Formation DevOps

Penser à bien valider sa présence sur : http://quest.ajc-formation.fr:8000 (matin et après-midi).

Code wifi : 7894561230

Algo et Objet

Prof. : ASSOUS Steeve

Code : 3404

Exercice

Conception d'algo :

ANALYSE -> **CONCEPTION** -> **PROGRAMMATION** -> **TEST**

En POO :

Objet contient 3 concepts fondamentaux :

- **polymorphysme**
- **encapsulation**
- **héritage**

Objets de bases :

- données en entrée et résultats intermédiaires
- régle opératoire
- ici on parle du nom commun objet (différent de POO)

Un objet contient :

- un **identificateur** (nom)
- un **type** (entier, numérique, caractère ...)
- une **valeur**

Types simples

Ce sont les **caractères**, **textes** (ou **chaînes de caractères**), **logiques**, **bouléens**, **indicateurs** et les **nombres**.

Types composés

Ce sont les **fiichiers** et les **tableaux**.

Actions de base

Elles sont :

- affectation : attribuer une valeur à un objet
- affectation particulière : initialisation, incrémentation/décrémentation

Ex :

```
// Debut

    // DECLARATION
        variable : type

    // INITIALISATION
        variable = 0

    // CORPS DU PROGRAMMES
        instruction 1
        instruction 2
```

...

Autres actions de base

- lecture : à partir du clavier / fichier
- ecriture : écran / sur un support externe
- concaténer : associer variable avec texte

Choix ou Structures Conditionnelles

On a le choix entre plusieurs actions :

- structures conditionnelles pures

```
ex: SI condition
        ALORS
            action 1
    FIN SI
```

- structures conditionnelles alternatives

```
ex: SI condition
        ALORS
            action 1
        SINON
            action 2
    FIN SI
```

- structures conditionnelles alternatives imbriquées

```
ex: SI condition
        ALORS
            action 1
        SINON
            SI cond
                ALORS
                    action 2
                SINON
                    action 3
            FIN SI
    FIN SI
```

Répétition

Sert à repeter n fois une action :

- TANT QUE
- REPETER ... JUSQU'A : effectue l'action une fois avant de vérifier la condition
- POUR

Tableaux à une dimension

Ils peuvent être préremplis.
L'indexation commence à 0.
Un tableau de N case aura pour indice i de 0 à N-1.

Tableaux à deux dimensions

Lorsqu'un traitement utilisie plusieurs tableaux à une dimension, on utilisera un tableau à deux dimensions.
On défini T(i,j) avec i lignes et j colonnes.
Pour lire un tableau à 2 dimensions on doit imbriquer deux boucles l'une dans l'autre :

```
ex: POUR ligne DE 0 A N-1
        POUR colonne  DE 0 A M-1
            AFF ligne, colone, T(ligne,colonne)
        FIN POUR
    FIN POUR
```

Les Concepts de l'Approche par Objets

3 concepts nécessaire à la POO :
- encapsulage
- héritage
- polymorphisme

Classe

Une classe est un modèle qui décrit une abstraction.
Permet de recenser une série d'éléments communs décrivant un concept précis.
Elles décrivent les propriétés des objets.
Analogie avec plan d'un pavillon, avec chaque pavillon qui peut avoir une couleur différente (plan = classe, pavillon = objet).
Elles sont de 2 types :
- propriétés contenant de l'information :
  - o modifiés au travers de la méthode
  - o ces propriétés sont les "attributs"
- propriétés actives
  - o fournissent un service et peuvent modifier l'état d'un Objet
  - o ses propriétés sont les "méthodes"

ex: classe compte
propriétés (attributs) : solde, decouvertAutorise, dateOuverture
propriétés actives (méthode) : crediter(), debiter(), obtenirSolde(), calculAgios()

Il est possible de conférer un type à un attribut ou à un argument d'une methode.
Un type désigne un ensemble de valeurs que peut prendre un attribut ou un argument : entier, chaine, booléen, reel...
Une classe peut également servir de type.

