Java

1) Introduction

- Bref Historique :
    - 1991-1993 : Projet "Oak", inventé par James Gosling.
    - Orienté contrôle d'appareils domotique, indépendant de l'architecture sous jacente.
    - Echec industriel du projet, malgré son architecture innovante.
    - Repris en 1995, par Bill Joy, et nommé "Java" (à cause de l'énergie du café…).
    - Réorienté applications graphiques web sécurisées, interprétées dans un browser.
    - Architecture applicative 3 tiers (byte-code, machine virtuelle, machine physique).
    - Philosophie du "Write Once, Run Anywhere !": portabilité du code compilé.
    - Succès en flèche avec l'avènement de l'Internet (deuxième partie des 90's).
- La plateforme Java:

J2SE	- JavaBeans: Modèle de composants objets suivants des standards établis. (Nomenclature, JAR, Réflexivité des propriétés objets, évènements objets) - Applets: Applications graphiques sécurisées embarquées dans des pages Web. (principe du "*code on demand*" != "*client/server*" classique) - Applications "*StandAlone*": Lancées par une commande dans un environnement système. (pas de restriction sur les services comme c'est le cas pour les Applets) - Librairies et extensions du langage (Swing, XML, Securité, LDAP, JDBC, Graphismes, Media, …)
J2EE	- Servlets (processus générique standard "*Question/Réponse*", pour le Web). - JSP: "*Java Server Page*" (équivalentes aux pages de Script PHP). - TagLibs: Spécifications sur le packaging de composants Web. - JSTL: "*Java Standard Tag Library*" (composants standards Web). - JSF: "*Java Server Faces*" (extension des JSP au sein d'une API MVC, Model Vue Controler) - EJB: Enterprise Java Beans: Composants distribués (Design Pattern 2 "*Bridge*") (Séparation "*Stub/Skeleton*" d'interfaces de signature de méthodes, de leur implémentation) - EJB "*Session*" (processus) "*Entity*" (mapping d'objets en base) "*Message*" (évènements) - Définition XML des parties composant une application d'entreprise (Accès base de données/ressources externes, définition des entités associées aux tables, de leurs procédures de persistance, du schéma des classes du modèle, des transactions en bases, de la distribution sur des clusters de serveurs, des éléments de configuration, …).
J2ME	- MIDP/CDLC: applications (Formulaire, Jeux, connexions réseau, …) pour *smartphone* (téléphone intégrant une VM restreinte) et *SmallDevice* (pager, etc…) - JavaCard: Applications pour *smartcard* (cartes à puces intégrant une micro-VM, capable de lancer des mini-programmes simples, dans une architecture restreinte possédant des *entrées/sorties* de données). - Personnal Java (applications pour les PDA et autre appareils aux architectures plus évolués)

http://creativecommons.org/licenses/by-nc-nd/2.0/fr/

2) Déroulement des TP 1ère année

1ère Partie

TP 0, Kick Off…… IN YER FACE !!!

- Installation de l'environnement de développement (JDK 1.6.*/1.7, Eclipse Europa).
- Présentation rapide d'Eclipse (IDE, Explorer, Perpectives, View, Debugger, Runtime, Console, Task, Updates, plug-in, préférences (Syntax colors, formatage du code, Compilateur (exception, warning, ignore, …), tests unitaires, server), configuration des projets, …), et comparaison avec NetBeans (fonctionnalités, politiques, en entreprise, frameworks et server, …).
- Exemple d'un package de model avec ses classes entités, illustrant les paradigmes classiques du développement orienté objet en Java J2SE :

  Classes, Interfaces, Classes Abstraites, Instances, Héritage, Constructeurs et chainage, Surcharge, Redéfinition, Polymorphisme, Référencement objets et cardinalités 1-1/1-n avec un exemple simple de Collections (vu en profondeur dans le TP3), Encapsulation avec getters (accessors)/setters (mutators), …
- Grammaire usuelle du langage: signature de classe/interface, condition, boucle, commentaires, javadoc, génerics et Pré-Typage du compilateur, …
- Présentation des classes standard du language (package java.lang.*) : String, primitives/Wrapper, … et de certaines classes du package java.util (Date, List, …)
- Déroulement de séquence d'une application Java présentant les appels d'instanciation lors de l'exécution de la Thread du Runtime.
- Discussion autour de la gestion mémoire au sein de la Machine Virtuelle, et de la structuration des packages de projet, et d'UML.

