de 1996 à 2003

à partir de 2003

Bases du langage Java

Applications de bureau

- Introduction à cette formation
  - Votre formateur … Et Vous
  - Le matériel et logiciels
    - L'IDE NetBeans plutôt que Eclipse
    - Gestion BD MySQL
  - L'organisation – horaires
    - Formation de 4 jours
  - La forme :
    - Un mélange de concepts avec application directe par un exemple simple
    - Des exercices
    - Un TP qui servira d'évaluation

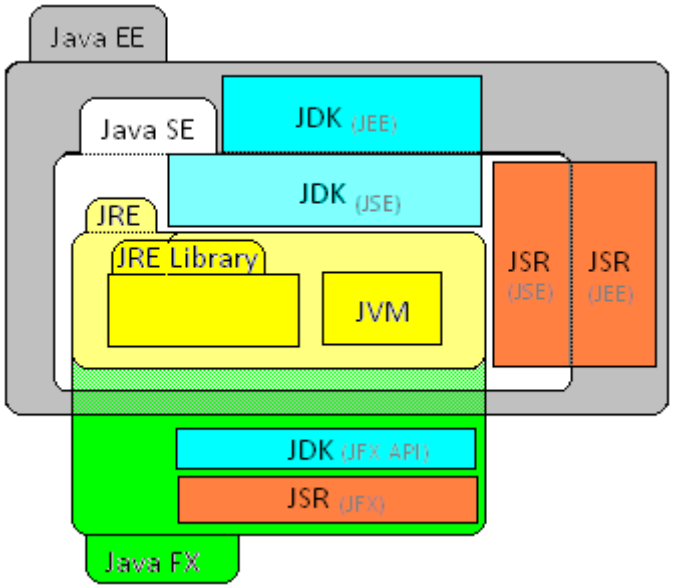
- Les liens utiles
  - https://www.w3schools.com/java/default.asp
  - https://www.jmdoudoux.fr/java/dej/chap-presentation.htm
  - d'autres liens pages suivantes

Sommaire

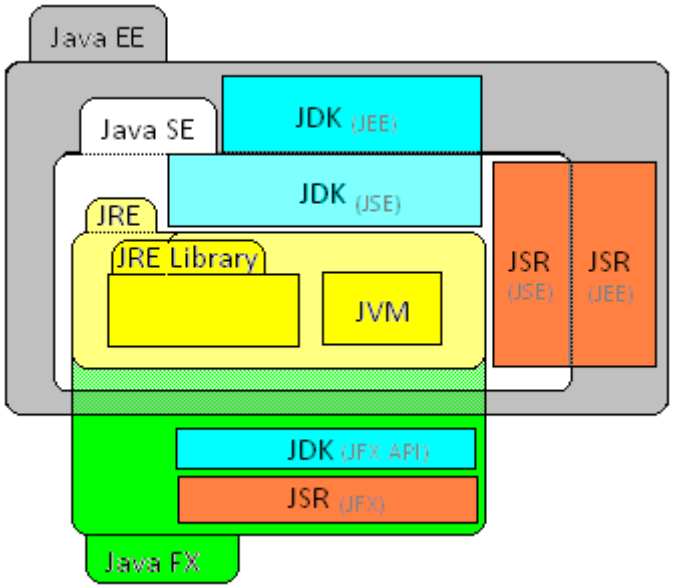
- L'environnement Java
  - Bref historique
  - Lexique autour des composants du langage
  - Les bases du langage
  - Conception Objet
  - Les annotations
  - Interfaces graphiques avec Swing
  - Persistance des données

Introduction

- Java est un langage de programmation
- Java SE (Java Standard Edition, ancien J2SE) est une plate-forme (framework) : contient de nombreuses bibliothèques pour réaliser des applications autonomes applications de bureau et serveurs

