Les objectifs de JAVA

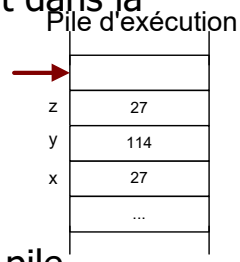
- Simple
  - syntaxe du langage C
- sûr
  - pas de pointeurs, vérification du code à l 'exécution et des accès réseau et/ou fichiers
- Orienté Objet
  - (et seulement !), pas de variables ni de fonctions globales, types primitifs et objet
- Robuste
  - ramasse miettes, fortement typé, gestion des exceptions
- Indépendant d'une architecture
  - Portabilité assurée par la présence d 'un interpréteur de bytecode sur chaque machine
- Environnement riche
  - Classes pour l'accès Internet
  - classes standard complètes
  - fonctions graphiques évoluées
- Support d'une méthodologie de conception basée sur les "Design Patterns"
  - Conception Orientée Objet

Simple : syntaxe apparentée C,C++

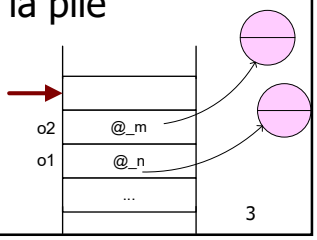
```
public class Num{
    public static int gcd( int n, int d){
        while( n != d)
            if (n > d)
                n = n - d;
            else
                d = d - n;
        return 0;
    }
}
```

Sûr par l 'absence de pointeurs accessibles au programmeur

- Deux types : primitif ou Object
- Variables locales de type primitif sont dans la pile
  - int x = 27;
  - int y = 114;
  - int z = x;

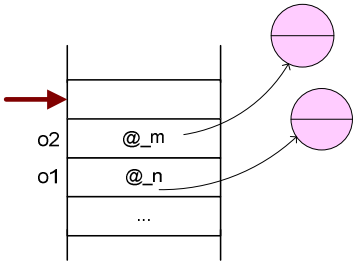


- référence locale d'objet sont dans la pile
  - //ref1 et ref2 références
  - Classe1 ref1 = new Classe1();
  - Classe1 ref2 = new Classe1();

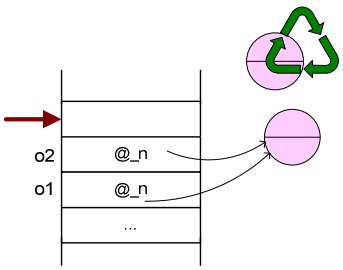


Garbage collector

- Classe1 o1 = new Classe1();
- Classe1 o2 = new Classe1();



- o2 = o1; //copie de référence

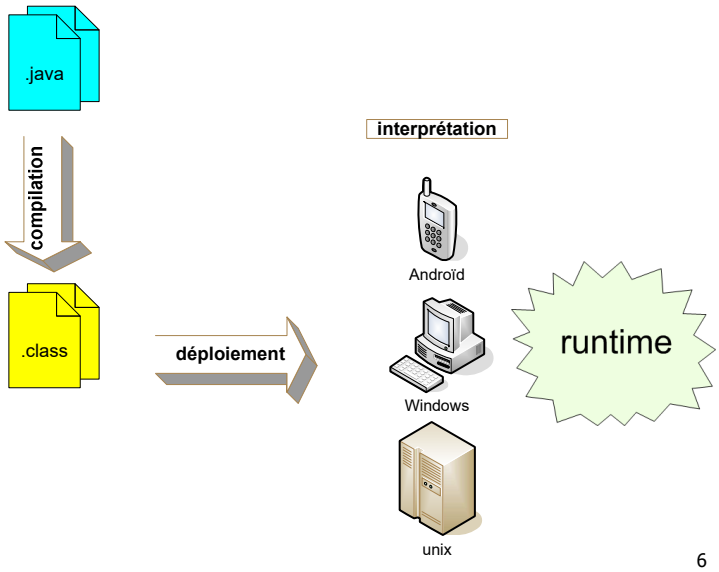


Le premier objet est « garbage collecté »