TP 00, Design Pattern State Machine !!!

- Evolution du contenu du TP précédent, afin d'intégrer une implémentation simple mais standard d'une State Machine (DP de famille « comportement ») pour les états de commande.
- L'exemple s'articule autour d'une Enumération des différents états possibles, de deux Interfaces (une d'état héritant d'une autre de validation – ayant sa propre Exception associée) et d'une classe de StateMachine (ayant la charge de centraliser les changement d'état).
- Présentation de la logique de segmentation des responsabilités dans les différents package (contrats d'interface, bascule d'état, gestion d'erreurs, …).
- Exemple de traces propres en consoles.

CC 1999-2012
Droits réservés

http://creativecommons.org/licenses/by-nc-nd/2.0/fr/

2ème Partie

TP 1 : Interfaces graphiques Swing/AWT, modélisation MVC !

- Démarrage progressif sur une petite application graphique en Swing.
- Illustration de certains Design Pattern (création, structuration, comportement). Producteur / Consommateur, MVC (Model View Controler), ...
- Présentation/utilisation des principales classes/packages du langage (String, Number, Formater, BigDecimal, Arrays, util, Event, GUI...).
- Discussion sur les alternatives de Frameworks d'Interfaces Graphiques.
- Exercice pratique.

TP 2, Structuration de données 1ère Partie: JDBC (Java DataBase Connectivity)

- Présentation de l'API JDBC permettant d'interfacer le dialogue avec les bases de données.
- Discussion sur la stratégie de développement des contrats d'Interfaces en Java.
- Mise en place d'une application se connectant à un SGBD (MySql, par défaut, ou autre).
- Présentation de composants Swing (JTable, JToolbar, JMenu, JDesktopPane).
- "Inner Class": Exemple d'utilisation, mécanismes Object / Runtime mis en œuvre.
- Présentation du package d'introspection/réflexivité en java.
- Exercice: Création d'un jeu de classes implémentant une Façade (Design Pattern 2 : « Structuration ») de dialogue JDBC (module d'opérations CRUD) avec des appels réflexifs.

TP 3, Structuration de données 2ème Partie: XML

- Comparatif entre avec JDBC et XML.
- Cas d'utilisation des parsers SAX (en mode itératif) et DOM (en mode récursif).
- Implémentation du mapping XML avec la Réflexivité/Introspection (Class, Method, Field, ...).
- Intégration dans un parser SAX implémentant une Factory (design pattern 1 : « Création »).
- Automatisation par mécanismes réflexifs avancés (discussion autour de la génération de structures de dimensions 1 (List/Set) et 2(Map)).
- Exercice: Suite de l'implémentation de la Facade JDBC consolidée par une Factory réflexive.

TP4, Best practices & intégration de libraries externes

- Développement d'un moteur de recherche à base de procédures réflexives.
- Présentation et cas d'utilisation de library complémentaires (Jakarta Commons, etc...).
- Méthodologie, Politiques de développement, Stratégies industrielles.
- Debriefing sur la première partie des TP et le développement en mode client lourd
- Présentation rapide de la suite/2ème partie des TP sur le développement Web
- Question/Réponse projet avant les soutenances de projet.

CC 1999-2012
Droits réservés

http://creativecommons.org/licenses/by-nc-nd/2.0/fr/

3ème Partie

TP 5, Environnement HTTP

- Installation, configuration, présentation des composants du serveur Tomcat (version 7)
- Présentation des HTTPServlets et des pages de scripts JSP (Java Server Pages).
- Rappels divers sur le développement Web, et le dialogue HTTP.
- Configuration des application/du serveur dans les fichiers Web.xml/Server.xml.
- Filtres HTTP: exemple d'implémentation d'une Chain of Responsability (design pattern 3)
- Présentation et Stratégie des Tags Lib et de JSTL (Java Standard Tag Library).
- Discussion autour des exemples du serveur.
- Questions/Réponses sur le projet EpiMarket

- Exercice: Implémentation d'un caddie JSP comme tiers Client dialoguant avec notre Facade JDBC.

