Programmation Orientée Objet POO

Université Saad Dahlab
de Blida

Mme BOUTOUMI

L2

2014 - 2015

A propos du cours

- Ce cours est conçu comme une introduction à la *programmation* orientée objet.
- Pré-requis: Le cours d'algorithmique et structures de données
- **Java** sera utilise comme exemple de langage oriente objet pour illustrer les notions vues en cours.
- **Eclipse** sera utilise comme environnement de développement.

A propos du cours

Objectifs:

- Introduire l'approche orientée objet,

A l'issue du cours,

- Vous aurez pris conscience de l'importance d'appliquer les principes de l'oriente objet (Abstraction, encapsulation, héritage, polymorphisme..etc.).
- Vous serez capables de concevoir une petite application en utilisant l'approche orientée objet.
- Vous serez capable de l'implémenter en java.

MODALITES DE CONTRÔLE DES CONNAISSANCES

- Un examen final
- Un TP test noté sur 10 points. 10 points pour le contrôle continue.
- Deux Interrogations pendant le semestre (I1 et I2)

I1 noté sur 1O points et I2 noté sur 5 points + 5 points assiduité et participation.

- 1 point sera retenue de la note de TD pour chaque absence pendant la séance de TD ou TP

Contenu du cours

- Chapitre 1:Introduction à la Programmation Orienté Objet.
- Chapitre 2:Les classes
- Chapitre 3:L’héritage et polymorphisme
- Chapitre 4:Interface et implémentation
- Chapitre 5:Interface graphique et Applet

Bibliographie

Il existe de nombreux livres sur java…

- Claude Delannoy, « Programmer en Java », Eyrolles

http://java.sun.com/docs/books/jls/

- *Thinking in Java by Bruce Eckel*

http://www.ibiblio.org/pub/docs/books/eckel/

- Lemay L, « Le Programmeur JAVA 2 », Campus Press.
- Horstmann et Cornell, « Au coeur de Java 2 Volume I - Notions fondamentales »,
- ***The Java Programming language fourth edition AW Ken Arnold, James Gosling, David Holmes***

Programmation Orientée Objet POO

Université Saad Dahlab
de Blida

CHAPITRE 1
Introduction à la programmation orientée objet

Disponible sur: https://sites.google.com/a/esi.dz/s-aroussi/

Mme BOUTOUMI (b_boutoumi@esi.dz)

L2

2014 - 2015

1. Mise en œuvre d'une application

Plusieurs étapes sont nécessaires pour mettre en œuvre une application :

Etape 1 :Définition du problème
Etape 2 :Analyse du problème
Etape 3 :Concevoir une solution au problème
Etape 4 :Implémenter la solution (Programmer)
Etape 5 :Mise au point du programme (Tester la solution)

Ceci doit être fait en considérant

- La maintenance du logiciel
- l'évolution du logiciel
- la réutilisation du logiciel

2. Paradigme de programmation

Un paradigme de programmation:

- Une façon d’aborder un problème.
- Une manière de penser.
- Concrétisation d’une philosophie de programmation.
- Un style de programmation.

3. Paradigmes des langages de programmation

Il existe 4 principaux paradigmes de programmation:

- La programmation **impérative** (procédurale): Pascal, C, Fortran. (programme = algorithme + structure de données).
- La programmation **fonctionnelle** : (adopte une approche beaucoup plus mathématique de la programmation) Lisp, Scheme,Haskell.
- La programmation **logique** : (la description d'un programme est sous forme de prédicats) Prolog, GHC.
- La programmation **Orientée Objet** : Smalltalk,C++, Java, C#

4. Programmation impérative VS Programmation Orienté objet (1)

- Comment avez-vous l'habitude de programmer?
- Quelle est la structure de votre code?

➢ Représentation procédurale:

- Séparer les données des traitements.
- Subdivision basée sur les traitements.

➢ Représentation Objet:

- Manipuler un ensemble d'objets qui interagissent entre eux.
- Subdivision basée sur les données.

