Lexique

Programmation Objet
Complément au cours 'Génie Logiciel', MIA, Sciences-U, 2005-2006.
Pierre Parrend.

Attribut : Variable existant dans une classe tout au long de son existance. Les attributs sont définis dans le corps de la classe, hors des méthodes.

Bloc de code : ensemble de commande, délimité par des accolades {}. Il peut s'agir de méthodes, ou de commande de contrôle de flux (if, while, etc.).

Classe : définition d'une entité représentant un élément du monde réel, ou une entité abstraite, qui intervient dans le traitement d'une situation par un système informatique. La classe, outil de définition, est réalisé sous forme de code source dans une langage de programmation donné. Elle doit être distinguée de l'Objet, qui est la réalisation en mémoire d'une classe lors de l'exécution du programme.
Une classe comprend des attributs, des méthodes, et éventuellement des sous-classes (nested classes).
En règle générale, chaque classe est définie dans un fichier propre.

Classe abstraite : classe ne pouvant pas être instanciée. L'utilisation d'une classe abstraite n'est possible que par le biais d'une classe non abstraite qui en hérite. Une classe abstraite peut contenir des signatures de méthodes, et des méthodes implémentées.

Garbage Collection : mécanisme de nettoyage de la mémoire, qui permet de libérer celle-ci des objets sur lesquels aucun pointeur n'existe – c'est à dire lorsqu'ils sont devenus innaccessibles par le programme.

Généralisation : opération inverse de l'héritage : la classe mère généralise la classe fille.

Héritage : extension d'une classe (la classe mère) par une autre (la classe fille), qui peut la compléter par de nouvelles méthodes et de nouveaux attributs, et/ou modifier les méthodes existantes.

Implémentation : réalisation du code correspondant à une description abstraite de tout ou partie d'une programme. Il peut s'agir d'implémentation d'une méthode conformément à un signature, ou d'implémentation d'une classe conformément à un modèle.

Instance : réalisation en mémoire d'un élément de conception. L'instance d'une classe est un objet, l'instance d'une association est un lien.

Instanciation : phénomène de création d'une instance à partir d'une définition abstraite : par exemple, un objet est créé lors de l'instanciation d'une classe.

Interface : définition de services de programmation, sous forme de signatures de méthode, et éventuellement de constantes.

Méthode : = opérations. bloc de code réutilisable, caractérisé par un nom de méthode, qui permet à une classe quelconque de l'appeler (sous réserve de visibilité suffisante), de paramètres, et d'un type de retour (qui peut être `void`).

Nested class : classe contenue dans une autre, qui n'a pas d'existence indépendante de la classe qui la contient. Le code d'une nested class est présent dans le même fichier que la classe qui le contient.

Objet : unité d'exécution d'un programme, présent en mémoire uniquement lors de l'exécution. Un objet est définit par la classe qu'il instancie, c'est à dire qu'il réalise.

Package : unité de groupement des classes. Un package peut contenir une ou plusieurs classes, ainsi que un ou plusieurs packages de niveau inférieur. Un package est réalisé par un répertoire dans le système de fichier.

Paramètre : variable transmise lors de l'appel d'une méthode.

Polymorphisme : possibilité de réalisation d'une méthode conformément à un service donné (c'est à dire une interface), de telle manière que l'on puisse exécuter cette méthode pour traiter des classes différentes, à condition qu'elle implémente cette interface.

Processus : ensemble de commandes exécutées sequentiellement. Plusieurs processus peuvent exister en même temps, ce qui permet d'exécuter des commandes en parrallèle.

Programmation fonctionnelle : modèle de programmation qui consiste à traiter séparément les données et les traitement, de manière comparable à ce qui se passe dans un processeur.

Programmation Objet : modèle de programmation qui consiste à représenter dans une même entité les données et leurs traitements. Ceci permet de gagner en modularité, et donc en réutilisabilité et en évolutivité.

Sémantique : signification des noms de classes, de méthodes et d'attributs, dans le contexte.

Service : au niveau de la programmation objet, un service est la définition d'un ensemble de tâches (définies par des signatures de méthodes), qu'une classe peut exécuter. Un service est donc représenté par une interface.

Signature de méthode : ensemble formé par le nom, les paramètres, le type de retour et

le modificateur de visibilité. Les signatures de méthode ne sont pas accompagnée de code d'implémentation. On les trouve dans les classes abstraites et les interfaces.

Substitution : possibilité de remplacer une classe par une autre de manière transparente.

Type : le type d'un objet définit ce qu'il est (attributs), et ce qu'il peut faire (méthodes). Deux objets de même type auront les mêmes attributs, et les mêmes méthodes. Deux objets de type différents ne peuvent pas être utilisés l'un pour l'autre, à moins qu'un des deux hérite de l'autre.
On distingue deux catégories de types : les types simples (int, long, etc.), et les types complexes, qui sont représentés par des classes.

Variable : pointeur sur un objet d'un type donné. Une variable peut être locale (dans une méthode), ou globale (dans une classe); dans ce second cas, on l'appelle alors attribut.

Visibilité : espace d'un programme dans lequel une variable donnée est accessible.
Les variables peuvent avoir une visibilité de niveau classe : elles sont accessibles depuis toutes les méthodes; on parle alors d'attributs.
Elles peuvent avoir une visibilité de niveau méthode : elles sont définies dans une méthodes, et ne peuvent être utilisées que dans cette méthode.
Elles peuvent également avoir une visibilité de niveau bloc : elles sont définies dans un bloc (par exemplele compteur i dans la boucle `for(int i; i< n;i++)`).