TP 6, Java Server Faces et le développement Web

- Présentation du Framework JSF optant pour une approche MVC au développement Web.
- Comparatif avec d'autres solutions de même type (et notamment Struts).
- Exemples concrets de Méta-Traitements effectués par un Singleton (design pattern 1) Controller.
- Eléments principaux: Managed Beans, TagLib et implémentation Web, description XML.
- Configuration de l'environnement de développement Web sous Eclipse.
- Déploiement d'application de tests/exemples.
- Mise en place de la coquille minimale servant à démarrer le projet EpiMarket en JSF.
- Conseils et orientations pour le projet Epimarket
- Exercice: migration de notre projet vers l'architecture JSF (pages JSP, configuration, ...).

TP 7, Consolidation du développement Web

- Présentation de composants supplémentaires fonctionnels pour le développement Web (Itext et la génération de fichiers PDF/RTF, POI et la génération de document Excel, JFreeChart et les diagrammes de statistiques, etc...) .
- Segmentation des packages du projet, et des appels client.
- Présentation des extensions Jakarta Tomahawk et des composants de la library Ajax.
- Questions/réponses + suivi de développement sur Epimarket

TP8, Débriefing général
(developpement Java, Web, achitectures applicatives, bibliothèques de développement, ...)

- Analyse du cursus Tek3, présentation du cursus Tek4
- Consolidation des projets avant la soutenance.
- Question/réponses.
- Pot de départ !)

CC 1999-2012
Droits réservés

http://creativecommons.org/licenses/by-nc-nd/2.0/fr/

3) Le Java Development Kit JDK

- Composants principaux du JDK (Java Development Kit) et du JRE (Java Runtime Environment):
  - Librairies (Bibliothèques de composants réutilisables)
  - Compilateur: "javac" (Conversion code source / byte-code)
  - Interpréteur: "java" ("Interprétation/Exécution" du byte-code, aussi appelé 'JRE')
  - Bridge Natif (Intégration de code natif: dll)
  - Archives (Gestion sous la forme de JAR utilisant un algorithme ZIP)
  - Clefs (Génération de clé permettant de signer des archives)
  - APT (JDK 1.5: Annotation Processing Tool)

  - **1995-1996: JDK 1.0: ~ 200 classes, ~ 4 Mo compressé**
  Noyau du Langage, Primitives, I/O, Utilitaires, Réseau, Applet Web, Graphisme.

  - **1997: JDK 1.1: ~ 500 classes, ~ 9 Mo compressé**
  Évolution du noyau objet (réflexivité/introspection), Math, RMI, Sécurité, SQL, Zip, JavaBeans, JDBC 1.0, inner classes, JNI.

  - **1998: JDK 1.2: ~ 1500 classes, ~ 20 Mo compressé**
  Améliorations : Performances, sécurité, cryptographie, graphisme, JDBC 2.0, JavaBeans, Audio, Internationalisation, RMI, Sérialisation, Réflexivité, JNI (appels Natifs), JAR, Version, Debug.

  Nouveautés: JFC Swing, Drag&Drop, Java IDL & CORBA, Collections, compilateur JIT ("*Just In Time*": compilation à la volée au moment de l'interprétation du code)

  - **2000: JDK 1.3: ~ 1800 classes, ~ 30 Mo compressé**
  Collections, corrections sur Swing, Amélioration de 30% du noyau réflexif (inclusion de "Hotspot" d'IBM dans la VM),

  - **2001-2002: JDK 1.4 "Merlin": ~ 3000 classes, 47 Mo compressé**
  Intégration de XML (JAXP), JDBC 3.0, java NIO, unification des librairies externes (JNDI, JMS, XML, Java Media, Java 3D, Java Servlet, Java Help, Jini, etc…) au sein du JDK, RegExp, Logging, JAAS (Authentification), outil Java WebStart, Assertions dans le code ("*assert*" expression boolean: instructions)

  - **2004: JDK 1.5 "Tiger": ~ 3300 classes, 44 Mo compressé**
  Intégration d'API: JMX (Managment et Administration), JDBC Rowset, Profiling, Accès concurrentiels, Monitoring et gestion de la VM, API Compilateur, Multilangues, JAXP 1.3, autoboxing des primitives, printf.

  Evolutions de la grammaire: Generics (équivalent des *templates* C++), boucle *for* simplifiée, opérateur *enum,* annotations/metadata pour la génération de code, paramêtres de méthodes "…".

