UEF4.3. Programmation Orientée Objet

Mme SADEG

s_sadeg@esi.dz

Ecole nationale Supérieure d'Informatique (ESI)

2010/2011

A propos du cours

- C'est un cours *introductif* à la programmation orientée objet.
- Pré-requis: Le cours d'algorithmique
- **Java** sera utilisé comme exemple de langage orienté objet pour illustrer les notions vues en cours
- **BlueJ** sera utilisé comme environnement de développement

A propos du cours

- Objectifs: Apprendre les principes de la programmation orientée objet
- A l'issue du cours, vous serez capables de
  - Assimiler les concepts de base de la P.O.O
  - Prendre conscience de l'importance d'appliquer les principes de l'orienté objet (Abstraction, héritage, polymorphisme..etc)
  - Implémenter une petite application en java.

A propos du cours

- Volume horaire
  - 28 heures de cours (1 séance de 2h/semaine)
  - 28 heures de TD/TP (1 séance de 2h/semaine)
- Crédits: 4
- Coefficient: 4
- Evaluation:
  - Note de participation
  - Note de TPs
  - Contrôle intermédiaire
  - Examen semestriel

Contenu du cours

- Introduction à la P.O.O
- Objets et classes
- Héritage et polymorphisme
- Traitement des exceptions
- Interface graphique
- Applets
- Flux et fichiers
- Application multithread

Bibliographie

- *Objects First with Java: A Practical Introduction using BlueJ. David J. Barnes & Michael Kölling, Pearson* Education
- Deitel et Deitel, « Programmer en JAVA », Les éditions Reynald Goulet
- Lemay L, « Le Programmeur JAVA 2 », Campus Press.
- Horstmann et Cornell, « Au coeur de Java 2 Volume I - Notions fondamentales »,
- Claude Delannoy, « Programmer en Java », Eyrolles

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

- *Thinking in Java by Bruce Eckel available electronically*

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

Chapitre I

Introduction à la Programmation Orientée Objet

Introduction (1)

Résoudre un problème

- Comprendre et analyser le problème
- Concevoir une solution
- Implémenter la solution (programmer)
- Tester la solution

Ceci doit être fait en considérant

- Maintenance
- Évolution
- Réutilisation

Du logiciel

Introduction (2)

- Un paradigme de programmation est un style de programmation.

- Il existe 4 principaux paradigmes de programmation:
  - Procédural: Fortran ,C, Pascal, …
  - Fonctionnel: LISP, Scheme,Haskell…
  - Logique:Prolog,GHC…
  - Orienté Objet: Smalltalk,C++, Java, C#

Procédural Vs Orienté objet

Dans la programmation procédurale

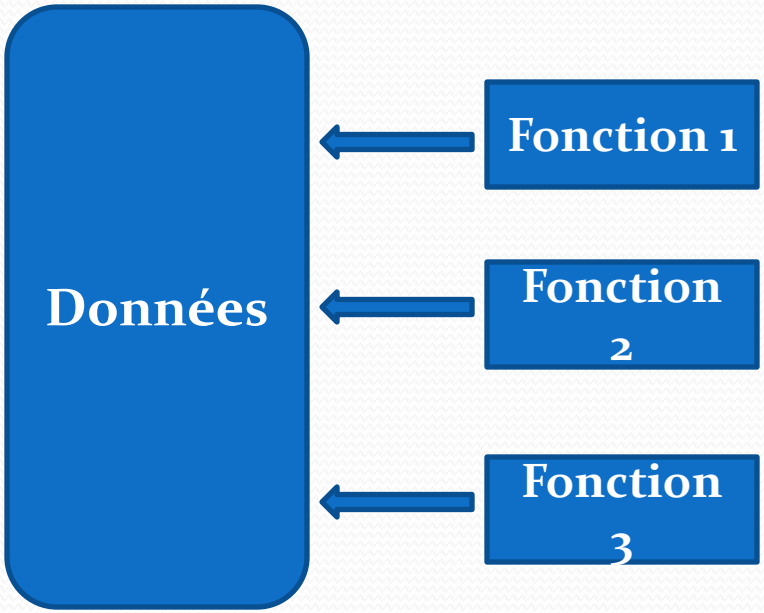
- on apprend à subdiviser un problème en une série d'étapes simples (des fonctions) permettant de résoudre un problème.
- D'abord, on décide de la manière dont on va manipuler les données, ensuite le type de structure de données les plus appropriés pour faciliter cette manipulation.
- Donne des logiciels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