4. Programmation impérative VS Programmation Orienté objet (2)

Dans la programmation procédurale

- On apprend à subdiviser un problème en une série d'étapes simples (des fonctions) permettant de résoudre un problème.
- D'abord, on décide de la manière dont on va manipuler les données, ensuite le type de structures de données les plus appropriées pour faciliter cette manipulation.
- Donne des logiciels difficiles à maintenir et à réutiliser

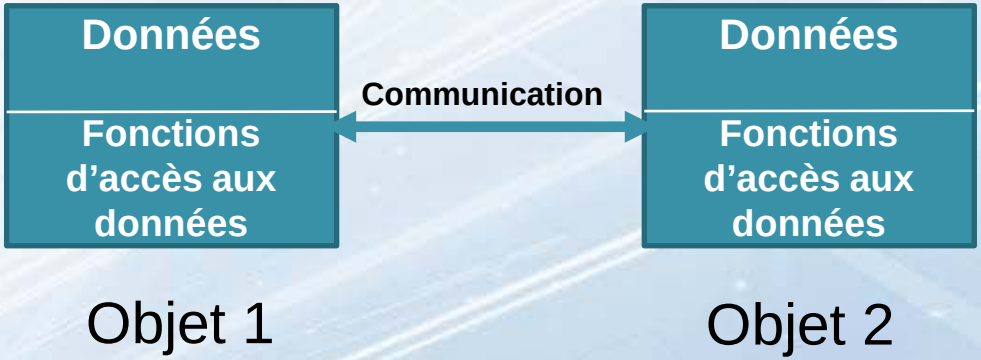
Dans La programmation orientée objet

- Un programme informatique est considéré comme étant un ensemble d'objets fonctionnant ensemble pour effectuer une tâche.
- On s'intéresse d'abord aux données, avant de déterminer l'algorithme qui sera utilisé pour opérer sur ces données.
- Les objets coopèrent par envois de messages (Appel de méthodes)

4. Programmation impérative VS Programmation Orienté objet (3)

Programmation procédurale

Programmation Orientée Objet



5. La programmation orienté objet

Définition:

La POO est une méthode d'implémentation dans laquelle les programmes sont organisés sous formes de collections coopératives d'objets, dont chacun représente une instance d'une classe quelconque et dont toutes les classes sont membres d'une hiérarchie de classes unis à travers des relations d'héritage.

6. Algorithmique Objet

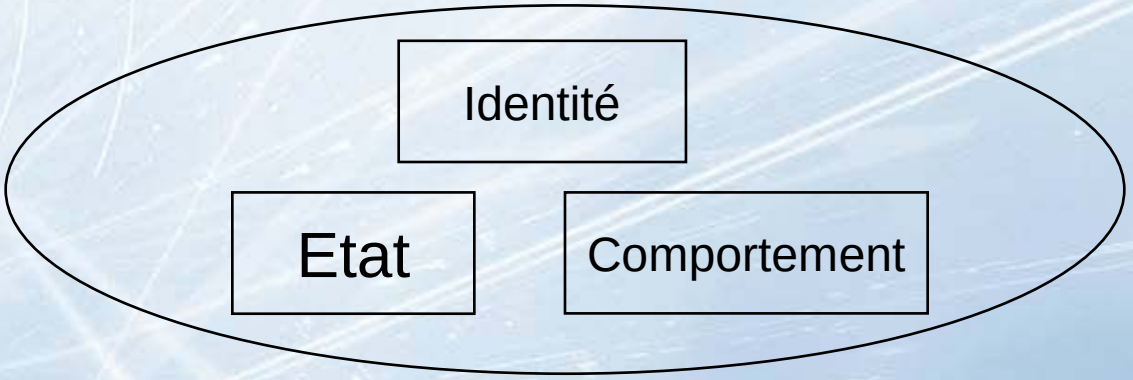
Un algorithme dans l'approche orienté objet sera essentiellement vu comme un ensemble d'objets auxquels l'utilisateur envoie des messages et qui s'en envoient pendent le fonctionnement. Ces objets seront toujours pour l'utilisateur des boites noires et qui contiendront des variables locales, inconnues de l'environnement, et qui ne s'y intéressera d'ailleurs pas. Le seul moyen d'accéder à ces objets sera l'envoie des messages qu'ils sont capables de comprendre.

7. C’est quoi un Objet ? (1)

L’objet est une entité logicielle qui représente la brique de base de la POO:

L’objet se caractérise par :

- Une identité (référence ou handle)
- Un état.
- Un comportement.



Objet

7. C’est quoi un Objet? (2)

Dans le monde réel, un objet peut être: un étudiant, un enseignant, un téléphone, une voiture, etc.

Exemple: Une voiture

- L’état = Nom, Marque, type, nombre de vitesses, Année, couleur,…
- Le comportement = rouler, tourner, accélérer, changer de vitesse, freiner,…

7. C’est quoi un objet? (3)

Dans la POO un objet est un mélange de variables et de fonctions ou procédures.

