# Initiation à JAVA
# et
# à la programmation objet

raphael.bolze@ens-lyon.fr

# Objectifs

- Découvrir un langage de programmation objet.
- Découvrir l'environnement java
- Découvrir les concepts de la programmation orientée objet.
- Etre capable de développer une application en java.
- Ne pas se noyer !

# Plan

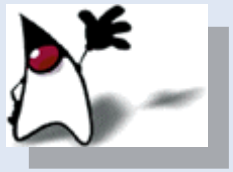
- Un peu d'histoire.
- Environnement java.
  - Comment ça marche concrètement
  - Installation de l'environnement de programmation
- Premier exemple
- Les outils autour de java
- Programmation Orientée Objet (POO)
  - Penser objet !
  - Vocabulaires
  - Notions essentielles
- L'histoire d'une application

# Un peu d'histoire

## Décembre 1990 :

Projet interne de Sun Micro System : Oak
Alternative au langage C++
Gestion de la mémoire
Portable
Machine embarquée

## Septembre 1992 :



Première démo sur un PDA  star7
Green OS, Oak langage, librairies, assistant Duke

## Juin-Juillet 1994 :

le projet prend son nom JAVA et se recentre sur le développement web.
Just Another Vague Acronym
Lien avec le café = java (en argos américain)
Les 4 premiers byte du bytecode java commence en base hexadécimale pour le nombre : 0xCAFEBABE

# Un peu d'histoire : les JDK de sun

* JDK 1.0 ( janvier 1996)

* JDK 1.1 ( décembre 1997)

* J2SE 1.2 (décembre 1998) — nom de code : Playground. =>  java 2

* J2SE 1.3 (mai 2000) — nom de code : Kestrel.

* J2SE 1.4 (février 2002) — nom de code : Merlin.

* J2SE 5.0 (septembre 2004) — nom de code : Tiger.

* Java SE 6 — nom de code : Mustang.

* Java SE 7 — nom de code : Dolphin.

Depuis 1998 : Java Community Process (JCP) qui utilise les Java Specification Request (JSR) pour spécifier les évolutions de la plate-forme JAVA ( JVM + librairies standards )

# Les différentes plate-forme

- J2SE : java 2 Standard Edition
- J2EE : java 2 Entreprise Edition
- J2ME : java 2 Mobile Edition