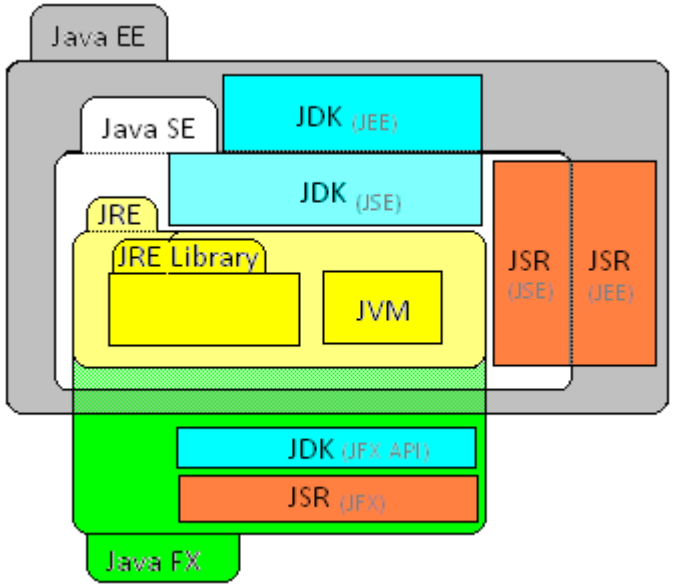


- Le JRE (Java Runtime Environment) contient l'environnement d'exécution des programmes JAVA
- Le JDK (Java Development Kit) utilisé pour les développeurs d'applications de bureau. Contient le JRE.
- Le JSR (Java Specification Request) : forme sous laquelle il faut proposer toute évolution de Java (Documentation)
- La JVM (Java Virtual Machine) : interpréteur de Byte Code



Introduction

- Java EE (Java Enterprise Edition, ancien J2EE) est une extension de Java SE pour le développement d'applications Web.



- Java ME Embedded pour des application Java Embarquées

Introduction

- Java et Javascript n'ont rien à voir
  - Un script Java est un bout de code en Java

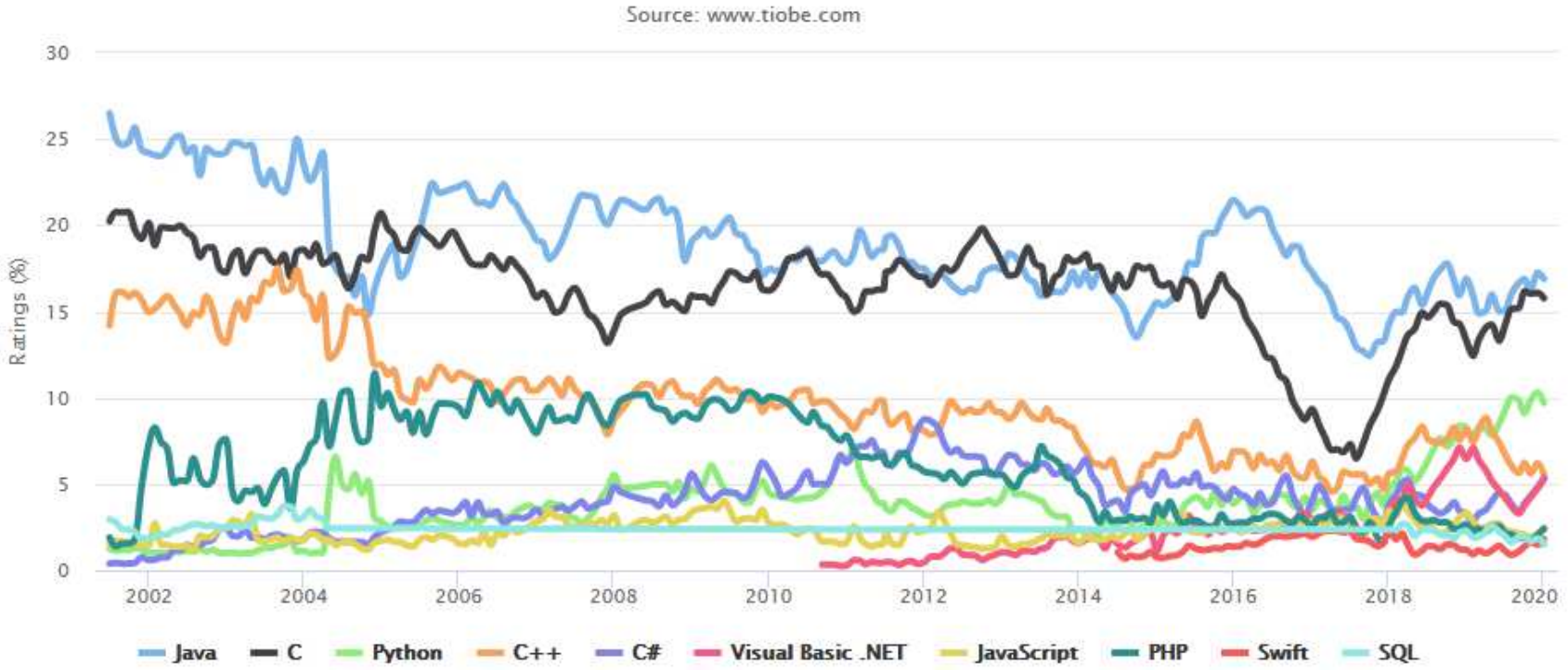
- Java EE, propriété d' Oracle a été cédé à la fondation Eclipse. Java EE se nomme Jakarta EE

- Java EE ou PHP pour un site Web ?