  - **2006-2007: JDK 1.6 "Mustang": ~ classes, 44 Mo compressé**
  Swing widgets, System Tray, Graphisme, Threading, Look & Feel, scripting, WebServices, boites de dialogue, Desktop API(system default props, monitoring, …), accès au compilateur,

http://creativecommons.org/licenses/by-nc-nd/2.0/fr/

4) Un exemple d'application minimale:

```
//Déclaration du package contenant la classe, et correspondant au FilePath
package org.sc.intro;

/* Equivalent des include */
import java.util.*;
import java.io.Serializable;

/**
 * Signature de la classe, qui déclare son seul lien d'héritage, et s'engage sur des contrats d'interface
 * @author Dim
 * @version 1
 */
public class MyMain extends Object implements Comparable, Serializable
{
   //Déclaration d'une variable de la classe
   private String description;

   public static final int VERSION = 42;

   //Point d'entrée d'une application "stand alone"
   public static void main(String argv[])
   {
      System.out.println("Class MyMain: Method main: enter !");

      MyMain newMain = new MyMain(argv);
   }

   //Constructeur portant le même nom que la classe, et ne signant pas le type de retour
   public MyMain(String argv[])
   {
      System.out.println("Class MyMain: Method constructor: enter at " + new Date().toLocaleString());

      if  (argv.length > 0)
         setDescription(argv[0]);
   }

   //Encapsulation de la propriété privée
   public String getDescription()                    {return description;}
   public void setDescription(String description)    {this.description = description;}

   //Respect du contrat d'interface par implémentation de la méthode du Type Comparable
   public int compareTo(Object o)
   {return (o instanceof  MyMain)? getDescription().compareTo(((MyMain) o).getDescription()) : -1;}

   //Redéfinition – override – de la méthode de java.lang.Object
   public String toString()                          {return getDescription();}
}
```


http://creativecommons.org/licenses/by-nc-nd/2.0/fr/

5) Grammaire et éléments du langage:

La syntaxe

- Une grammaire identique au C (opérateurs *for, while, break continue, if - else, switch - case*).
- Les Primitives: des données atomiques (*short, int, float, ...*) != Object (avant le JDK 1.5)
- Les tableaux en Java sont des objets à part entière.
- Gestion des exceptions: (opérateurs *try - catch – finally, throws).*

Les opérateurs particuliers

- "*this"*: Obtenir un descripteur sur l'objet courant.
- "*super"*: Atteindre la classe père d'un composant
- "*null"*: Permet de tester l'existence d'un objet.
- "*instanceof"*: Vérifie la cohérence de type pour un objet.

Les principes

- Pas de pointeurs, ni de manipulation de la mémoire !!!
- Orienté Object: pas d'héritage multiple (problème de conception...), mais des contrats d'interfaces
- Garbage Collector (ramasse-miettes) intégré à la machine virtuelle.
- Modèle évènementiel "producteur-consommateur" (*Event* et *Listener*).

Les mécanismes et principes objet.

- L'*héritage*: Différent du C++: héritage simple (complété par contrat d'interfaces): OO.
- L'*encapsulation*: Couche logique d'accès aux données (méthodes *getter/setters*).
- Le *Polymorphisme*: Plusieur(e)s: types d'un objet, méthodes dans l'arborescence de classes.
- L'*Override*: La *Redéfinition* (réécriture) de méthodes héritées.
- L'*Overload:* La *Surcharge*: services semblables, mais signatures de méthodes différentes
- Le lien dynamique: Accès automatique de méthodes redéfinies dans les classes filles.
- Le *Cast* Transtypage d'un objet selon un type donné. (*throws exception*)

Les concepts associés

- *Class* : Un ensemble de champs et de méthodes
- *Interfaces*: Un contrat de spécification de champs et de méthodes pour les classes.
- *Object*: Des instances de Classes
- *Constructor*: Appel à l'opérateur "*new"*, méthodes spécifiques portant le nom de classe.
- *Modifiers*: Attributs "*public"* "*protected"* et "*private"*, "*static*", "*synchronized*"
- *Abstract Class*: Un modèle intermédiaire entre les Classes et les Interfaces, forçant l'implémentation de méthode "*abstract*" lors de l'héritage