Objet

C'est une matérialisation de la classe (concrétisation).
L'acte de concrétiser une classe s'appelle le mécanisme d'instanciation.
On dit qu'on instancie une classe ou que l'on crée un Objet.
Objet = instance de Classe.
ex: MaVoiture:Automobile
marque = “Peugeot”
modèle = “607”
puissance = 10
couleur = “rouge”

Encapsulation

Pour chaque attribut et ou méthode d'une classe, on précisera son niveau de visibilité.
C'est l'idée de masquer le fonctionnement exacte du mécanisme bien que je puisse utiliser ce mécanisme (idée de la grosse boite noire)

public	+	élement visible par tous
protégé	#	élément visible par les sous-classes de la classe
privé	-	élément visible seulement par la classe
paquetage	~	élément encapsulé visible uniquement dans les classes du paquetage

ex:

```
Personne                | |=> classe
-------------------------------
-nom : chaine           | |
-prenom : chaine        | |=> attributs
-dateNaissance : Date   | |
-------------------------------------------------------
+calculeAge() : entier                      | |
+getNom() : entier                          | | => methode
+setDateNaissance(dateNaissance : Date) | |
-------------------------------------------------------
```

+getNom() est un accesseur en lecture/getter
+setDateNaissance est un accesseur en ecriture/mutateur/setter

Propriétés de Classes

Un attribut de classe est partagée avec sa valeur par l'ensemble des instances.
Une méthode de classe est une méthode qui est directement liée à la classe elle-même.
Au sein d'une telle méthode, seuls les attributs dits "de classe' sont accessibles.
Un attribut de classe est un attribut transverse partagé par toutes les instances (ex: TVA pour des produits en ventes, si ils ont la même TVA).

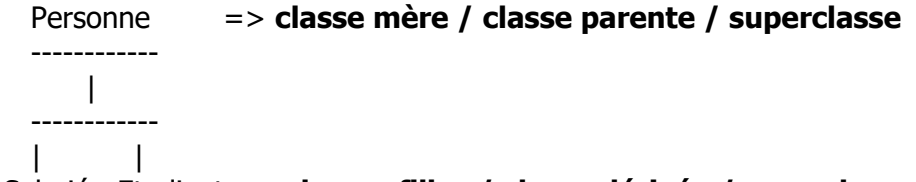
ex: reprenant la classe Personne. l'attribut -majorite : int = 18. On la met en valeur en la soulignant.

ex: en reprenant la classe Perssonne, une méthode de classe +changeMajorite(nouvelMajorite : entier)

Généralisation/Spécialisation

La généralisation/specialisation est une association particulière.
Elle ne porte pas sur les instances mais sur les classes.
Elle exprime que les instances d'une classe (classe fille) sont également des instances d'une autre classe (classe mère).
La généralisation a pour but de factoriser des attributs et des méthodes communs à plusieurs classes.

ex: on peut créer une classe spécialisé de personne : Salarié avec -salaire. Une autre Etudiant avec -note et -niveau.



Plus on remonte : généralisation. Plus on descent spécialisation.

On peut creer des classes stériles qui ne peuvent avoir d'enfants.

Héritage

Les instances d'une sous-classe sont aussi instance de ses surclasses.
Ces dernières sont donc également décrites par les attributs et méthodes introduites dans les surclasses.

Par conséquent, une sous-classe hérite des attributs et des méthodes de sa sur-classe.
Elle n'hérite que attributs et méthodes publiques (ou protégés : #) mais pas les publiques.

Cet héritage provient de la relation Généralisation/Spécialisation.

Polymorphisme

Le polymorphisme s'inscrit dans une logique de Généralisation/Spécialisation.