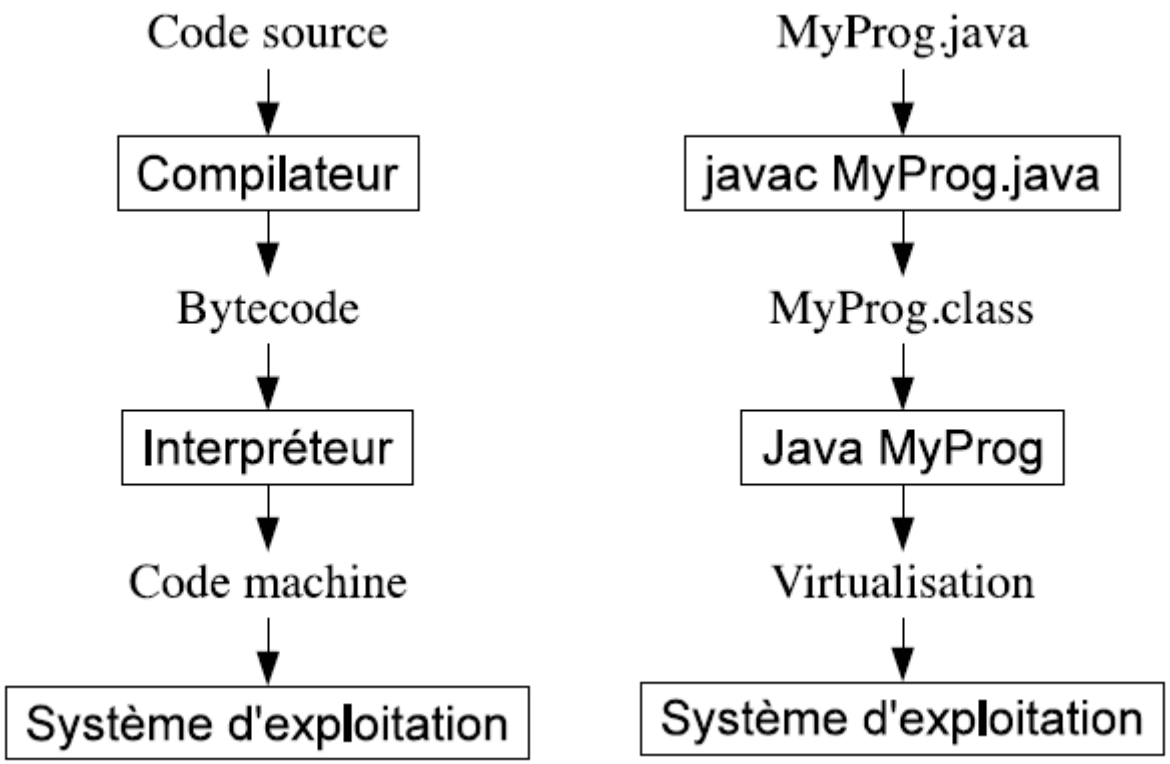
  https://blog.axopen.com/2014/06/java-vs-php-creation-dune-application-web-site-web-en-2014/

- La place de Java parmi les langages

  https://www.tiobe.com/tiobe-index/



- Le processus Java



Les bases

- Tout fichier source a pour extension .java
- Un fichier source peur contenir une ou plusieurs classes ou interfaces mais il ne peut y avoir qu'une seule classe ou interface déclarée publique par fichier
- Le nom du fichier doit correspondre à la classe ou interface publique qu'il contient
- La compilation d'un source .java produit du J-code ou byte code dans un fichier .class interprété par la machine virtuelle JVM lors de l'exécution
- La langage Java est fortement typé.
- Le J-Code est multi-plateformes

