SI4 –
Bases de la programmation

Langages

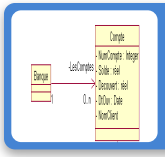
IDE

Préambules à la POO

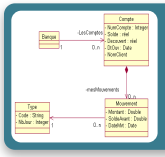
Préambule

* Avant d'aborder la POO :
* Point sur les langages
* Les IDE
* Le management par les Frameworks
* Réutilisation de briques
* Utilisation programmation procédurale

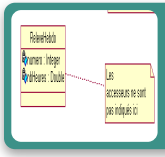
Couches



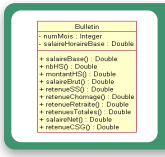
Interface de l'application



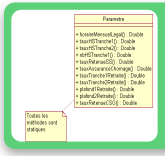
exécutable



OS hôte



VM



OS

Cheminement

Programmation procédurale

* Elle intervient dans tous les langages
* Elle est nécessaire à n'importe quel traitement
* C'est la base de la programmation
* C'est ce que vous apprenez en semestre 1 (SI4)

COBOL, C, batch, bash, C#, JAVA, CL, ASM, Swift, C++

Points sur les langages

* Langages de bas niveau : xASM, 16, 32, 64
* L3G : C, Pascal, COBOL, …
* L4G : SQL, ADA, …
* Langages objets : C++, …
* Langages managés (framework) : C#, JAVA, ObjC, Swift
* Framework : cadre de travail
* Snippets abondants …

Framework

* Pas une simple bibliothèque logicielle
* Impose des patterns : modèles de développement => assemblage de classes à moitié finis …
* En implémentant le pattern, le développeur contribue à finaliser le modèle et à le rendre conforme :
  * Au Framework
  * À l'application métier

Langages managés

Fonctionnalités avancées :

Test unitaire

Garbage collector (desallocation)

Memory management (allocation)

Virtual machine infrastructure (LLVM)

Les IDE

VS 2015

NetBeans

Eclipse

Xcode 7.x

Les IDE

Fonctions communes :

Applications/mobiles

Simulateurs

Débogage/test

Versionning (TFS, git, …)

Design (composants prédéfinis)

Génération UML

Programmation évènementielle

* Dépend des interactions utilisateurs (programmation réactive)
* Ne dépend pas de la POO
* SDL propose de gérer des évènements sans POO
* Le programme capture les évènements, et lance les traitements
* Utilise toujours la programmation procédurale
* Programmation basée sur des fonctions

POO

* Classes utilitaires/métiers
* Réutilisation
* Principe d'héritage
* Encapsulation
* Polymorphisme
* Visibilité des composants
* Intègre l'évènementiel
* Intègre la programmation procédurale

POO

* Ne veut pas dire forcément interface graphique
* Traitement possible en background (&)
* Traitement possible en ligne de commande (PowerShell)
* Implique plus de réflexion/conception (UML)
* Lourd pour les petits traitements
* Indispensable pour les applications actuelles

POO

* Présente du smartPhone/tablette aux mainframes
* Présente dans de nombreux projets
* Rend désuet de vieux langages (COBOL, GAP, RPG)
* Niveau d'abstraction élevé, programmation haut niveau
* Tant à donner des possibilités aux débutants, résultats rapides (RAD, agilité, …)

POO : Warnings

* Les langages managés tendent à supprimer les notions d'espace et de temps
* Gestion automatique mémoire (Garbage Collector, ARC, …)
* Gestion persistance transparente
* Cloud, SaaS, PaaS, à portée de main …
* Bien penser que les couches du dessous existent toujours …
* L'enjeu est peut être là !

POO et BD

* S'interface parfaitement avec les SGBDR
* Apporte la touche objet à du relationnel
* La « sérialization » fait le reste …
* Même PHP propose l'objet …

POO

* Problématique du programmeur :
* Quels langages, quels niveaux de pratique(s)
* Formations, auto-formation
* Environnement de travail : SSII, PME, collectivité, service public, indépendant, formateur, …
* Avenir ?

Conclusion

* En informatique savoir utiliser le produit le plus en adéquation avec la tâche à automatiser et se moquer du reste !

* Sinon, cela voudrait dire que tout le monde pense au moins « objet », sans parler des traitements distribués et du parallélisme massif !