Le polymorphisme est le mécanisme qui consiste à appeler la méthode en fonction du type de l'objet instancié et non pas du type de l'objet réellement déclaré.

ex:

```
Salarié sal1;
sal1.affiche(); => va chercher la méthode affiche() de la classe Salarié

Personne p1;
p1.affiche(); => va chercher la méthode affiche() de la classe Personne

Personne TPers[3];
tPers[0] <- p1;     => possible car p1 est une Personne
tPers[1] <- sal1;   => possible car sal1 est une Personne
```

tPers[0].affiche(); => va chercher la méthode affiche() de la classe Personne
tPers[1].affiche(); => va chercher la méthode affiche() de la classe Salarié car on appelle le type de l'objet instancié (Salarié) par réellement déclaré (Personne)

Classe abstraite

Une classe “concréte” est une classe instanciable car elle décrit un modèle complet, tous les attributs et toutes les méthodes sont totalement décrits.

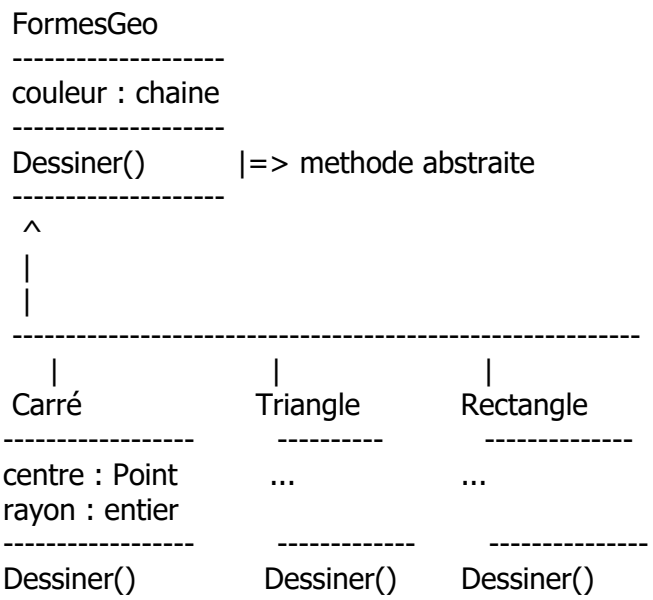
Une classe abstraite provient de la “Généralisation”.

Il s'agit d'une factorisation de propriétés communes.

Une classe abstraite ne peut avoir d'instances directes.

Les méthodes d'une classe abstraite peuvent être décrites de façon limitée à la signature de méthodes.
On parle de “Méthode Abstraites”.

ex:



Traditionnelement, on indique une classe abstraite en l'écrivant en italique.
Ici la classe FormesGeo est abstraite (correspond à rien), mais permet de factoriser l'attribut couleur.

Forcer à avoir une methode abstraite (donc vide) force a avoir la méthode Dessiner() défini dans les sous-classe.
On garanti le **polymorphisme**.
On impose un “schéma”.
Si la sous-classe est elle aussi abstraite il n'est pas nécessaire de définir les méthode abstraite de la surclasse.

Interface

Une interface est une classe abstraite ne contenant que des signatures de méthode publiques.

Une interface est réalisée par une classe.

On dit qu'une classe implémente une interface.

Une interface est employée pour décrire les fonctionnalités d'une classe ou d'un composant.

C'est un élément fondamental de la conception objet.

On indique formellement qu'une classe implémente une interface avec une flèche en pointillées ou le formalisme lollipop Classe Interface ---O)--- Classe

Ligne de Commande

Prof. : ASSOUS Steeve

Code : 2518

Sommaire

Présentation de Linux/Unix
Système de fichiers : l'arborescence
Manipulation de fichiers
Edition de fichiers texte : VI
Redirections et Filtres
Recherche de Fichiers via “find”
Rercherche de motifs “grep”
Gestion des processus

Présentation

Définition

C'est un OS

Programme ou ensemble de programmes et d'API, interface matériel et applications.

Disponible pour de simple micro (PC, MAc, Atarie...) jusqu' au gros syst_me (IBM Z Series) et même dans des PDA

Unix ? Linux ? Telle est la question.