La Sécurité

- Gestion d'exception : (*try - catch – finally*), peut être annoncée par la clause "*throws" (*signature)
- Model étendue, ayant une granularité fine, à partir du JDK 1.2 (cf *SecurityManager*).
- Classe *ConstantPool* (pattern "singleton" de la JVM), Décompilation et Obfuscation.
- Une seule classe *public* par classe Java (protection de la nomenclature des classes produites)
- Les classes sont chargées au moment ou elles sont requises par l'application (cf *ClassLoader*).

http://creativecommons.org/licenses/by-nc-nd/2.0/fr/

6) Packages importants de l'API:

- *java.applet*

  Rôle: Dédié à l'environnement d'un client Web sous la forme d'une applet Java
  Classes majeures : *Applet, AppletContext*
  Fonctionnalité:
  * Contexte du navigateur,
  * Accès aux ressources serveur *(getImage(URL url), getAudioClip(URL url), ...)*
  * Lecture de Méta-Informations *(getCodeBase(), getDocumentBase(), getLocale())*

- *java.awt*

  Rôle: fournir une couche pour les Composants Graphiques de 1ère génération, (JDK 1.0 et 1.1).
  Classes majeures : *Layouts, Color, Component, Event, Font, Graphics, Image, Panel,*
  Fonctionnalité:
  * Primitives d'objets graphiques *(Button, List, Checkbox, TextField, TextArea, , ...)*
  * Containers haut niveau *(Component, Panel, Windows, Dialog, ...)*
  * Responsables de positionnement interne *(GridLayout, BorderLayout, ...)*
  * Objets graphiques de bas niveau *(Rectangle, Point, Polygon, Color, ...)*
  * Objets graphiques de haut niveau *(Toolkit, Graphics, PrintJob, ...)*
  * Evènement et Exceptions spécifiques *(Event, EventQueue, AWTException, ...)*
  Sous-packages:
  * *color, datatransfer, dnd, event, font, geom, im, image, print*

- *java.beans*

  Rôle: Définir et accéder les propriétés communes d'objets Java.
  Classes majeures : *BeanDescriptor, Beans, Encoder, Introspector, XMLDecoder, XMLEncoder*
  Interfaces majeures : *BeanInfo, ExceptionListener, PropertyChangeListener, Visibility*
  Fonctionnalité:
  * Introspection d'objets.
  * Persistence d'objects par sérialization vers un médium cible (file, socket, DB, …),
  * Model évènementiel fort pour l'administration d'état d'objets.
  Sous-packages:
  * *beancontext*

- *java.io*

  Rôle: Fournir des objets permettant la manipulation de canaux générique d'entrée-sortie.
  Classes majeures : *File, InputStream, OutputStream, PrintStream, Reader, Writer*
  Interfaces majeures : *DataInput, DataOutput, ObjectInput, ObjectOutput, Serializable*
  Fonctionnalité:
  * Descripteurs générique de Flux d'Entrée-Sortie (*InputStream, OutputStream)*
  * Différents canaux mis à disposition *(Buffer, Byte, Char, Data, File, LineNumber, Object, ...)*
  * Objets de haut niveau pour la lecture et l'écriture (*Reader, Writer)*

CC 1999-2012
Droits réservés

http://creativecommons.org/licenses/by-nc-nd/2.0/fr/

- *java.math*

  Rôle: Dépasser les limites de valeurs imposées par les primitives.
  Classes majeures : *BigDecimal, BigInteger*

- *java.lang*

  Rôle: Met en place les classes propres au noyau du langage.
  Classes majeures : *Number, String, Object, System*
  Interfaces majeures : *Cloneable, Comparable, Runnable*
  Fonctionnalité:
  * Classes de base du langages *(Class, Object, Package, Void)*
  * Classes fonctionnelles basiques *(Math, String, StringBuffer, ...)*
  * Module de la Machine Virtuelle *(ClassLoader, Compiler, Runtime, SecurityManager, ...)*
  * Abstractions (*System, Thread, Process, Throwable, ...)*
  * Mapping des primitives en objet *(Boolean, Byte, Character, Double, Float, Integer, Long, Short)*
  * Exceptions principales *(Exception, RuntimeException)*
  * Sous-Exceptions *(NullPointerException, ClassCastException, NumberFormatException, ...)*
  * Erreur principale *(Error)*
  * Sous-Erreurs *(OutOfMemoryError, InternalError, UnknownError, ...)*
  Sous-packages:
  * reflect: Classes de manipulation génériques du noyau objet du langage.
  Elles permettent d'atteindre des ressources de classes *(Field, Method, Constructor, Modifier, ...)*

