**2. Java - Programmation Orienté Objet – Algorithmique - Syntaxe**

**Structure d'un programme Java**

Un programme Java est décomposé en fichiers et doit répondre à un certain nombre de règles.

**le projet**

Un programme Java, c'est un projet.  
Donc le point de départ c'est la déclaration d'un projet.  
Un projet est défini par un certain nombre de critères :

* un répertoire de travail
* un nom
* des packages
* des classes
* des fichiers de paramétrage
* un builder
* une JVM Un projet permet entre autre de créer une unité entre tous ces fichiers.

**Le package**

Les sources sont ordonnées en package. Un package est un dossier qui contient un ensemble de classes.  
Un package doit respecter une norme d'écriture :

* les caractères sont alphanumériques (a-z;0-9;.).
* les caractères sont toujours écrits en minuscule.
* les mots sont séparés par un point. Ceci défini un sous package (un dossier).
* un package contient un nom de domaine écrit en inversé (ex:fr.promoP20) Exemple: fr.promoP20.monpremierpackage

**La classe**

Une classe est définie dans un fichier qui porte le même nom. Le nom commence toujours par une majuscule. Il n'y a qu'une seule classe par fichier (sauf exception).  
Exemple:  
Classe : Animal  
Fichier : Animal.java

class <nom\_de\_la\_classe> { 🡨 Nom de classe

<contenu de la classe> 🡨 Contenu: Attribut/Methodes

}

**Programmation orienté objet avec java**

**L'approche objet**

En Java tout est objet (ou presque).  
Les objets appartiennent forcément à une classe, qui divisent cet univers. Tout code doit être écrit dans une classe.

**La classe**

Une classe déclare des propriétés communes à un ensemble d'objets d'une part, et les méthodes d'autre part.  
C'est en quelque sorte un squelette.  
Lorsque l'on instancie un objet, on le fait par rapport à une classe.  
A ce moment, la JVM alloue une place en mémoire contenant les propriétés de l'objet.  
Et uniquement les propriétés, car les méthodes sont communes à tous les objets et ne sont pas modifiables une fois compilées.  
Une classe contient aussi des variables et des méthodes propres à elle-même. Ces dernières ne peuvent pas être utilisées par un objet. Elles ont statiques.  
Une classe est identifiée par le mot réservé class, son nom, et son body contenu dans des accolades {}.  
Voici un exemple de déclaration de la classe Animal par le mot clef class.

class Animal {

}

**les propriétés**

On déclare les propriétés en début de classe. Il s'agit de variables propre à un objet.  
On fait référence à une propriété en mettant this. devant l'identifiant de la propriété.

class Animal {

String nom;

int nombreDePatte;

}

**les méthodes**

Les méthodes correspondent aux comportements. C'est l'équivalent des fonctions. On les repère par les parenthèses.  
Une méthode renvoie un résultat typé. void signifie qu'il n'y a pas de résultat.

class Animal {

String nom;

int nombreDePatte;

void parler() {

System.out.println("Je suis un animal");

}

}

**Les classes et objets**

Il ne faut pas confondre classe et objet.  
Une classe est une description et un conteneur.  
Un objet est une entité. En POO, on appelle cela une instance.  
Un objet appartient à une classe.  
On peut comparer cela à un ensemble contenant des éléments.  
Prenons l'exemple d'un sac de pommes :

* le sac ne contient que des pommes (pas de poire)
* une pomme est définie par des propriétés
  + sa forme
  + son poids
  + sa couleur
* une pomme a un comportement ou des aptitudes
  + elle pourrit
  + elle est comestible
  + on peut la manipuler
* une pomme est spécifique
  + chaque pomme a ses propres propriétés
  + toutes les pommes ont le même comportement
* on peut considérer la classe comme étant le sac
  + c'est un conteneur
  + les pommes sont le contenu
  + on pourrait mettre sur le sac une étiquette indiquant le nombre de pommes (variable de classe)

**Constructeur et destructeur**

Tout objet a une vie, et donc naît et meurt.  
L'acte de créer une instance d'une classe est appelé instanciation de cet objet.  
On instancie un objet dans une classe.  
Pour créer un objet on utilise le mot **new**. Ceci appelle une méthode : le constructeur.  
Cette méthode réserve une place en mémoire et renvoie l'adresse de cet emplacement.  
Cette adresse est stockée dans une variable de typée.  
Lorsqu'un objet est détruit, la méthode appelée est le destructeur.