Les bases

- Une application autonome (console, ou avec une ihm dite de bureau) doit comporter un point d'entrée connue de l'OS : un « main »
- Une des classes de l'application doit donc comporter la définition :
  `public static void main(String[] args)`
- Pour compiler un code source : javac
  - Récupérer le fichier `MonApplication.java`
  - Ouvrir une fenêtre commande à cet endroit
  - `Javac MonApplication.java`
  - Regarder les fichiers produits

- Pour déployer une application ; utiliser une archive .jar
  - `jar cfe MonApplication.jar MonApplication *.class`
  - Lancer l'application par
    `java –jar MonApplication.jar`
    Ou par double clic sur MonApplication.jar dans un explorateur de fichiers

Les bases - packages

- Les packages
  - Pour nos développements nous avons intérêt à organiser/partitionner/structurer les classes (donc les fichiers .java) par catégories, au sens des packages UML
  - En java à un package correspondant un sous répertoire pour le développement, et à un espace de noms : une série de mots séparés par un '.'
  - Ex java.lang javax.swing com.cesi.CoursJava

Les bases - packages

- Les principaux packages standards de Java 6 sont :

java.applet	Création d'applets
java.awt	Création d'interfaces graphiques avec AWT
java.io	Accès aux flux entrants et sortants
java.lang	Classes et interfaces fondamentales
java.math	Opérations mathématiques
java.net	Accès aux réseaux
java.nio	API NIO
java.rmi	API RMI (invocation de méthodes distantes)
java.security	Mise en oeuvre de fonctionnalités concernant la sécurité
java.sql	API JDBC (accès aux bases de données)
java.util	Utilitaires (collections, internationalisation, logging, expressions régulières,...).

Les bases - packages

- Les principaux packages d'extensions de Java 6 sont :

javax.crypto	Cryptographie
javax.jws	Services web
javax.management	API JMX
javax.naming	API JNDI (Accès aux annuaires)
javax.rmi	RMI-IIOP
javax.script	API Scripting
javax.security	Authentification et habilitations
javax.swing	API Swing pour le développement d'interfaces graphiques
javax.tools	Api pour l'accès à certains outils comme le compilateur par exemple
javax.xml.bind	API JAXB pour la mapping objet/XML
javax.xml.soap	Création de messages SOAP
javax.xml.stream	API StAX (traitement de documents XML)
javax.xml.transform	Transformation de documents XML
javax.xml.validation	Validation de documents XML
javax.xml.ws	API JAX-WS (service web)

Les principaux packages tiers de Java 6 sont :

org.omg.CORBA	Mise en oeuvre de CORBA
org.w3c.dom	Traitement de documents XML avec DOM
org.xml.sax	Traitement de documents XML avec SAX

- Quand nous décrivons notre package dans un source :
  - `package com.cesi.CoursJava;`
- Quand nous utilisons un package existant :
  - `import javax.swing.JFrame;`

Les bases- Le classpath

- Le classpath est une chaîne de caractères indiquant un ensemble de chemins séparés par ‘;’

- permet de préciser au compilateur et à la JVM où ils peuvent trouver les classes dont ils ont besoin pour la compilation et l'exécution d'une application

- echo %classpath% sur votre machine

Les bases du langage

- Java est un langage 100% objet.
- Toute donnée, type de base ou nos objets, hérite implicitement de la classe object
- Types de base :

Type	Désignation	Longueur	Valeurs
boolean	valeur logique : true ou false	1 bit	true ou false
byte	octet signé	8 bits	-128 à 127
short	entier court signé	16 bits	-32768 à 32767
char	caractère Unicode	16 bits	\u0000 à \uFFFF
int	entier signé	32 bits	-2147483648 à 2147483647
float	virgule flottante simple précision (IEEE754)	32 bits	1.401e-045 à 3.40282e+038
double	virgule flottante double précision (IEEE754)	64 bits	2.22507e-308 à 1.79769e+308
long	entier long	64 bits	-9223372036854775808 à 9223372036854775807

```
import java.math.BigDecimal;  pour une précision supérieure à double
```

- Valeur par défaut du langage

Type	Valeur par défaut
boolean	false
byte, short, int, long	0
float, double	0.0
char	\u000
classe	null