- *java.net*

  Rôle: Permettre les fonctionnalités usuelles afférentes au Réseau et a ses protocoles classiques.
  Classes majeures : *InetAddress, ServerSocket, Socket, URL,*
  Interfaces majeures : *ContentHandlerFactory, SocketImplFactory, SocketOptions,*
  Exceptions majeures : *BindException, ConnectException, ProtocolException, SocketException,...*
  Fonctionnalité:
  * Client TCP/UDP (*DatagramSocket*, *Socket, MulticastSocket, ...)*
  * Serveur *(ServerSocket)*
  * Manipulation de ressources *(URI, URL, HttpURLConnection, ...)*
  * Gestion de droit *(Authenticator, NetPermission, PasswordAuthentication, SocketPermission, ...)*
  * Encodage/décodage *(URLDecoder, URLEncoder)*
  * Gestion d'adresse IP dans différentes versions *(InetAddress, Inet4Address, Inet6Address)*

- *java.nio*

  Rôle: Mise à disposition de *Buffer* pour les entrées/sorties.
  Classes majeures : *Buffer*
  Fonctionnalité:
  * Buffer dédiés aux types des primitives *(ByteBuffer, CharBuffer, DoubleBuffer, FloatBuffer, ...)*
  Sous-packages:
  * *channels*: Connection vers des ressources de type entrée/sortie,
  permet aussi de définir des *Selector* (opération I/O non bloquante multiplexées)
  * *charset*: fourni des *Encoder* et *Decoder* pour les conversion de caractère byte/Unicode.
  * *channels.spi / charset.spi*: classes permettant l'**I**mplémentation par des **S**ervice **P**rovider.

CC 1999-2012
Droits réservés
http://creativecommons.org/licenses/by-nc-nd/2.0/fr/

- *java.rmi*

  Rôle: Permet l'invocation de méthodes distantes à travers le réseau.
  Classes majeures : *MarshalledObject, Naming, RMISecurityManager*
  Interfaces majeures : *Remote*
  Sous-packages: *activation, dgc, registry, server*
- *java.security*

  Rôle: Fourni les classes et interfaces propre au Framework de Sécurité du langage
  Classes majeures : *AccessControler, KeyFactory, MessageDigest, Permission, Signature, Signer*
  Interfaces majeures : *Certificate, Key, Principal, PrivateKey, PublicKey*
  Fonctionnalité: Vérification et manipulation diverses dans le cadre de sécurité de la VM.
  Sous-packages: *acl, cert, interfaces, spec*

- *java.sql*

  Rôle: Framework permettant d'accéder des données issues de *DataSource* (SGBD, …)
  Classes majeures : *Date, DriverManager, Time,*
  Interfaces majeures : *Connection, Driver, ResultSet, Statement*
  Fonctionnalité:
  * Manipulation de types SQL *(Blob, Clob, Struct)*
  * Fonctionnalité avancées SQL *(CallableStatement, PreparedStatement, ...).*
  * Méta-Informations *(DatabaseMetaData, ParameterMetaData, ResultSetMetaData, ...)*

- *java.text*

  Rôle: Manipulation de textes, dates, nombres et messages, indépendamment du format naturel.
  Classe majeures : *Bidi, BreakIterator, Collator, DateFormat, DecimalFormat, NumberFormat*
  Fonctionnalité:
  * permet d'effectuer les manipulations usuelles sur des chaînes de caractères.

- *java.util*

  Rôle: Fournir des classes et interfaces utilitaires riches.
  Classe majeures : *Calendar, Collections, Date, HashMap, Locale, Random, Stack, Timer, Vector*
  Interfaces majeures : *Collection, Comparator, EventListener, Iterator, Map, Observer, Set*
  Fonctionnalité:
  * Structures simples de données *(HashTable, Vector, Stack, StringTokenizer, (JDK 1.0))*
  * Enrichie par le Framework Collection *(Map, Set, List, Iterator, ...)*
  * Gestion d'évènements génériques *(EventListener, EventObject, ...)*
  * Internationalisation *(Locale, ResourceBundle, Properties, ...)*
  * Utilitaires pratiques *(Timer, Calendar, TimeZone, Random, ...)*
  Sous-packages: *jar, logging, pres, regex, zip*