Dans La programmation orientée objet

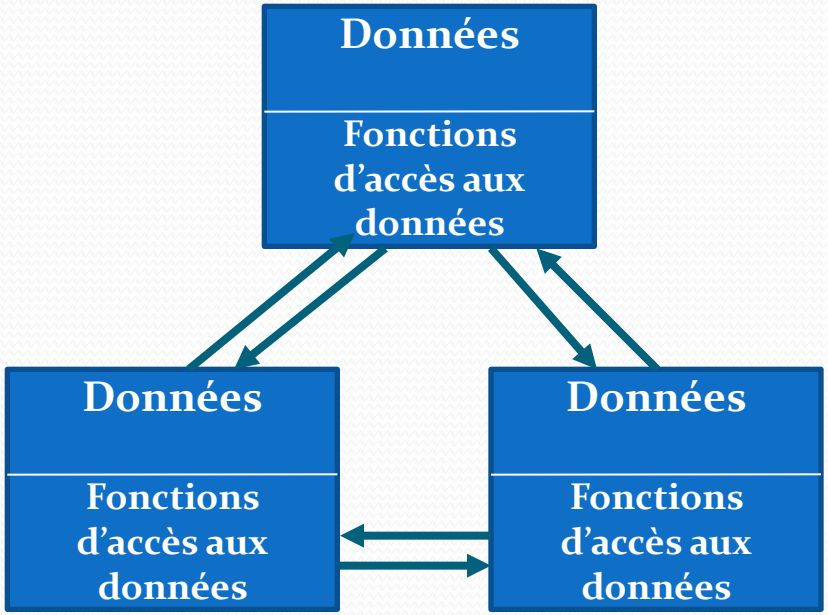
- un programme informatique est considéré comme étant un ensemble d'objets fonctionnant ensemble pour effectuer une tâche.
- On s'intéresse d'abord aux données, avant de déterminer l'algorithme qui sera utilisé pour opérer sur ces données.
- Les objets coopèrent par envois de messages (Appel de méthodes)

Procédural Vs Orienté Objet

Programmation procédurale



Programmation Orientée Objet



Qu’est ce qu’un objet ?

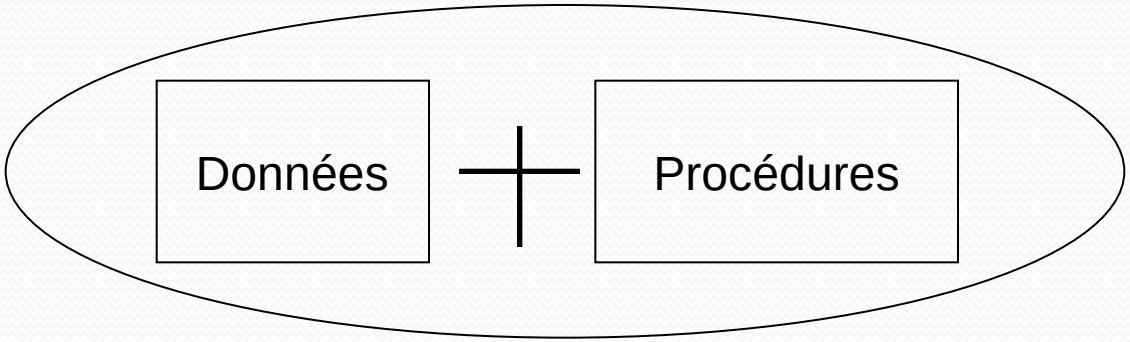
- La Programmation orientée objet est basée sur la notion d’objet

- Dans le monde réel, un objet peut être:
  - Une voiture,
  - Un pays,
  - Un enseignant
  - Un ordinateur
  - …

Qu’est ce qu’un objet ?

- Dans le monde réel, les objets ont des attributs et des comportements:

Objet	attributs	comportements
• Une voiture,	Accélérateur	Accélère
• Une bibliothèque,	abonnés	prête des livres
• Un enseignant	Cours	Enseigne
• Un ordinateur	Processeur	Calcule



Objet

Objets et classes

- Ce sont des notions essentielles en P.O.O

- **Objet:** Un *objet* est un élément dans un programme rassemblant des données et du code manipulant ces données. Il est défini par trois caractéristiques : Une identité, un état et un comportement.