Historique de UNIX

1969 : Bell Laboratoire; Ken Thompson travaille sur MULTICS; Bell Lab abandonne le projet
K. Thomspon renomme UNICS et dev. sur DEC PDP-7
1974 : refonte du syst. en language C; Unix gagne la fav. des univ.
1978 : V7 annoncée
1979 : cout des licences incite Berkley à continuer ses travaux. Création de BSD; le Darpa utilise BSD Unix
1983 : AT&T met en vente la version commercile de Unix Syst. V
1984 : Wenix 1er Unix sur PC
1991 : Apparation des premiers clones Unix comme Linux et FreeBSD
1992 : Sun sort Solaris

D'où vient Linux ?

1985 : création de la FSF par R. Stallman
Copyleft et GPL avec 4 libertés fondamentales :
exec. le programme pour tout usage commercial ou non et par n'importe quel type de personne ou d'organisation
étudier le fonctionnement du pregramme et l'adapter
redistribuer des copies gratuitement ou non
améliorer le programme et publier les améliorations
1991 : création de Linux par Linus Torvald

Caractéristiques

Unix/Linux est un OS :
- multi-taches
- multi-utilisateurs
- multiplateformes
- stable

Les composants de base d'un Unix/Linux sont les noyau (kernel) et les outils (shell et commandes).

L'OS a pour principales tâches les points suivants :
- gestion de la mémoire
- accès aux périphériques
- accès disque/ système de fichiers
- gestion des programmes (processus)
- sécuriité / accès aux données
- collecte d'information système : statistiques

LAMP pour les serveurs : Linux (OS) Apache (serveur Web) MySQL (SGBDR) PHP (language de programmation serveur).

Quelle distribution ?

Le choix d'une distribution dépend de :
- du cout
- de la suite logicielle
- de la compatibilité matérielle
- des outils d'administration
- mais aussi les préférences de chacun

Les distributions “grands publics”
- Fedora (distribué par Red Hat)
- Open Suse
- Mandriva (anciennement Mandrake)
- Ubuntu

Les distribution “pro.”
Red Hat
- Suse
- Debian
- Slackwave

Les distributions “mobiles”
- Knoppix
- Mandriva Flash
- Fedora Live CD

Les distributions “spécialisées”
- Tombsrbt
- µLinux

Connexion : consoles et terminaux

Chaque utilisateur sera doté d'un login et d'un MdP
- root
- règle de mot de passe

La connexion peut se faire vie différentes manères
- console virtuelle
  → Texte
  → Graphique
- emulateur de terminaux
- client ssh

Prompt (Invite shell)

Le shell
- interprétation en ligne des commandes
- il traduit les requètes en actions

Prompt : PS1
- il s'agit de l'invite présente au moment de la saisie de'une commande
- cette invite diffère suivant les environnements et suivant les utilisateurs

ajc1@DebServ:~$
- ajc1 : username
- DebServ : hostname
- ~ : repertoire de connexion (ici HOME)

La Documentation
- man
- info
- option --help
- internet :
  → www.linux.org
  → www.tldp.org
  → www.redhat.fr
- groupe de discution

Deconnexion
Trois manières de se déconnecter d'un terminal texte :
- exit
- logout
- ...

Quelques commandes
Info utilisateur
- who
- whoami
- passwd

Affichage
- clear
- echo

Temps
- date
- cal

Système de Fichiers : Arborescence

Organisation
Un syst. de fichiers (FileSystem) définit comment sont gérés les fichiers par l'OS

Présenté de façon arborescente, le FS est une hiérarchie de répertoires ayant pour racine unique "/"

Organisation logique et indépendante du stockage physique des données.
Complétement transparente pour l'utilisateur.

Diff. rep. :
- bin : contient les commandes
- boot : gere le demarage du système
- cdrom : pour monter le cdrom
- dev : contient les périphériques
- etc : contient les fichiers de config
- home : repertoire des utilisateurs
- lib : contient les librairies
- lost+found : garde les fichiers non correctement fermés en cas de crash

- root : rep. du root
- sbin : commande admin
- tmp : fichiers temporaire
- mnt : montage fichiers
- proc : syst. de fichiers virtuel représentant l'etat actuel du syst.
- sys : syst. de fichier vituels représentant l'état des différents peripheriques.
- var : fichiers dont la taille vaire au cours de la vie du systeme
- ...