CC 1999-2012
Droits réservés

http://creativecommons.org/licenses/by-nc-nd/2.0/fr/

- *javax:* Compléter le noyau des packages *java* par des eXtensions.

accessibility :	Gestion de fonctionnalité propres aux mécanismes d'assistance.
crypto :	Opérations de cryptographie.
imageio :	Manipulation avancée d'images.
naming, naming.ldap :	Gestion d'annuaire, localisation d'objets par nomenclature.
net, net.ssl :	Divers *Factory* de *socket* (client/serveur), protocole SSL.
print :	Gestion d'impression avancée
rmi, rmi.CORBA :	*PortableRemoteObject*, et *bridge* vers CORBA/IIOP.
security.auth, security.cert :	Authentification, Autorisations, Certificats, …
sound.midi, sound.sampled :	Gestion de l'interface Midi, et de processus de sampling
sql :	*PooledConnection, DataSource, RowSet, XAConnection.*
swing :	Refonte du Framework de composants graphique selon MVC.
transaction, transaction.xa:	Exceptions spécifiques, Transactions distribuées.
xml.parsers, xml.transform:	Parsers SAX et DOM, transformations associées.

- *org.ietf:* librairies, publiées par l'*Internet Engineering Task Force,* orientées réseau et sécurité.

jgss :	Api unifiée d'accès à des services de sécurité.

- *org.omg:* librairies, publiées par l'*Object Management Group*, dédiées à CORBA.

CORBA:	Mapping de la technologie sur le langage Java.
CosNaming:	Service de *naming* pour les IDL Java.
Dynamic:	Objects spécifiques aux spécifications de l'OMG.
IOP:	Module IOP des spécifications de CORBA.
Messaging:	Module de messages de CORBA.
PortableInterceptor:	Gestion des flux de l'ORB.
PortableServer:	Classes et Interfaces communes (haut niveau).
stub.java.rmi:	Stub RMI-IIOP pour le package java.rmi.

- *org.xml:* librairies de parser XML.

sax :	Classes et interfaces du parser SAX.
sax.ext:	Extensions de classes et d'interfaces pour le parser SAX2.
sax.helpers:	Classes "helpers" pour les processus SAX.

–

- Exemples d'API spécifiques <u>externes</u> :
  - Commons Diverses librairies Apache (math, collections, beans, io, string,…)
  - Jasper: Modélisation avancée de l'impression.
  - Log4j: Gestion de logs.
  - JCap: Librairie Réseau
  - Spring : Famille de FrameWork : injection de dépendance, Web, sécurité, ..
  - IText: Publication PDF/RTF
  - JFreeChart Factory de diagrammes (chart, pie, …)
  - POI Generation de documents Excel
  - Hibernate Persistance d'objets en base de données.
  - ….

http://creativecommons.org/licenses/by-nc-nd/2.0/fr/

- Outils additionnels:

    - Test: JUnit, Cactus, DBUnit.
    - Déploiement: Ant, Maven.
    - Génération: Xdoclet

CC 1999-2012
Droits réservés

http://creativecommons.org/licenses/by-nc-nd/2.0/fr/

- Les API standards ou additionnelles fournies par Sun (source http://java.sun.com/products)