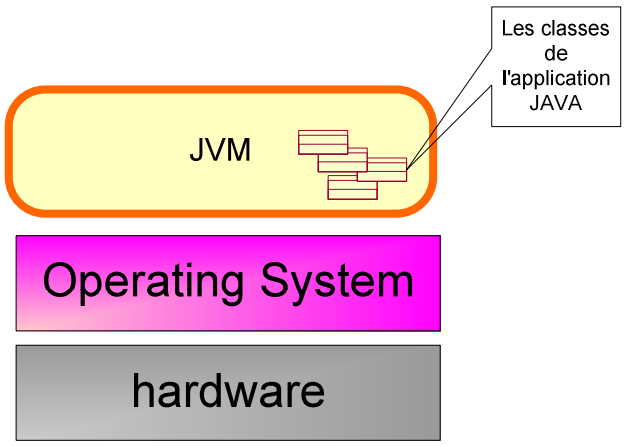
Robuste

- Ramasse miettes ou gestionnaire de la mémoire
  - Contrairement à l'allocation des objets, leur dé-allocation n'est pas à la charge du programmeur
    - (Ces dé-allocations interviennent selon la stratégie du gestionnaire)
- Fortement typé
  - Pas d'erreur à l'exécution due à une erreur de type
- Exceptions
  - Mécanisme de traitements des erreurs,
  - Une application ne devrait pas s'arrêter à la suite d'une erreur, (ou toutes les erreurs possibles devraient être prises en compte …)

Portable



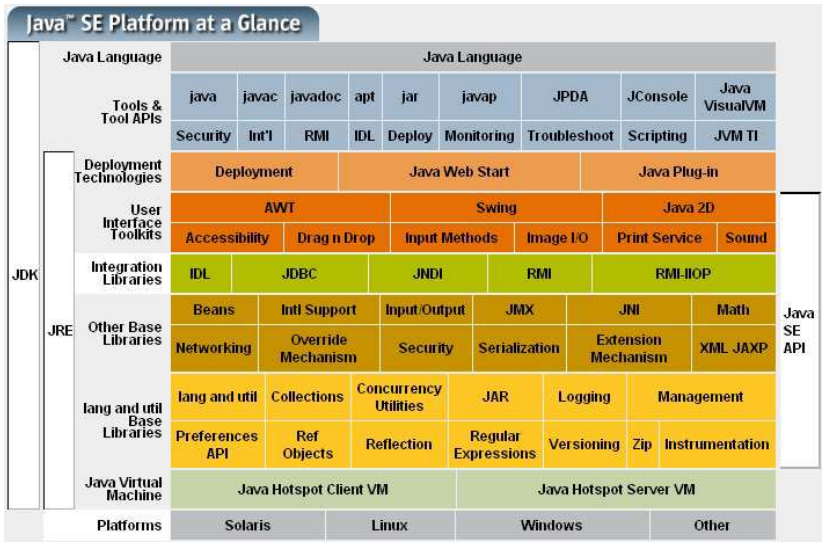
La pile logiciel au runtime



Acronyme

- JVM= Java Virtual Machine
- MV= Machine Virtuelle

API (1) très riche et gratuite



API (2)

- java.applet
- java.awt
- java.beans
- java.io
- java.lang
- java.math
- java.net
- java.rmi
- java.security
- java.sql
- java.text
- java.util
- javax.accessibility
- javax.swing
- org.omg.CORBA
- org.omg.CosNaming