L’objet :

- L’objet possède une **identité** unique qui le distingue des autres objets indépendamment de son **état** et de son **comportemen**t.
- maintient son **état dans des variables appelées attributs ou champs.**
- implémente son **comportement** à l'aide de fonctions ou procédures appelées **méthodes.**

Remarque: Deux objets peuvent avoir le même état et le même comportement mais ont toujours des identités différentes, et réciproquement, le comportement et l’état d’un objet peuvent changer durant l’exécution sans que cela affecte son identité.

7. C’est quoi un objet? (4)

Exemple: Un rectangle est défini par:

- ses attributs : longueur, largeur, couleur.
- son comportement déclenché par des méthodes:
  - Dessiner.
  - Calculer son périmètre.
  - Calculer sa surface.

8. Cycle de vie d’un objet

Un objet en programmation orienté objet (POO) a un cycle de vie de 3 phases :

1. Construction (en mémoire).

2. Utilisation (changements d’état par affectations, comportements par exécution de méthodes)

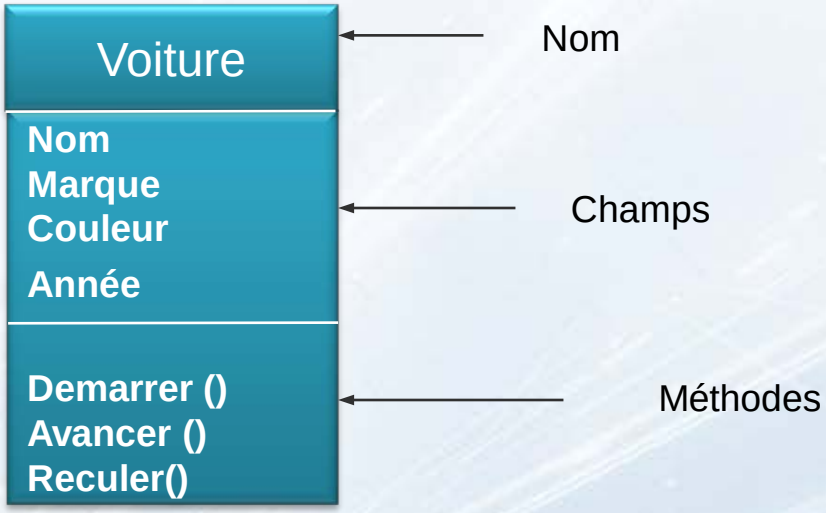
3. Destruction

9. La classe

La classe est une description d'une famille d'objets ayant une même structure et un même comportement. Elle est caractérisée par :

- Un nom
- Une composante statique : des **champs** (ou **attributs**) nommés ayant une valeur. Ils caractérisent l'état des objets pendant l'exécution du programme
- Une composante dynamique : des **méthodes** représentant le comportement des objets de cette classe. Elles manipulent les champs des objets et caractérisent les actions pouvant être effectuées par les objets.

10. Représentation graphique d'une classe



11. Classe et objet (1)

- La classe est la **machin**e qui fabrique les objets. Elle peut être considérée comme un **moule** ou un **modèle** à partir duquel on peut créer des objets.
- Une classe est un **modèle** de définition pour des objets partageant des **caractéristiques communes**, mais l'**état** de chaque objet est indépendant des autres
- Des objets créés à partir de la même classe auront des aspects semblables (mêmes attributs et même méthodes).
- Un **objet d'une classe** est aussi appelé ***instance*** de cette classe.

11. Classe et objet (2)

- Exemple:

La classe

Les objets (instances)

11. Classe et objet (3)

- Les objets V1,V2 ont les mêmes valeurs d'attributs, on dit qu'ils sont égaux car ils ont le même *état*. Ce sont néanmoins deux objets distincts car ils ont des *identités* différentes .
- Les objets d'une même classe ont toujours une identité distincte et généralement un état distinct.

12. Concepts de base de la POO

- La Programmation orienté Objet (POO) est basée sur les concepts suivants:
  - L’encapsulation
  - L’héritage
  - Le polymorphisme

13. L’encapsulation (1)

- L’encapsulation est un principe clé dans la POO.
- Elle consiste à regrouper des données (attributs) et un comportement (méthodes) dans une même classe et à réglementer l’accès aux données de l’extérieur (par d’autres objets).
- Cela signifie qu’il n’est pas possible d’agir directement sur les données d’un objet ; il est nécessaire de passer par ses méthodes.
- L’appel de la méthode d’un objet est aussi appelé *envoi de message* à l’objet.