Objet = Identité + état + comportement

Objet (1)

Identité

L'objet possède une identité unique qui le distingue des autres objets indépendamment de son état et de son comportement. 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.

Objet (2)

Etat

ce sont les données qui caractérisent l'objet, i.e., des informations sur l'objet stockées dans des variables appelées aussi *attributs* ou *champs*.

Objet (2)

Comportement

(méthodes ou fonctions)

C'est l'ensemble de traitements que peut accomplir un objet.

Classe

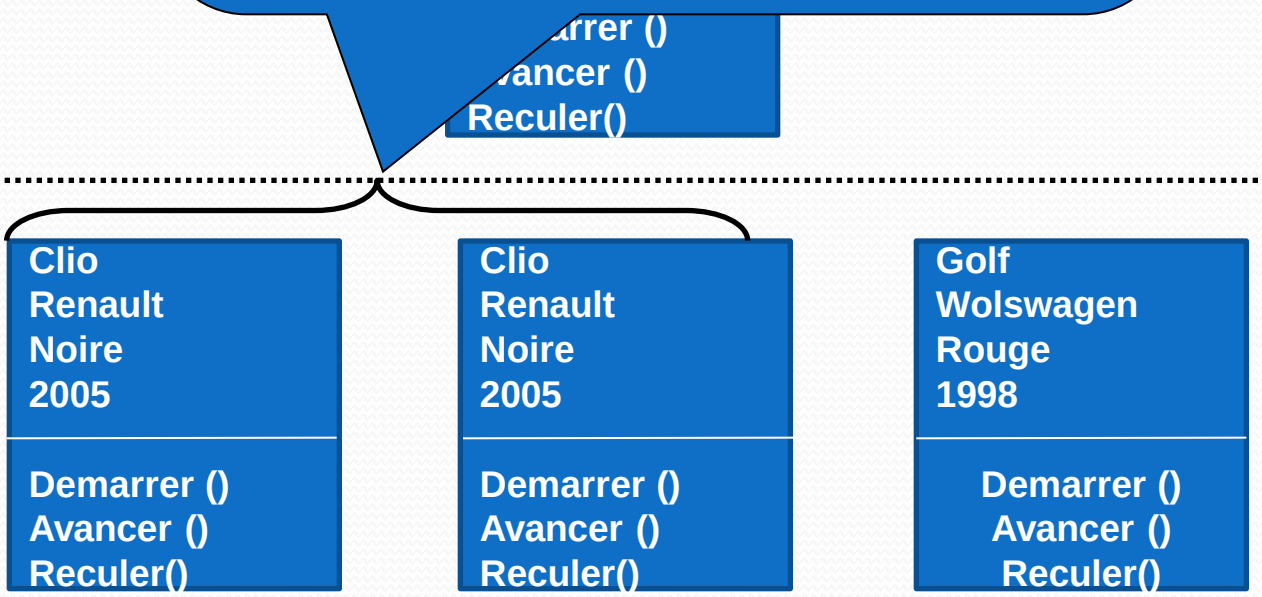
- Une *classe* peut être considérée comme un moule ou un modèle à partir duquel on peut créer des objets.
- Des objets créés à partir de la même classe auront des aspects semblables (mêmes attributs et même méthodes).
- Un objet d'une classe est aussi appelé *instance* de cette classe.

Obj

Ces objets 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.

- Exemple:

Classe véhicule



Objets (instances)

Concepts de base de la programmation orientée objet

- La P.O.O. est basée sur trois principaux concepts
  - L'encapsulation
  - L'héritage
  - Le polymorphisme

Encapsulation (1)

- L'encapsulation est un principe clé dans la POO.
- Elle consiste à regroupe 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.

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).