JVM + librairies standards

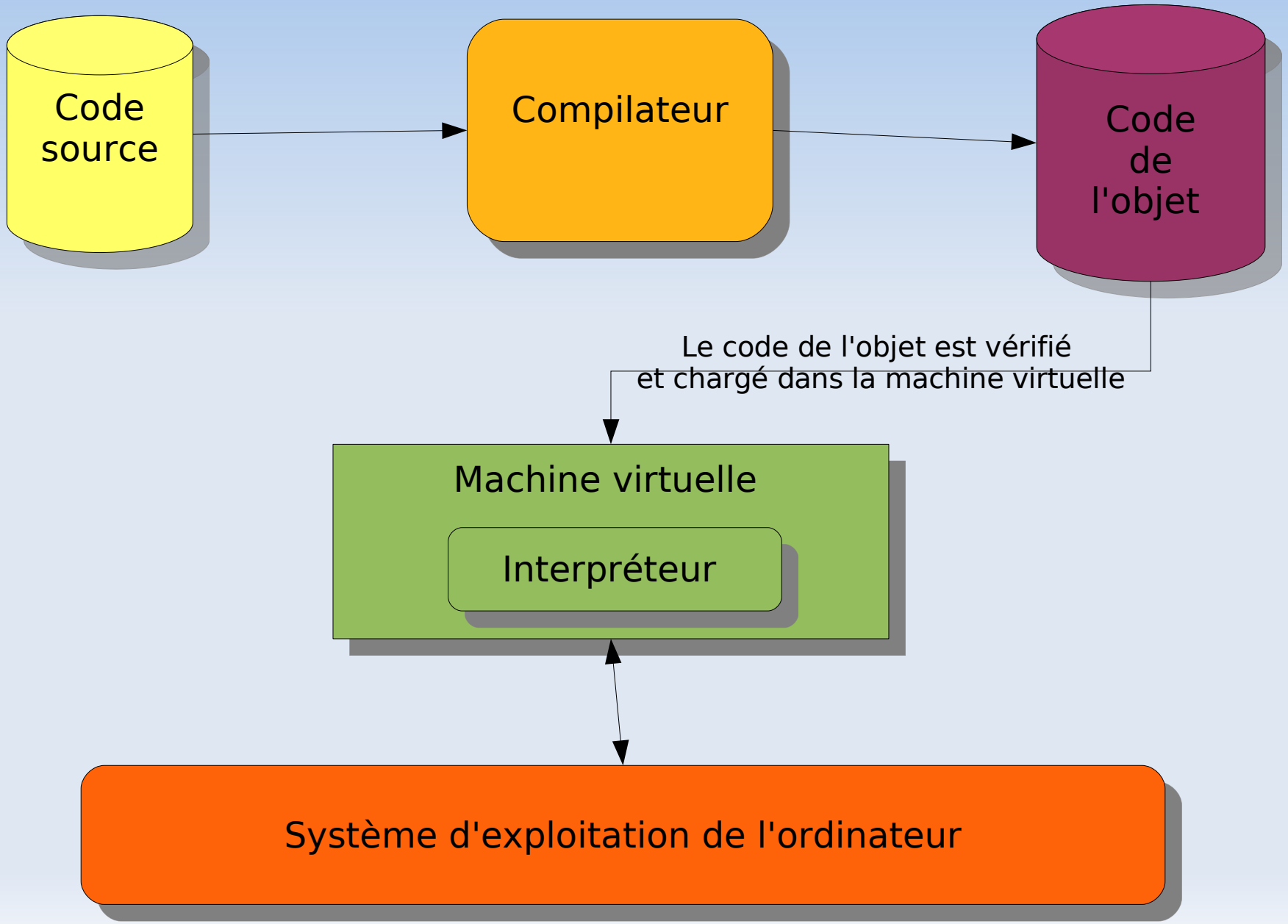
# Que faut-il pour faire du java

- **Un éditeur de texte :**
  - vi,emacs ...
- **Un “kit” de développement :**
  - SDK : Software Development Kit :
    - javac (compileur)
    - jar (archiveur)
    - javadoc (documentation)
    - jdb (débugueur)
    - java (machine virtuelle)

# Sun J2SE – Sun's current stable release (1.5.0).
# GNU Classpath – a free JDK replacement.
# IBM JDK – IBM's JDK.
# Blackdown – Free JVM/JDK for Linux.
# Apple Java – JVM/JDK for Mac OS X

  - Des outils pour compiler
    - ant
    - makefile
- **Des IDE (Integrated Development Environement):**
    - Netbeans, Eclipse

# La plate-forme JAVA.



# Premier programme

```java
/** Ma premier essai
 *  Affiche : “Hello Toto”
 * @author moi
 * @date juste_avant
 */
public class Toto {
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        System.out.println("Hello Xavier");
    }

}
```

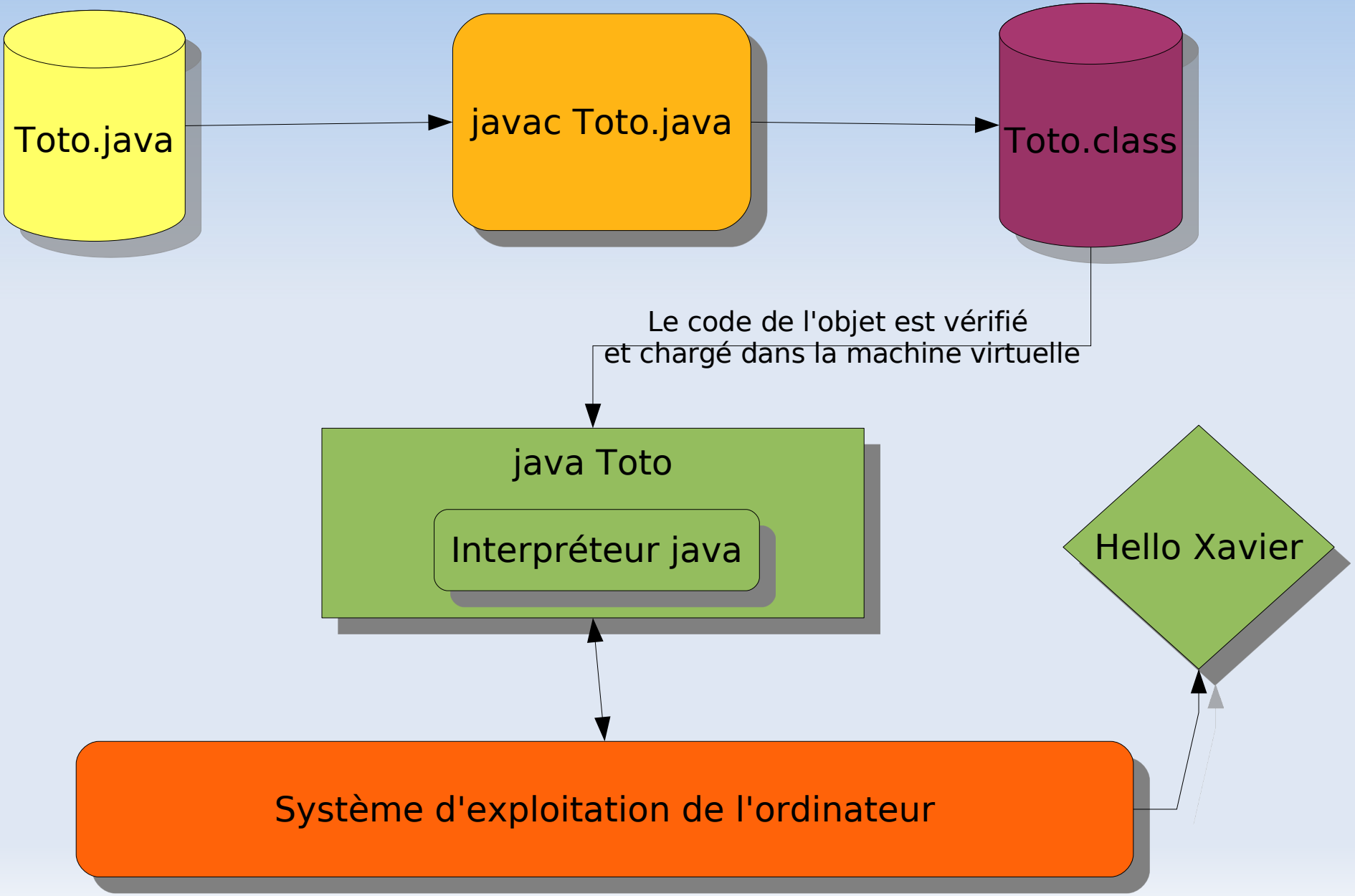
Compilation :

```
#>javac Toto.java
```