**Constructeur**

Le constructeur est une méthode publique qui porte le nom de la classe.  
En fait on doit parler de constructeurs, car on peut créer plusieurs de constructeurs avec des paramètres différents.  
Dans cette méthode, on renseigne les propriétés par des valeurs passées en paramètre.  
On peut aussi appeler le constructeur du parent.  
Par contre n'appelez pas de méthode dans le constructeur, l'exécution doit être le plus rapide possible.  
Un constructeur n'est pas obligatoire.

public Oiseau(int nombreDePatte, boolean teteSepare, int nombreDAile, String couleurDuBec ) {

super(nombreDePatte, teteSepare); //appel du constructeur parent

this.nombreDAile = nombreDAile;

this.couleurDuBec = couleurDuBec;

}

Oiseau oiseau = new Oiseau(2, true, 2, "noir");

**Destructeur**

En Java, on ne supprime pas d'objet.  
C'est le ramasse-miette (garbage collector) qui s'exécute en tache de fond et détruit tous les objets qui ne sont plus référencés.  
Un objet n'est plus référencé quand on lui assigne la valeur null, ou quand son scope (périmètre de vie) est fini (notion vue plus tard).  
Il est très rare de forcer la destruction d'un objet par son assignation à null.  
Nous n'avons donc aucune idée du moment où sera détruit l'objet.  
Lorsque que l'objet est détruit, juste avant sa destruction le moteur Java appelle la méthode finalize.  
Il est très rare d'implémenter du code dans le destructeur. Cependant, certaines situations nécessitent un nettoyage spécial que le ramasse-miettes ne peut pas effectuer lorsque la classe disparaît.  
Par exemple, certains fichiers ont été ouverts pendant la durée de vie de l'objet et vous voulez vérifier qu'ils sont correctement fermés quand l'objet est détruit.

**Static**

Le mot clef static est très important.  
Jusqu'à présent nous avons vu les méthodes et propriétés liées à un objet.  
Cependant, il peut arriver d'avoir un besoin lié à une classe. Dans ce cas on utilise le mot static dans la déclaration.  
C'est ce mot qui va permettre de faire la distinction entre les variables et méthodes de la classe ou de l'instance.  
Un élément déclaré static appartient à une classe et non à ses instances.

Une méthode (ou une propriété) static est commune à tous les objets. Elle n'est pas instanciée.  
Une méthode static ne peut pas appeler une méthode (ou une propriété) d'instance sans définir un objet.  
Ceci se rapproche du paradigme de programmation procédurale : notion de variable et de fonction.

**exemple complet**

public class Animal {

//propriété de la classe

private static int totalAnimal = 0;

//propriétés de l'objet

protected String nom;

private int nombreDePatte;

//constructeur

public Animal(String nom, int nombreDePatte) {

super(); //appel du constructeur parent

this.nom = nom;

this.nombreDePatte = nombreDePatte;

totalAnimal++;

}

//une methode de l'objet

private String formulerMonNom() {

return "Je m'appelle " + this.nom; //appel de la propriété nom

}

//une methode de l'objet

public void parler() {

System.out.println(this.formulerMonNom()); //appel d'une méthode de l'objet

System.out.println("Je suis un animal et j'ai "+ this.nombreDePatte + " pattes");

}

//une methode de la classe

public static void afficherNombreDAnimaux() {

System.out.println("Il y a " + totalAnimal + " animaux");

}

}

public class AppelAnimal {

public static void main(String[] args) {

Animal animal = new Animal("titi",2); //creation animal

animal.parler(); //appel d'une méthode d'instance

Animal.afficherNombreDAnimaux(); //appel d'une méthode de classe

}

}

* **Note Important -**
* tous les programmes Java sont composés d'au moins une classe qui doit contenir une méthode main (méthode principale du programme)
* Qu’est-ce qu’une méthode ? une suite d'instructions à exécuter.
* Qu’est-ce qu’elle contient ?
  + **une entête** : la carte d'identité de la méthode
  + **un corps** : le contenu de la méthode (Délimité par des accolades {})
  + **une valeur de retour** : le résultat que la méthode va retourner

\*\*pour les méthodes de type void : renvoient rien

**Plus Loin :**

**Livres :**

* Programmer en Java, 7e Edition, Claude Delannoy, Eyrolles, 2011
* The Java Tutorial : A Short Course on the Basics, 4th Edition, Collectif, Prentice Hall, 2006 Effective Java, 2nd Edition, Joshua Bloch, Prentice Hall, 2008
* Java in a nutshell, 5th edition, David Flanagan, O’Reilly, 2005

**Sites web** :

* Le site officiel Java, http://www.oracle.com/technetwork/java/index.html
* Le tutorial Java, http://docs.oracle.com/javase/tutorial/
* l’API du JDK 1.7, <http://docs.oracle.com/javase/7/docs/api/>
* Tutoriaux jGuru, http://java.sun.com/developer/onlineTraining/