- Les structures de contrôle

```
int time = 22;
if (time < 10) {
  System.out.println("Good morning.");
} else if (time < 20) {
  System.out.println("Good day.");
} else {
  System.out.println("Good evening.");
}
// Outputs "Good evening."
```

```
int i = 0;
while (i < 5) {
  System.out.println(i);
  i++;
}
```

```
for (int i = 0; i < 5; i++) {
  System.out.println(i);
}
```

```
int i = 0;
do {
  System.out.println(i);
  i++;
}
while (i < 5);
```

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
for (String i : cars) {
  System.out.println(i);
}
```

```java
int day = 4;
switch (day) {
  case 6:
    System.out.println("Today is Saturday");
    break;
  case 7:
    System.out.println("Today is Sunday");
    break;
  default:
    System.out.println("Looking forward to the Weekend");
}
// Outputs "Looking forward to the Weekend"
```

- Les tableaux

```
int tableau[] = new int[50]; // déclaration et allocation

// OU

int[] tableau = new int[50];

// OU

int tab[];          // déclaration
tab = new int[50]; // allocation
```

```
int dim1[][] = new int[3][];
dim1[0]      = new int[4];
dim1[1]      = new int[9];
dim1[2]      = new int[2];
```

```
int tableau[] = {10, 20, 30, 40, 50};
```

- Les listes

```
ArrayList<String> cars = new ArrayList<String>();
cars.add("Volvo");
cars.add("BMW");
cars.add("Ford");
cars.add("Mazda");
for (String i : cars) {
  System.out.println(i);
```

- Les dictionnaires

```
// Create a HashMap object called capitalCities
HashMap<String, String> capitalCities = new HashMap<String, String>();

// Add keys and values (Country, City)
capitalCities.put("England", "London");
capitalCities.put("Germany", "Berlin");
capitalCities.put("Norway", "Oslo");
capitalCities.put("USA", "Washington DC");
System.out.println(capitalCities);
```

- Conversion vers/depuis une String
  - int val = Integer.parseInt(”109”);
  - float val = Float.parseFloat(”109.10”);

  - String st = String.valueOf(xxx);

POO

- Voir le résumé joint
- Implémente tous les mécanismes POO connus
- Visibilités :
  - public
  - private
  - protected
  - Si rien de mis, la valeur par défaut est ”package friendly”
    L’entité en question (classe, attribut, méthode) est alors de type public pour toutes les classes du même package

- Définir une constante : `final double pi=3.14;`

- Les accesseurs (ou mutators en anglais)

```
private int valeur = 13;

public int getValeur(){
    return(valeur);
}

public void setValeur(int val) {
    valeur = val;
}
```

- Ecriture automatique avec netBeans, essayer en particulier avec un booleen

Arguments variables

- Les arguments variables : permet de ne pas définir l'ensemble des paramètres d'une méthode

```java
public class TestVarargs {

  public static void main(String[] args) {
    System.out.println("valeur 1 = " + additionner(1,2,3));
    System.out.println("valeur 2 = " + additionner(2,5,6,8,10));
  }

  public static int additionner(int ... valeurs) {
    int total = 0;

    for (int val : valeurs) {
      total += val;
    }

    return total;
  }
}
```

Enumération

- Enumération

```
public enum FeuTricolore {
  VERT, ORANGE, ROUGE
};
```

```
FeuTricolore feu = FeuTricolore.VERT;
```

les annotations

- Une annotation ajoute de l'information dans les méta-données
- Les méta-données sont utilisées à la compilation et/ou exécution
- Font partie du byte-code généré
- https://www.jmdoudoux.fr/java/dej/chap-annotations.htm
- Une annotation se place au dessus d'une classe ou méthode @<nom_annotation>

```
@Deprecated
public void maMethode() {
  System.out.println("test");
}
```

```
@Override
public void maMethode() {
}
```

Les Java beans

- https://www.jmdoudoux.fr/java/dej/chap-javabean.htm

- Les beans sont des objets autonomes pouvant être facilement assemblés et utilisés par des frameworks.

- La définition d'un bean se fait par la définition d'une classe :
  - Comportant un constructeur par défaut sans paramètre
  - Comportant les Propriétés que l'on souhaite exposer. Une Propriété est un attribut pour lequel on a ajouter des mutateurs (accessors) : une méthode get et une méthode set, is dans le cas d'un booléen.
  - Peuvent implémenter l'interface serializable
  - Peuvent émettre des événements en gérant une liste d'écouteur : mise en place du design Observer

Les Java beans

- Les Propriétés
  - Simples