13. L’encapsulation (2)

- Le principe de l’encapsulation est que, vu de l’extérieur, un objet se caractérise uniquement par ses méthodes visibles appelées interface. Les données, elles, restent invisibles et inaccessibles.

- Ce principe permet de concevoir des programmes (logiciels) plus sûrs, plus lisibles et plus faciles à maintenir, car une modification de la structure des données d’un objet ne se répercute pas sur les objets qui communiquent avec lui (invoquent ses méthodes).

14. L’héritage (1)

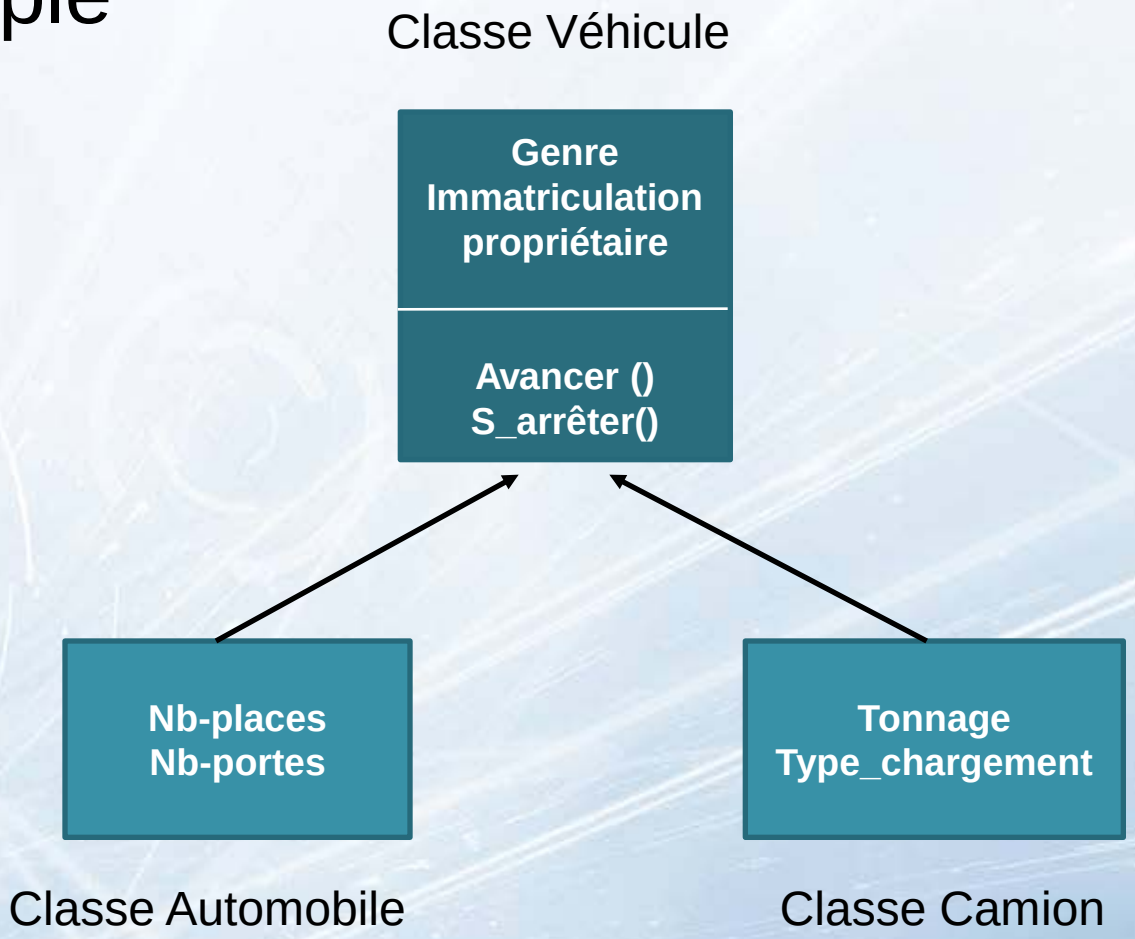
- L’héritage est un autre concept important en POO.

- Il permet de définir une (ou plusieurs) nouvelle classe (appelée *classe dérivée, classe fille* ou *sous-classe*) à partir d’une classe existante (appelée *classe de base, classe mère* ou *super-classe*), à laquelle on ajoute de nouvelles données et de nouvelles méthodes.

- L’héritage facilite largement la réutilisation de produits existants, ce qui est un avantage important de la POO.