Héritage

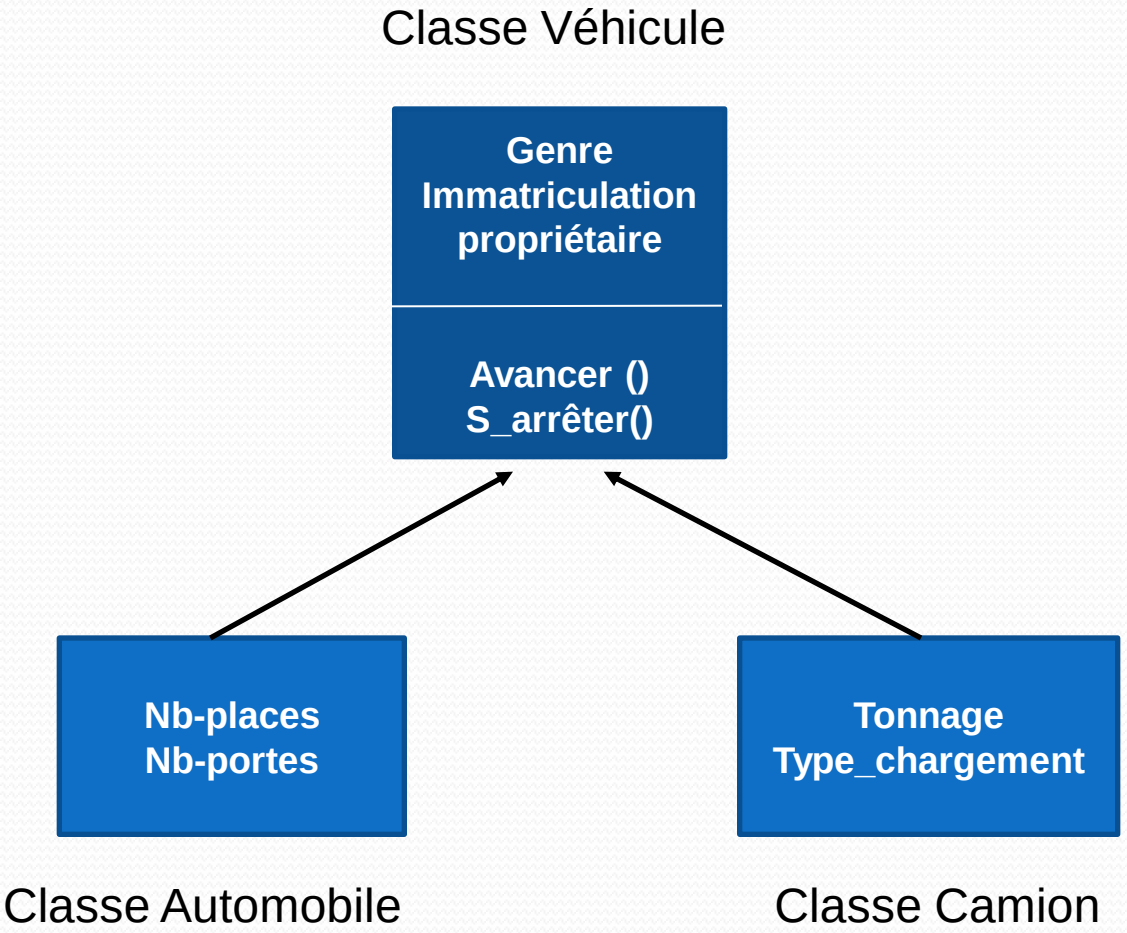
- L'héritage est un autre concept important en P.O.O.

- Il permet de définir une (ou plusieurs) nouvelle classe (appelée *classe dérivée, classer 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 P.O.O.

Héritage

- Exemple



Polymorphisme

- C’est un concept puissant de la P.O.O. qui vient compléter celui de l’héritage.
- Le terme polymorphisme décrit la caractéristique d'un élément qui peut se présenter sous différentes formes.
- En P.O.O., cela peut s’appliquer aussi bien aux objets qu’aux méthodes.

Polymorphisme

Polymorphisme d'objets

Il permet une certaine flexibilité dans la manipulation des objets en exploitant la relation d'héritage entre les classes. Pour cela, 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.