J2SE:	J2ME:
- Java Accessibility - Java Access Bridge - Java Plug-in Software - JavaBeans Architecture - Javadoc Tool - Java Foundation Classes (JFC/Swing) Technology - Java Platform Debugger Architecture (JPDA) - Java Plug-in Software for WindowsXP - Java 2D API - Java Web Start (JWS) Software - Java DataBase Connectivity (JDBC) Technology - Java Remote Method Invocation (Java RMI) - Core Java Technologies (Security) Extensions: - Java Advanced Imaging (JAI) API - Java Authentication and Authorization Service (JAAS) - Java Communications API (JCA) - Java Cryptography Extension (JCE) - Java Data Objects (JDO) - JavaHelp System - Java Management Extensions (JMX) - Java Media Framework (JMF) API - Java Naming and Directory Interface (JNDI) API - Java Secure Socket Extension (JSSE) Software - Java Speech API - Java 3D API J2EE : - Java BluePrints - ECperf Software - Enterprise JavaBeans Technology - Java API for XML Processing (JAXP) Software - Java API for XML Registries (JAXR) Software - Java API for XML-Based RPC (JAX-RPC) API - JavaMail API - Java Message Service (JMS) API - JavaServer Faces Technology - JavaServer Pages Technology - Java Servlet API - J2EE Client Provisioning Software - J2EE Connector Architecture - SOAP with Attachments API for Java (SAAJ) - Java Transactions	- Connected Limited Device Configuration (CLDC) - Mobile Information Device Profile (MIDP) - Sun Java Wireless Toolkit Software CLDC OPTIONAL PACKAGES - Java Technology for Wireless Industry (JSR-185) - Mobile Media API (JSR-135) - Location API for J2ME (JSR-179) - SIP API for J2ME (JSR-180) - Wireless Messaging API (JSR-120, JSR-205) - Security and Trust Services API for J2ME (JSR-177) - J2ME Web Services (JSR-172) - Mobile 3D Graphics API for J2ME (JSR-184) - J2ME Content Handler API (JSR-211) JAVA PARTNER SITE: - Java API for Bluetooth (JSR-82 Motorola) Connected Device Configuration (CDC, JSR-036, JSR-218) - Foundation Profile (FP, JSR-046) - Personal Basis Profile (PBP) - Personal Profile (PP) CDC OPTIONAL PACKAGES - J2ME RMI Optional Package - MobileMedia API (JSR-135) - Location API for J2ME (JSR-179) - Wireless Messaging API (JSR-120, JSR-205) - Security and Trust Services API for J2ME (JSR-177) - J2ME Web Services (JSR-172) Java Card Technology Java Embedded Server Technology Personal Java Technology Java TV API JavaPhone API Java Telematics Technology

CC 1999-2012
Droits réservés

http://creativecommons.org/licenses/by-nc-nd/2.0/fr/

Forums de discussion sur le langage Java:

http://forum.java.sun.com/index.jsp

General
- New To Java Technology
- Java Programming
- Java Programming [Archive]
- Adding Generics
- Advanced Language Topics
- Native Methods
- 100% Pure Java
- J2EE SDK
- Java Community Process Program
- Java 2 Software Development Kit
- Java BluePrints
- Java Game Development
- Java Desktop Application Development
- Java Errors and Error Handling
- UML, OO Design, Patterns
- Algorithms

Runtime Environment
- Java Runtime Environment
- Java Virtual Machine
- Java HotSpot

GUI Building
- Project Swing
- Project Swing [Archive]
- Abstract Window Toolkit (AWT)
- Java 3D
- JavaBeans
- Java Media Framework
- Java 2D
- Accessibility APIs
- Java Applet Development
- Java Event Handling

Essential Classes and Documentation Generation
- Java Collections Framework
- Internationalization
- Javadoc Tool
- JavaHelp
- Serialization

Majc
- New To MAJC
- Chip Level Multi Processing
- Wirespeed Computing

Distributed Computing
- Jini Network Technology
- The Brazil Project
- JDBC
- Enterprise JavaBeans
- JavaMail
- Interface Definition Language (IDL)
- Naming and Directory (JNDI)
- Remote Method Invocation (RMI)
- RMI-IIOP
- Serialization
- Java Transactions (JTA/JTS)
- Java Message Service (JMS)
- Distributed Computing General
- J2EE Patterns

Web Services and Applications
- Java Technologies for Web Services
- Java Technology & XML
- JavaServer Pages
- Java Servlet Technology
- JavaServer Faces Technology

Security
- Signed Applets
- Cryptography
- Security Managers
- Java Secure Socket Extension
- Security General

Installation and Compiling
- Compiling
- Installation

Debug and Deploy
- Java Archive (JAR) Files
- Java Extension Mechanism
- JDB Tool
- Java Plug-In
- JavaBeans ActiveX Bridge
- Java Web Start & JNLP

Sun Developer Network
- Discuss the Sun Developer Network Web Site
- New Forum Suggestions

Liens additionnels intéressants:

http://java.sun.com
http://www.ibm.alphaworks.com
http://www.javaworld.com
http://www.gamelan.com
http://developer.java.sun.com/developer/codesamples/examplets/index.html

CC 1999-2012
Droits réservés

http://creativecommons.org/licenses/by-nc-nd/2.0/fr/