14. L'héritage (1)

- Exemple



15. Le Polymorphisme (1)

- Le terme polymorphisme issu du grec signifie la faculté de prendre plusieurs formes.
- C’est un concept puissant de la POO qui vient compléter celui de l’héritage.
- En POO, cela peut s’appliquer aussi bien aux objets qu’aux méthodes.

15. Le Polymorphisme (2)

Polymorphisme d’objets

- Il permet une certaine flexibilité dans la manipulation des objets en exploitant la relation d’héritage entre les classes.
- Il applique la règle suivante :

  Si la classe A est dérivée de la classe B

  alors un objet de la classe A peut aussi être considéré comme étant un objet de la classe B.

15. Le Polymorphisme (3)

Polymorphisme de méthodes

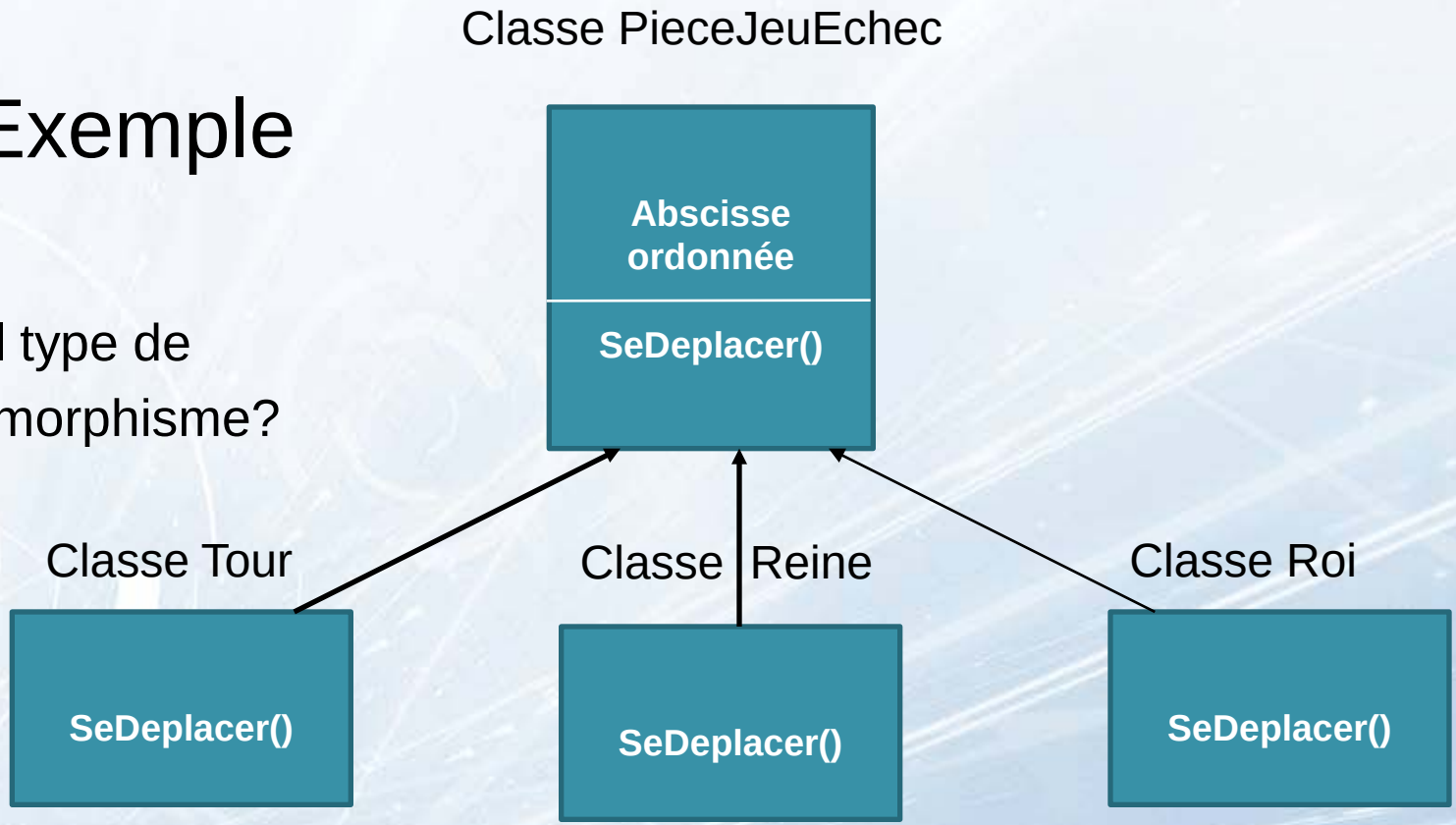
Il exprime le fait que :