"Tout est Fichier" : nomenclature

Unix fait la distinction entre min et maj.

La plupart des caracs. (chiffres, lettres, maj. min. certains signes, caractères accentués) sont acceptés, y compris l'espace (déconseillé)

Adressage Absolut/Relatif

Absolu : chemin identifié à partir de la racine

Relatif : chemin défini à partir du répertoire courant

Rep. :
- courant : .
- parent : ..
- personnel : appelé aussi répertoire HOME
  → définit le rep. de connexion, l'espace personnel de l'utilisateur (/home/nomUser)

Manipulation de Fichiers/Répertoires

Commandes permettant de manipuler les fichers
- pwd : print working directory
- cd : change directory
- ls : liste le contenu du repertoire
  → ls -r : recursif
  → ls -1 : affiche sur une seule colonne
- file : renvoie le type de fichier

Répertoire
- mkdir
- rmdir

Fichiers
- touch
- cp
- rm
- mv
- cat
  → cat > nomFichier : permet d'ecrire dans le fichier (attention efface ce qui est écrit)
  → cat >> nomFichier : écrit à la fin du fichier
- more
- head/tail : affiche les 10 premières/dernières lignes du fichier
  → head/tail -x : affiche les x premières/dernières lignes
- cut : affiche une partie des lignes du/des fichier(s) en argument
  → La selection peut se faire par champ ou par caractères
  → cut -d: (délimiteur ici :) -f3 (champ ici 3)
  → cut -c1-8 (caractères de 1 a 8)
- wc : comptabilise le nombre de lignes, mots, caractères

ex :
more /etc/passwd : affiche tous les utilisateurs inscrits
ajc1:x:1151:1003::/home/ajc1:/bin/bash
nomUser:PrésencePasseword:uid:gid:NomGECOS:RepertoireDeCMX:PossibilitéDeSeConnecter

drwxr-xr-x 4 ajc1 unix 4096 oct 5 12:13 dos
- d : type (d : directory, - : fichier ordinaire, l : lien symbolique, b: block, c : caractere)
- rwxr-xr-x : permission User|Group|Other
- 4 : nombre de lien physique
- ajc1 : user proprietaire
- unix : groupe propriétaire
- 4096
- oct 5 12:13 : horodatage
- dos : nom

ls -l | cut -c1-10,41- : liste les fichiers avec détails | en ne gardant que les 10 premiers caractères puis les caractères à partir du 41e.

wc fichHello : affiche le nombre de lignes, mots et caractères dans fichHello
2 5 27 fichHello : 2 lignes, 5 mots, 27 caractères
wc -l /etc/passwd : affiche le nombre de ligne dans /etc/passwd, correspond au nombres d'utilisateurs déclarés.

Permission

rwx :
- r : readable
- w : writable
- x : executable

Pour un fichier normal :
- r : contenu peut être lu, chargé en mémoire, visualisé, recopié
- w : le contenu du fichier peut etre modifié, on peut écrire dedans. La suppression n'est pas forcément liée à ce droit (voir droit rep.)
- x : le fichier peut etre executé depuis la ligne de commande, s'il s'agit soit d'un programme binaire (compilé), soit d'un script

Pour un repertoire
- r :
- w : droit de suppression
- x : droit d'accès

chmod : modifie les permissions
- chmod u+x : ajoute le droit d'executer à l'utilisateur
- chmod g-w : retire au groupe le droit de lire
- chmod og+x : ajoute au autres et au groupe le droit d'executer
- chmod u=rwx : donne à l'utilisateur le droit de lire, écrire et executer
- chmod a-x : retire à tout le monde le droit d'executer

chmod peut aussi être utiliser avec une notation octale :
- r : 4
- w : 2
- x : 1

Shell

Role et Comportement du Consultant

Red Hat System Administration

Installation et Configuration de Microsoft Windows Server 2012

Savoir se Présenter, les Nouvelles Compétences Acquises

Cloud

L'Ingenieurie DevOps sur Amazon Web Services

Oracle SQL et Exploitation

DevOps

Docker

Jenkins

Ansible

Puppet