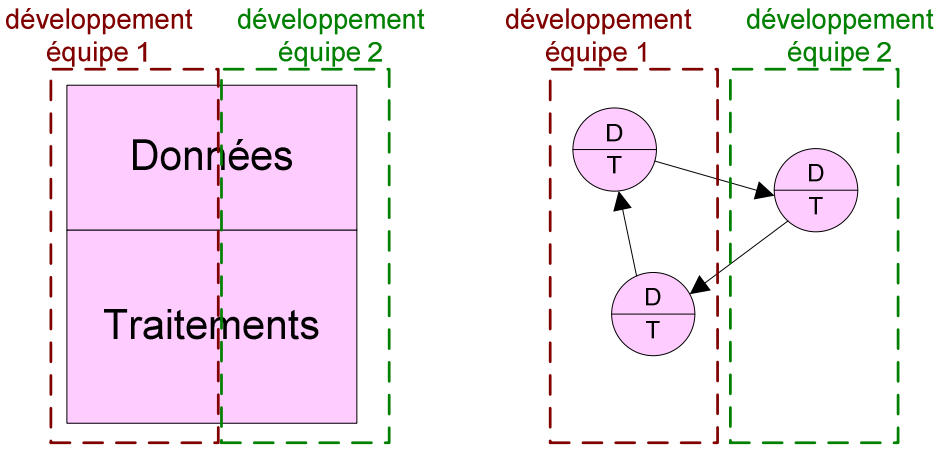
- Liste des principaux paquetages de la plate-forme JDK 1.2 soit environ 1500 classes !!! Et bien d 'autres A.P.I. JSDK, JINI, ...
- le JDK1.3/1850 classes, le JDK 1.4/2700 classes et bientôt le 1.5