- Des méthodes d'une même classe puissent avoir le même nom et des signatures différentes (ceci est appelé ***surcharge*** ou ***surdéfinition*** de méthodes)
- La même méthode peut se comporter différemment sur différentes classes de la hiérarchie. Autrement dit, des méthodes peuvent avoir des noms similaires dans une hiérarchie de classes et être différentes. Ceci est appelé ***redéfinition*** ou ***spécialisation*** de méthodes.

15. Le Polymorphisme (4)

- Exemple

Quel type de polymorphisme?



16. Les TAD (1)

- **Le type** d'une variable est l'ensemble des valeurs qu'elle peut prendre.

- Un **type de données**, ou simplement type, définit le genre de contenu d'une donnée et les opérations pouvant être effectuées sur la variable correspondante.

Exemples: les types élémentaires de donnée : entier, réel, chaîne, caractère, booléen et pointeur.

- Une **structure de données** est une manière particulière de stocker et d'organiser des données dans un ordinateur de façon à pouvoir être utilisées efficacement.

Exemples: Structure (Enregistrement), Tableau, Liste,etc.

16. Les TAD (2)

- Un **Type Abstrait de Données (TAD)** est une spécification mathématique d'un ensemble de données et de l'ensemble des opérations qu'elles peuvent effectuer. On qualifie d'abstrait ce type de données car il correspond à un cahier des charges qu'une **structure de données** doit ensuite implémenter.
- Un TAD est un ensemble de valeurs muni d'opérations sur ces valeurs, sans faire référence à une implémentation particulière.

TAD

16. Les TAD (3)

- Les TAD sont des entités purement théoriques utilisés principalement pour simplifier la description des algorithmes.
- Pour mettre en œuvre les TAD, il faut recourir à des **structures de données**, qui sont des ensembles de variables, à priori de types différents et reliées de différentes façons.
- **Une structure de données:** correspond à l'implémentation physique d'un TDA.

Exemple:

- Le TAD Pile sera défini par les opérations : CréePile, PileVide, Pilepleine, Empiler et Dépiler.
- Le TAD File sera défini par les opérations : CréeFile, FileVide, Filepleine, Enfiler et Défiler.

16. Les TAD (4)

- Le TAD permet de définir des types de données non primitifs, c'est-à-dire non disponibles (non déjà implémentés) dans les langages de programmation courants.

- En réalité, on a utilisée toujours les TAD, sans le savoir. En effet, dans un algorithme on doit manipuler des nombres entiers ou réels, on ne se préoccupe pas de la représentation de ces nombres, mais uniquement des propriétés essentielles des quatre opérations habituelles (+,-,*,/).

- Un TAD ne spécifie pas la façon dont les données sont stockées ni comment les opérations sont implémentées.

17. Caractéristiques des TAD

Un TAD est caractérisé par:

- Sa signature: définit la syntaxe du type et les opérations.
  - Le nom du TAD,
  - Les noms des types des objets utilisés par le TAD,
  - Pour Chaque opération, l'énoncé des types des objets qu'elle reçoit et qu'elle renvoie.
- Sa sémantique: définit les propriétés des opérations.
  - Les domaines de définition (ou d'application) des opérations,
  - Les propriétés des opérations.

18. Implémentation des TAD

Un TAD correspond à une classe contenant deux parties:

- Une partie visible (publique) correspondant aux opérations (méthodes) que le TAD peut effectuée.
- Une partie cachée (privée) qui contient les attributs de la classe décrivant la structure de donnée (SDD) choisie.

18. Implémentation des TAD (2)

- Les opérations sur un TAD sont de quatre types :
  - Constructeurs
  - Modificateurs
  - Sélecteurs
  - Itérateurs

18. Implémentation des TAD (3)

- Le TAD Pile :
  - Constructeurs: Créer_pile
  - Modificateurs: Empiler, Dépiler, vider_pile, etc.
  - Sélecteurs: Pile_vide, Pile_pleine, longueur _pile, etc.

5. Références bibliographiques

- Hugues Bersini, La programmation orientée objet, ÉDITIONS EYROLLES, 2009
- N Bousbia et S SadeG, Programmation Orienté Objet, Ecole nationale Supérieur d'Informatique ESI, 2013/2014.