```
private int longueur;

public int getLongueur () {
  return longueur;
}
```

```
private int longueur ;

public void setLongueur (int longueur) {
  this.longueur = longueur;
}
```

  - Indexées

```
private float[] notes = new float[5];

public float getNotes (int i ) {
  return notes[i];
}

public void setNotes (int i, float notes) {
  this.notes[i] = notes;
}
```

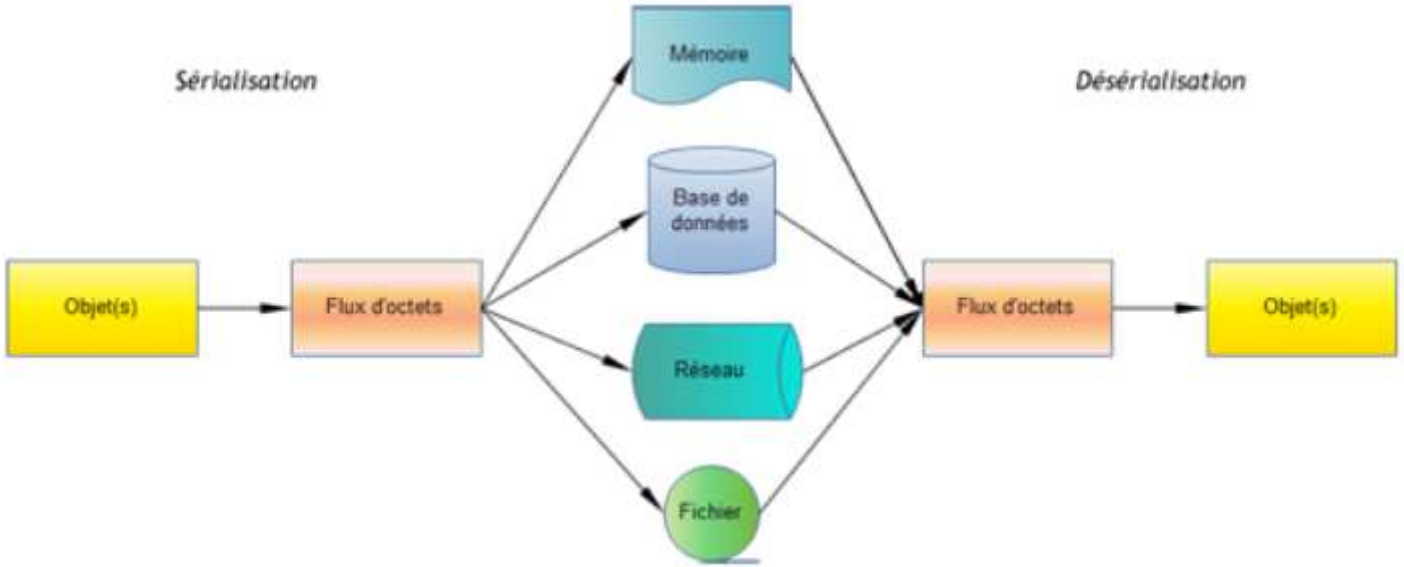
Les Java beans

- Les Propriétés liées
  - Les JavaBeans peuvent mettre en place un mécanisme qui permet pour une propriété d'enregistrer des composants qui seront informés du changement de la valeur de la propriété.
    => utilisation du design pattern Observer

  - Le process s'effectue suivant ces étapes :
    - Le bean doit implémenter l'interface Serializable
    - Le bean contient un objet PropertyChangeSupport
    - Le bean contient une méthode addPropertyChangeListener() pour ajouter un écouteur sur ce bean
    - Le bean contient une méthode removePropertyChangeListener() pour enlever un écouteur sur ce bean
    - Toute Propriété de type set appelle la méthode firePropertyChange() pour déclencher le process de réveil sur la modification de l'attribut

La sérialisation

- https://www.jmdoudoux.fr/java/dej/chap-serialisation.htm#serialisation-4
- Permet d'offrir une « persistance » des objets : les données de l'objet sont stockées sur disque ou transmises par réseau



La sérialisation

- Exemple sérialisation en XML
  - XMLEncoder
  - XMLDecoder

Exercice

- https://www.jmdoudoux.fr/java/dej/chap-swing.htm#swing
- Plusieurs bibliothèques existent pour le graphisme en java :
