Cours Java

Chapitre 1 Java historique et versions

Création 1991 chez sun microsystem :

Gausling et naughton car que le C++ et ce n’était pas adaptable au fonctionnement

Nom réservé donc choix : C++-- mais finalement ils ont choisi JAVA (tasse de café)

Echec au début

Web en pleine explosion

Donc favorise java en début 1995 Mai 1995 devient openSource

2000 sortie officielle de Java

Chapitre 2 les caractéristiques

Langage orienté objet : encapsulation /

Encapsulation = dédier une partie de la gestion du programme a un objet

Java est portable :

Ecrire le code java quand compiler créer un fichier bytecode pas un .exe

Donne un Bytecode qui peut s’exécuter sur plusieurs supports

Non portable obligé de recompiler pour chaque support

Right one run anywere

Multitaches :

Plusieurs parties du programme exécuté en même temps

Utilisé quand il y a beaucoup de donné

Langage Java

Prend du temps de compilation car il créer un bytecode qui doit s’adapter

JRE = Java runtime Environnement

Minimum requis pour exécuter le programme Java , contient la commande permettant d’executer un programme java. Java monProgramme

JVM = Java Virtual Machine

LA commande java permet d’éxecuter une appli lance une JVM incluse dans la JRE.

permet de créer une VM qui va gérer l’exécution de l’app en arrière plan, interactions avec le SE, gestion de la mémoire, optimisation du code

JDK = Java dev kit = JRE + outil de dev

Outils de dev

Compilateur /debuggeur / générateur de doc

Bibliothèque de classes

Documentation outil et classe

Type de programme :

Application autonome :

S’execute avec la commande Java

Point d’entrée : service spécial main

Servlet

Petite app serveur qui répond au requête http

S’execute au sein d’un server d’appli

Applet

Petite app graphique embarué

S’éxecute au sein d’un navigateur

A été a l’origine de Java mais obselète

Java SE (console)

Java EE (web )

Java ME

Structure de programme :

Code Java

Physiquement : des fichiers java organisé en répertoires

Logiquement : des classes organisées en packages

On peut écrire du code java avec un simple éditeur de texte.

Exemple de programme :

Main : point d’entrée

Args : les arguments de main sont des chaines de caractères

Static : méthode appliqué a la classe et nom une instance

Void : la méthode ne retourne rien

System.out.println(« … ») affiche la sortie

Caractère ; est un termineur d’expression

Bibliothèque de base du langage

Les bibilothèques :

Constitue un noyau pour le déploiement

Regroupé par packages

Dev en java

Les packages :

Elément d’organisation qui permet de rangé les clases

Découpe par responsabilité

Racine du projet

La racine du projet est le répertoire qui contient le package racine

Packages rôle :

Ranger par responsabilité

Fonctionnalité

Packages et règles d’import on doit l’importer

Import par défault

Packages classification et règle d’import

Import par défault des java lang

Programme détaillé et support

Programme détaillé ou sommaire

Les littéraux valeurs littérales

Types primitif byte (-128 -> 127)/ short( -32000 -> +32000)/ int (-2.1milliard -> +2.1 milliard) /long (codé sur 8 octet )

réels : float(-3.4\*10^38 et 3.4\*10^38 ) double(-1.79\*10^308 -> 1.79\*10^308 ) booléen(0-1) char (‘ ‘) String

Stocké dans la pile les variable primitive et les objet au niveau du tas

Appel a des méthodes avec un point

Définir une variable : int var1= 1

Nom de variables sont autoriser suivant

Tt les lettres de a a z

Les chiffres de 0 a 9

Les charcatères $ et \_

Les caractères unicodes

Règles les chiffres ne sont pas autoriser comme premier charactère

Conversion implicite

Byte < short < int <

Blocs et instructions :

Bloc et instructions :

Une variable n’est visible que dans son bloc et ses sous bloc l’instruction peut être soumis a des instructions

Approche itérative

Nous manipulons des concepts et non des valeurs isolées

Plus de définitions de variables

Approche objet

Les changements impacte peu le code

Constructeurs

Constructeur : initialise les objets de la classe on peut le créer rapidement avec un click droit generate dans l’ide

Surcharger un constructeur on peut créer un constructeur avec moins d’attributs pour lui mettre des valeurs par défault dans la surcharge on rappelle le constructeur avec this(param du construct)

Le this permet d’atteindre

Destruction

Automatique en Java

Une fois qu’il n’est plus référencer

Effectué par le garbage collector

Libère la mémoire

Interface il n’y a que des méthodes elle contient que des comportements

Il faut mettre implements devant la classe

L’encapsulation et l’association

Une classe doit être simple d’utilisation but de l’encapsulation masquer les traitement interne possibilité de restreindre la visibilité des attributs et méthodes.

Mettre les méthodes privé sauf celles qui intéresse les autres classes (classique)

Quand il n’y a rien la visibilité c’est package

Protected quand il y a héritage

Class

Soit public

Soit en mode package

Getter setter en privé ou public intérêt des getters setters

L’association :

Une classe peut porter des des attributs de type classe

Héritage

Permettre la réutilisation de code existant

Duplication de code difficile a maintenir / créer un type générale

Mettre la classe fille en extend / mettre en protected les attributs de la classe mère

Protected pour que ça reste privé mais que la classe fille y est accès

Une classe ne peut avoir qu’une superclasse

Super peut récupérer la méthode de la classe parent contraintes le type de retour identique, visibilité égal ou supérieur

Polymorphisme

Interface gère les comportements les classes mère ne connaissent pas leur fille

Downcasting

Besoin de convertir dans une autre classe pour avoir accès au méthode spécifique

Interface donne un contrat permet de passer au-dessus du problème de cast

Mot clé de l’implementation : implements

Existe car héritage multiple n’éxiste pas en java donc on peut mettre plusieurs interfaces

Bloc statique est executé une seule fois la première fois