



Bases de POO

1

Partie I: Bases de la POO

1

Paradigmes et langages de programmation

2

Concepts fondamentaux de la POO

3

Introduction à Java

3

Bases de POO

1

Paradigmes et langages de programmation

4

Bases de POO

Petit historique de la POO

le développement de logiciels **orientés objet** (OO) existe depuis le début des années **1960**

Ce n'est qu'entre le **milieu et la fin** des années **1990** que le paradigme orienté objet a commencé à pénétrer le monde de la programmation

L'orienté objet né à la fin des années soixante avec **SIMULA** (Simple universal language)

Le paradigme orienté objet a commencé à pénétrer le monde industriel durant les **années 80**, notamment avec le langage **SMALLTALK**, pour connaître un succès très important à partir des **années 90**, avec **C++ et JAVA**.

5

Bases de POO

Petit historique de la POO

L'idée originelle consiste à proposer un **modèle de programmation** qui soit le plus proche possible du **monde réel** (des **objets** interagissent entre eux)

De nos jours, de très nombreux langages permettent d'utiliser les principes de la POO dans des domaines variés, citons comme exemples: **Java**, C++, C#, Python, Ruby, PHP, VB.NET, Objective-C, Kotlin, etc.

Une connaissance minimale des principes de la POO est donc **indispensable à tout informaticien**, qu'il soit développeur ou non.

6

Bases de POO**Langages de programmation**

Les langages de programmation sont généralement classés en trois niveaux: les **langages machines**, les langages **d'assemblage** et les **langages de haut niveau**

Les instructions en langage machine sont codées en **binaire** et de **très bas niveau**

le **langage assembleur** permet une programmation symbolique «de niveau supérieur». Au lieu d'écrire des programmes en binaire, le langage d'assemblage permet aux programmeurs d'écrire des programmes en utilisant des codes **d'opération symboliques**.

7

Bases de POO**Langages de programmation**

Les langages de **haut niveau** ont été développés pour permettre aux programmeurs d'écrire des programmes plus rapidement qu'en utilisant des langages d'assemblage

8

Bases de POO

Paradigmes de programmation

Un paradigme de programmation est une **façon de conceptualiser** ce que signifie effectuer un calcul et comment les tâches qui doivent être **effectuées** sur un ordinateur doivent être **structurées et organisées**.

9

Bases de POO

Paradigmes de programmation

Un paradigme de programmation est une **façon de conceptualiser** ce que signifie effectuer un calcul et comment les tâches qui doivent être **effectuées** sur un ordinateur doivent être **structurées et organisées**.

Un algorithme est un ensemble **d'étapes** qui opèrent sur des **données** pour arriver à une **solution d'un problème**.

Différents paradigmes de programmation impliquent d'avoir la solution d'un problème en **combinant** des **données** et des **algorithmes** de **différentes manières**

10

Bases de POO

Paradigmes de programmation



11

Bases de POO

Programmation procédurale VS POO

	Paradigme procédurale	Paradigme orienté objet
Programmes	Le programme principal est divisé en petites parties selon les procédures	Le programme principal est divisé en petits objets en fonction du problème
Données	Les attributs et les procédures sont séparés	Les attributs et les méthodes sont contenus dans un seul objet
Masquage de données	Il n'y a pas un moyen idéal pour masquer les données	Permet de restreindre l'accès à certains attributs et/ou méthodes (Masquage)
Accès aux données/procédures	Données locales, globales, passages de paramètres	Utilisation des spécifications d'accès public, private, et protected aux données et aux méthodes
Communication	Passage de paramètres	Communication via des messages
Résolution de problème	Ensemble des étapes dans un ordre d'exécution bien défini	Représentation du problème en termes d'objets et de comportement

12

Bases de POO

Réutilisation de code

En informatique le terme **réutilisation du code** consiste à utiliser un **logiciel existant**, de **connaissances** sur ce logiciel, ses **composants logiciels** ou son **code source**, pour créer de nouveaux logiciels sans avoir à réécrire à nouveau.

Cela peut se faire par la création des **fonctions** qui peut être **appelé** ou **invoqué** dans le même programme, ou regroupés en **bibliothèques** qui peuvent, à son tour, être utilisés aux besoins dans des différents programmes ou API sans avoir à les réécrire.

13

Bases de POO

Réutilisation de code

La réutilisation de code permet

- de **réduire** de manière considérable le **temps** de développement d'un programme par le programmeur,
- réduction **de l'effort de test** du code,
- **faciliter** la tâche de programmation
- d'améliorer la **lisibilité et la structuration** du code.
- de **réduire le code** de tel sorte il devient plus court et synthétique.

La réutilisation du code est **disponible** dans la quasi-totalité des langages de programmation. Elle est basée sur la notion de "**modularité**"

14

Bases de POO

Modularité

un **module** consiste à **une entité de données et d'instructions** qui fournissent une solution à une partie bien définie d'un problème plus **complexe**. Un module peut faire **appel** à d'autres modules, leur **transmettre des données** et **recevoir** des données en retour.

L'ensemble des modules **reliés** doit alors être capable de résoudre le **problème global**

La plupart des langages de programmation permettent de **décomposer** les programmes en **sous-programmes**, **fonctions** ou **procédures** plus simples et plus compacts.

15

Bases de POO

Modularité

Parmi les avantages de la programmation modulaire on peut citer les suivantes :

- ☆ Meilleure lisibilité du programme
- ☆ Diminution du risque d'erreurs
- ☆ Possibilité de tests sélectifs de chaque module
- ☆ Dissimulation des méthodes
- ☆ Réutilisation de modules déjà existants
- ☆ Simplicité de l'entretien et la modification du code
- ☆ Favorisation du travail en équipe par affectation de la programmation des modules à différentes personnes ou groupes de personnes.

16