Polymorphisme

Polymorphisme de méthodes

Il exprime le fait que :

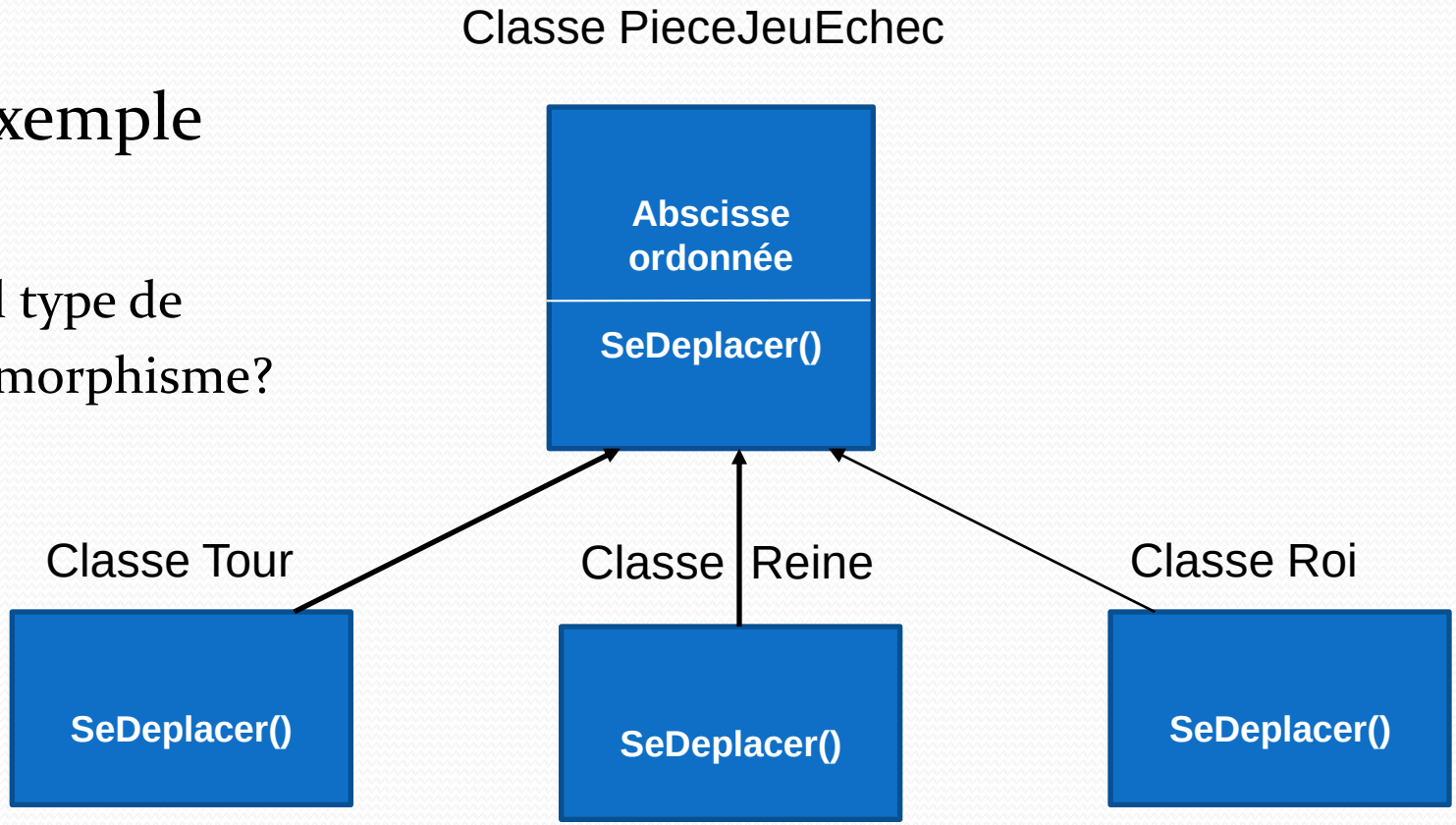
- Des méthodes d'une même classe puissent avoir le même nom et des signatures différentes (ceci est appelé ***surcharge*** ou ***surdéfinition*** de méthodes)

- La même méthode puisse 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.

Polymorphisme

- Exemple

Quel type de polymorphisme?



Conclusion

- La programmation orientée objet permet de concevoir, maintenir et réutiliser plus facilement les programmes grâce à ses principes d'encapsulation, d'abstraction, d'héritage et de polymorphisme.

- Chacun de ces concepts sera abordé plus en détails lors des prochains cours, Tds et Tps.