi de ce que vous voulez

 public String getPrenom()
 {
     return prenom;
 }

 public String getNom()
 {
     return nom;
 }

 public int getAge()
 {
     return age;
 }
```

```
// Maintenant on cree les mutateurs (acces en ecriture) qui
//commencent par le mot set suivi de ce que vous voulez

 public void setPrenom(String P)
 {
     this.prenom = P;
 }

 public void setNom(String N)
 {
     this.nom = N;
 }

 public void setAge(int age)
 {
     this.age = age;
 }
```

Maintenant, après avoir créer les accesseurs et mutateurs, on va créer des méthodes (comportements) communes à toute personne et qui doivent être d'accès public. A cette fin, **complétez la classe Personne** comme suit.

```
        // Maintenant on cree les actions ,c-a-d, les methodes
        //communes a toute personne et qui doivent etre d'acces public

        public void manger()
        {
            System.out.println("Je mange, entre autres, pour
continuer de vivre");
        }

        public void dormir()
        {
            System.out.println("Il faut dormir; \t" + "Le sommeil
regenere les cellules");
        }

        public String parler(String mot)
        {
                  String blabla = mot + "\nBravo vous etes parfait";
            return blabla;
        }
```

Maintenant, on souhaite tester notre projet. Pour ce faire,

- Créez une nouvelle classe de nom **Test** et qui comporte la procédure main qui sert de point d'entrée à l'application.

- Complétez la procédure main et ce, comme suit.

```java
public class Test
{

	public static void main(String[] args)
	{
		// On cree l'objet P0 qui est une instance de la classe Personne

		Personne P0 = new Personne();
		System.out.println("Le prenom de l'objet P0 est " + P0.getPrenom());
		System.out.println("P0=(" + P0.getPrenom() + "; " + P0.getNom() + " ;"
				+ P0.getAge() + ")");

		Personne P1 = new Personne("Trevor", "MonNom", 19);
		System.out.println("P1=(" + P1.getPrenom() + " " + P1.getNom() + " "
				+ P1.getAge() + ")");

		// on change l'age de l'objet P1 en lui affectant la valeur 21

		P1.setAge(21);
		System.out.println("L'age de P1 est devenu " + P1.getAge());

		// comme les methodes sont d'acces public, les objets peuvent
		// directement les utiliser et on peut ecrire objet.methode
		// Par contre, comme nos attributs sont private, alors on ne peut ecrire
		// objet.attribut. Dans ce cas, on utilise les
		// accesseurs et les mutateurs comme ci-dessus

		P1.manger();
		P1.dormir();
		System.out.println(P1.parler(P1.getPrenom()));

		Personne P2 = P1;
		if (P2.equals(P1))
			System.out.println("Mêmes Objets");

	}

}
```

Les variables de classe

Les variables de classe dites également variables statiques ou encore attributs statiques sont des variables globales à une même classe. Autrement dit, ce sont des variables partagées par tous les objets (exemples) d'une même classe. En fait, ce sont des variables qui ont la même valeur pour tous les objets. Par exemple, on peut citer le cas d'une variable qui compte le nombre d'objets instanciés (exemples) de la classe.

Les variables de classe, variables globales, bénéficient des accès public, private, protected et package.

Pour déclarer un variable globale, on utilise le terme static.

Remarques

- ✓ Lorsqu'une variable globale, varG, est en accès public, on peut l'appeler en dehors de sa classe, Class, soit
  - A travers un objet, obj, de cette classe en utilisant l'instruction obj.varG = « … »
  - A travers le nom de sa classe, Class, en utilisant l'instruction Class.varG = « … »
- ✓ • Les méthodes qui n'utilisent que des variables globales doivent elles aussi être déclarées static.

Les méthodes de classe

Les méthodes de classe dites également méthodes statiques sont des méthodes qui n'ont pas besoins d'objets pour être appelées. Elles concernent non pas un objet, mais la classe elle-même. Par exemple, l'instruction Math.sqrt(x) ou encore une méthode qui affiche le nombre d'objets.

En fait, une méthode statique est une méthode qui n'utilise que ces paramètres ; Elle n'utilise ni le mot clé this qui désigne l'objet courant, ni les attributs. Par contre, elle peut utiliser les attributs statiques.

Pour déclarer un méthode de classe, on utilise le terme static.

Comme une méthode de classe n'est pas liée à un objet, alors elle ne peut utiliser directement les attributs

Comme une méthode de classe n'est pas liée à un objet, alors elle ne peut utiliser le mot clé this qui désigne l'objet courant.

Une méthode de classe peut utiliser les variables globales.

Une méthode de classe est appelée via le nom de la classe.

```
class Personne
  {
      // Afin d'eviter toute modification des attributs par les objets, il est
suhaitable de les definir via le mot clé private
      private String nom;
      private String prenom;
      private int age;

          private static int nb_personnes=0; // on cree une variable globale que
l'on initialise a la valeur 0
```

Maintenant, afin de pouvoir compter toute nouvelle personne (objet) ajoutée au projet, on va modifier les constructeurs comme suit.

```
public Personne()
      {
          nb_personnes = nb_personnes + 1; //Afin de pouvoir compter toute
nouvelle personne (objet) ajoutée
          prenom = "Inconnu";
          //nom = "Inconnu";

      }
```

```
        public Personne(String P, String N, int age)
        {
            nb_personnes = nb_personnes + 1;  //Afin de pouvoir compter toute
nouvelle personne (objet) ajoutée
            prenom = P;
            nom = N;
            this.age = age;

           }
```

Maintenant saisissez le code suivant.

```
public static int getNbP()
        {
            return nb_personnes;
        }
        // la méthode getNbP doit etre statique car elle ne fait intervenir qu'une
variable statique
        // pas de set avec une variable globale
```

Maintenant ajoutez à la classe Personne la méthode suivante.

```
public static void comptepersonnes()
        {
            System.out.println("le nombre de personnes dans mon projet est de " +
Personne.getNbP());


           }
```

Maintenant modifiez la procédure Main comme suit.

```java
public class Test
{

	public static void main(String[] args)
	{
		// On cree l'objet P0 qui est une instance de la classe Personne

		Personne P0 = new Personne();
		System.out.println("Le prenom de l'objet P0 est " + P0.getPrenom());
		System.out.println("P0=(" + P0.getPrenom() + "; " + P0.getNom() + " ;"
				+ P0.getAge() + ")");

		Personne.comptepersonnes();

		Personne P1 = new Personne("Trevor", "MonNom", 19);
		System.out.println("P1=(" + P1.getPrenom() + " " + P1.getNom() + " "
				+ P1.getAge() + ")");

		// on change l'age de l'objet P1 en lui affectant la valeur 21

		P1.setAge(21);
		System.out.println("L'age de P1 est devenu " + P1.getAge());

		Personne.comptepersonnes();

		// comme les methodes sont d'acces public, les objets peuvent
		// directement les utiliser et on peut ecrire objet.methode
		// Par contre, comme nos attributs sont private, alors on ne peut ecrire
		// objet.attribut. Dans ce cas, on utilise les
		// accesseurs et les mutateurs comme ci-dessus

		P1.manger();
		P1.dormir();
		System.out.println(P1.parler(P1.getPrenom()));

		Personne P2 = P1;
		if (P2.equals(P1))
			System.out.println("Mêmes Objets");

		Personne.comptepersonnes();

	}

}
```