Programmation Par Objets

Historique et motivations

Origines

- **Simula (Oslo, 1967)**
  - Projet : simulation du port d'Oslo
  - Logiciel = {Objets interagissant}
  - Objet = données + opérations
  - Programmer = mimer la réalité
- **Smalltalk (Utah, 1969) : «tout est objet»**
  - Objets d'applications: client, inventaire, bilan, commande
  - Objets graphiques et d'interface : points, rectangle, boutons, fenêtre, menus
  - Objets de base : integer, booleans, arrays, strings, sets, streams, files
  - Objets outils : File manager, compiler, ...

Depuis…

- **Langages**
  - C++ (1983). Compatibilité ascendante avec C.
  - Java (Sun 1995) : entre Smalltalk et C++
  - Et d'autres : Ada 95 (orientés génie logiciel), C# (.Net), PHP 4, ...
- **Bases de données**
  - GEMSTONE, Oracle 8 (et même Cobol-Objets)...
- **Méthodologies : UML**
- **Outils de développement d'applications**
  - Visual Studio (.NET), Eclipse (java), WinDev, ...

Java (Sun 1995)

- Entre Smalltalk et C++
- C++
  - Syntaxe familière à la C/C++
  - fortement typé
  - gestion des exceptions
- Smalltalk
  - "tout objet"
  - machine virtuelle
  - gestion automatique de la mémoire: garbage collector (pas de pointeurs explicites)
- Portable
  - machine virtuelle (bytecode)
  - standards (arithmétique IEEE 754, Caractères 16 bits Unicode)
- Intègre le réseau
  - Applets (clients WEB) / Standalone applications (interprète java)
  - Code mobile (internet), chargement dynamique de code
- Nombreuses bibliothèques de classes (JDK : Java Development Kit)
  - java.util : SD (listes, piles, itérateurs...) ...
  - java.sql : accès aux BD (jdbc)
  - java.awt, javax.swing : graphique et interface
  - java.rmi : réseaux et objets distribués
- Free:
  `http://java.sun.com`

Génie Logiciel

« Ensemble des activités de conception et de mise en œuvre des produits et des procédures tendant à rationaliser la production du logiciel et son suivi »

- logiciels de grande taille et de longue durée de vie
- coût du soft >> coût du hard
- coût de la maintenance > coût du développement
- améliorer la productivité:
  constante (15 lignes/jour) quelquesoit le langage utilisé (ASM ou langage évolué)!
- améliorer la **qualité du logiciel**

Qualités externes (pour le client)

- **Validité**
  - le logiciel répond au cahier des charges
- **Fiabilité**
  - le logiciel est robuste, résiste même aux conditions anormales
  - => notion d'exception
- **Efficacité**
- **Portabilité et compatibilité**
- **Evolutivité**
  - capacité à prendre en compte des changements de spécifications (nouveaux besoins, nouvelle configuration...)
  - c'est la "partie noble" de la maintenance, l'autre partie est la correction d'erreurs et concerne le concepteur

Qualités internes (pour le concepteur)

- **Maître mot : la modularité**
  Concevoir un système comme un ensemble de composants logiciels (modules) “indépendants” ou en tout cas à interfaçage limité et explicite.
- **Décomposition**
  - décomposer un système en sous-systèmes plus simples : diminuer la complexité
  - conception structurée descendante (top-down)
  - division du travail et lisibilité
- **Composition**
  - “réutilisabilité” (économie de code et d’effort, productivité)
  - conception structurée ascendante (bottom-up)
  - exemples : bibliothèques de fonctions, d’objets, de composants

Qualités logicielles (suite)

- **Suivi du logiciel**
  - la modularité offre des garanties pour la maintenance évolutive et corrective
  - modifications limitées à un petit nombre de modules (ajout/retrait/remplacement)
  - localité des erreurs

- **Utiliser un langage support modulaire**

Quand les langages de programmation se rapprochent de la conception...

Types de données évolués
Programmation structurée : procédures et fonctions
Modules de compilation séparée (C, packages ADA)
Objets et Classes
Composants logiciels...