  - AWT obsolète
  - SWT (Eclipse foundation)
  - Swing
  - JavaFX(inspiré du web)

- Nous présentons ici Swing :
  - multi-plateformes

Graphisme en Java

- Exerçons nous sur une petite application graphique

Exercice

JavaDoc

- https://www.jmdoudoux.fr/java/dej/chap-javadoc.htm#javadoc

- Permet la documentation technique automatique à partir du code source

Le multi-tâches

- https://www.jmdoudoux.fr/java/dej/chap-threads.htm

Persistance des données

- A définir

BD et pattern DAO

- Dans la partie Modèle la couche Métier peut s'interfacer avec la couche Données

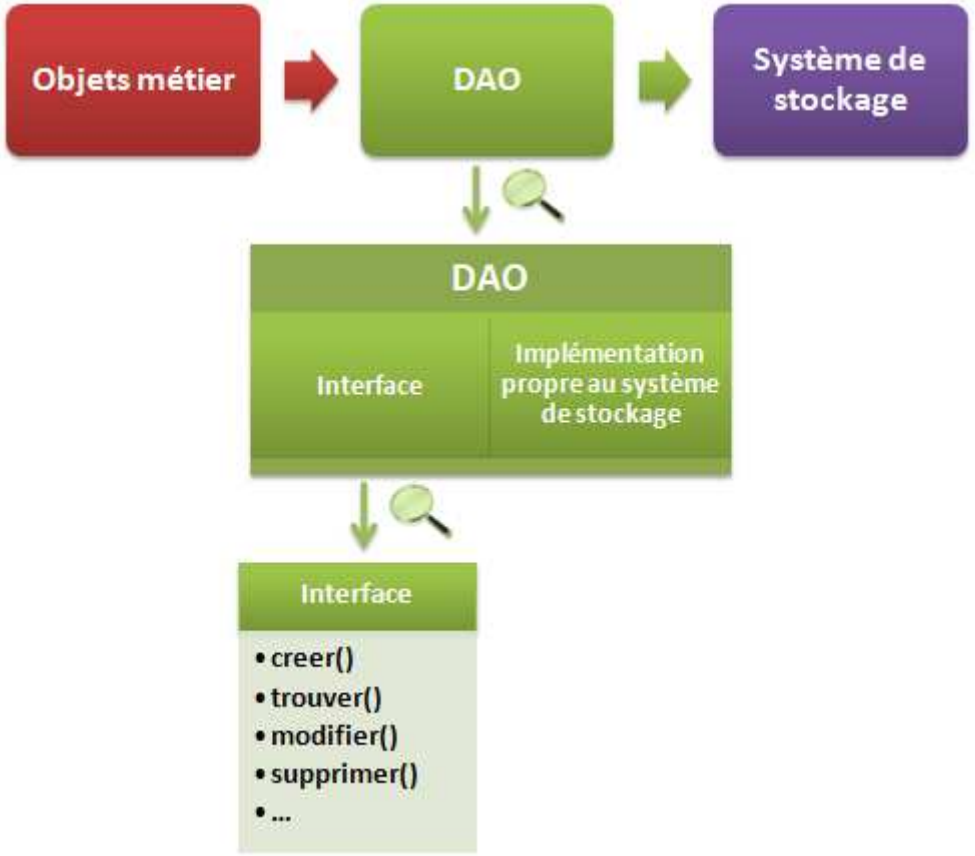
Mais :

  - Tests unitaires difficiles
  - Changer le support de stockage revient à modifier la couche métier

- sites openclassroom utiles :
  - https://openclassrooms.com/fr/courses/626954-creez-votre-application-web-avec-java-ee/624784-le-modele-dao
  - https://openclassrooms.com/fr/courses/26832-apprenez-a-programmer-en-java/26830-liez-vos-tables-avec-des-objets-java-le-pattern-dao

BD et pattern DAO

- Le pattern DAO :



BD et pattern DAO

Le pool de connexions BD :

- Problématique :
  - Toute connexion BD prend du temps
  - Non acceptable dans un environnement multi utilisateurs

- Solution : utiliser le « connection pooling »
  - Remplacer l'appel à DriverManager.getConnection() à une méthode pool.getConnection()
  - Connection.close() ne provoque plus la fermeture de la connexion mais rend la ressource au suivant.
  - L'interface « DataSource » décrit les méthodes de pooling.
  - BoneCP est une bibliothèque qui implémente cette interface.