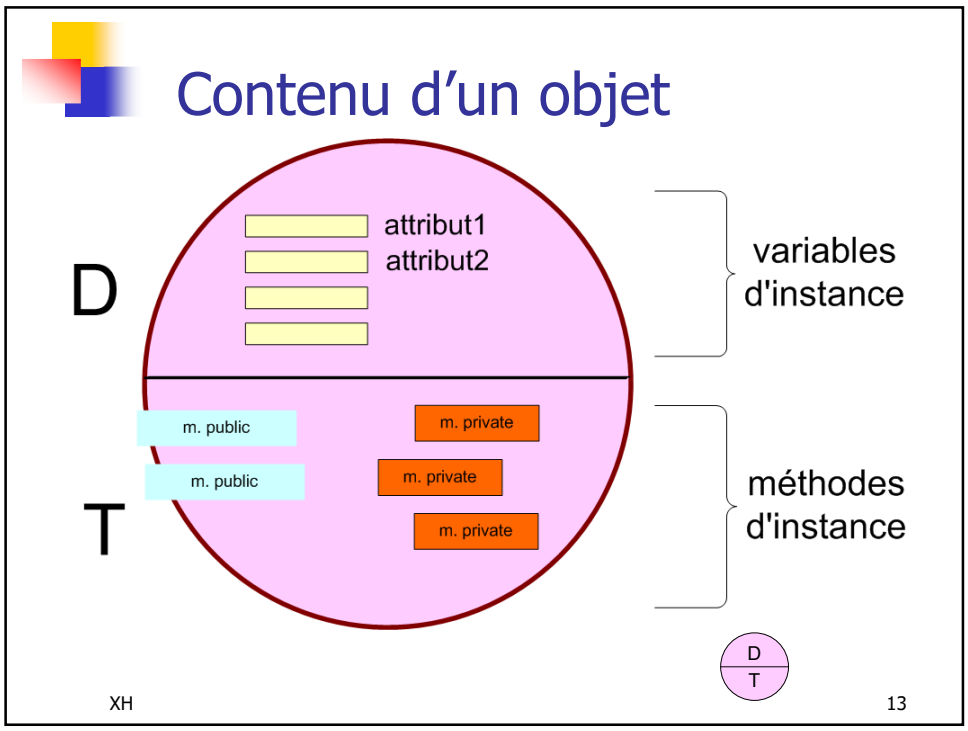
Design Pattern

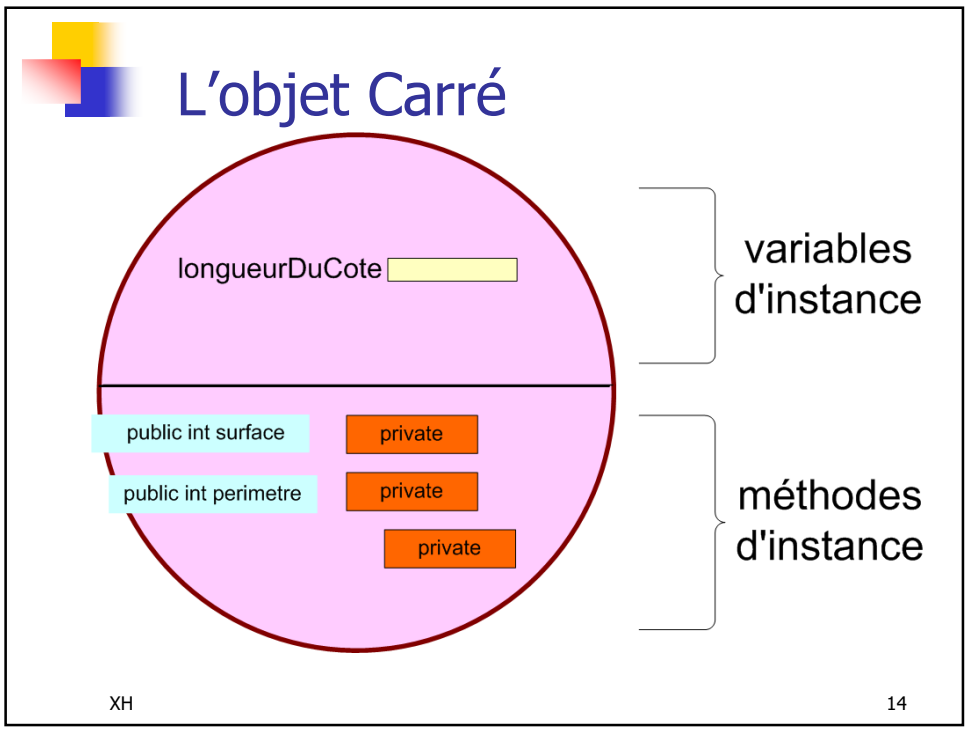
- Certains design pattern sont utilisés dans l'API Java
  - L'API AWT utilise le modèle composite ???
  - Les évènements de Java sont dérivés du Pattern Observateur ???
  - Etc…
- Design Pattern : Catalogue de modèles de conception réutilisables
  - Assemblage de classes pour un discours plus clair
  - Un modèle == plusieurs classes == Un nom de Pattern
  - Une application = un assemblage de pattern

Langage procédural versus langage objet

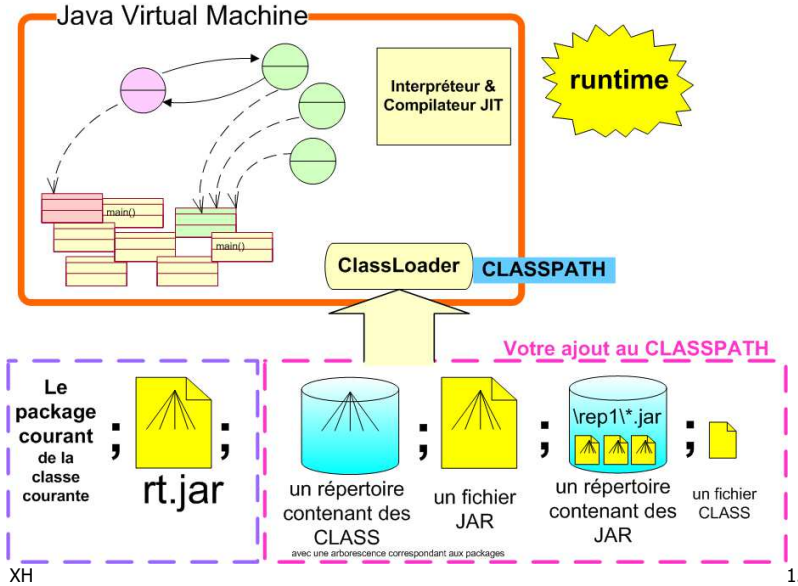
- à l'exécution !!!




Contenu d'un objet
D
attribut1
attribut2
variables
d'instance
T
m. public
m. public
m. private
m. private
m. private
méthodes
d'instance
D
T
XH
13


L'objet Carré
longueurDuCote
variables
d'instance
public int surface
public int perimetre
private
private
private
méthodes
d'instance
XH
14

La machine virtuelle Java à l'exécution



END