Exécution :

```
#>java Toto
Hello Xavier
```

# Concrétement : compiler exécuter



# Que faut-il pour faire du java

- **Un éditeur de texte :**
  - vi,emacs ...
- **Un “kit” de développement :**
  - SDK : Software Development Kit :
    - javac (compileur)
    - jar (archiveur)
    - javadoc (documentation)
    - jdb (débugueur)
    - java (machine virtuelle)

# Sun J2SE – Sun's current stable release (1.5.0).
# GNU Classpath – a free JDK replacement.
# IBM JDK – IBM's JDK.
# Blackdown – Free JVM/JDK for Linux.
# Apple Java – JVM/JDK for Mac OS X

  - Des outils pour compiler
    - ant
    - makefile
- **Des IDE (Integrated Development Environement):**
    - Netbeans, Eclipse

# La programmation objet

Les notions cléfs de la programmation objet :

- classes
- objets / instances
- attributs
- méthodes
- instanciation
- encapsulation
- héritage
- surcharge
- polymorphisme
- abstraction

# Programmation Orientée Objet

## Classes, objets, attributs, méthodes :

Tout est un objet.

Les objets contiennent :

- Des données : Attributs
- Des actions : Methodes

En Java, on définit la nature et le comportement d'un objet dans une classe.

# Programmation Orientée Objet : Instanciation

## Instanciation d'objet :

On instancie un objet en appliquant l'opérateur new sur un constructeur de classe.

## Allocation mémoire :

Pour qu'un objet puisse réellement exister au sein de la machine, il faut qu'il puisse stocker son état dans une zone de la mémoire. C'est le rôle de l'opérateur new qui détermine la taille de l'espace mémoire requis en appellant le constructeur.

Le constructeur crée l'objet et détermine l'état initial de l'objet. On appele ensuite les méthodes sur cette objet pour le faire vivre.

exemple ?

# Encapsulation

l’encapsulation est l'idée de cacher l'information contenue dans un objet et de ne proposer que des méthodes de manipulation de cet objet.

- public : les attributs dits public sont accessibles à tous.
- private : les attributs privés sont accessibles seulement par l'objet lui-méme.

# exemple : Instanciation

```
public class Test {
    // attribut de classe
    private int nb=5;
    public static alpha=10
    //constructeur
    public Test() {
        System.out.println("Contructeur 1");
        System.out.println("number="+nb);
    }
    public Test(int initial_nb) {
        System.out.println("Contructeur 2");
        System.out.println("number="+nb);
        this.nb=initial_nb;
        System.out.println("number="+nb);
    }
    // methodes
    public void augmenteNB(){nb++;}
    public void augmenteNB(int newNb){nb+=nb;}
    public void doubleNB(){nb<<=1;}
    public String toString(){return "number="+nb+", alpha="+alpha;}
}
```

```
public class ExempleInstanciation {
 // point d'entee
 public static void main(String[] args)
{
      Test t1 = new Test();
      Test t2 = new Test(3);
      t1.doubleNB();
      t2.doubleNB();
      Test.alpha++;
      System.out.println(t1.toString());
      System.out.println(t2.toString());
 }
}
```

# Un peu de vocabulaire

- Package
- Classe
- Variable de classe
- Variable d'instance/objet
- Constructeur/ Destructeur
- Méthode de classe
- Méthode d'instance/objet

```
package MonPaquet

class MaClasse {
    // Propriétés de la classe
      static Property c_propl
      Property i_propl
    // Constructeur(s)
      MaClasse(){}
    // destructeur
      public finalize(){}
    // Méthodes
      static void method2(){}
      void method1(){}

}
```

# Quelques règles de bonne conduite

- Un fichier par classe;
- Le fichier et la classe ont le même nom;
- les noms de classes et de packetages commence par des majuscules;
- les variables commence par des minuscules;
- les classes sont regroupées dans des packages

# Programmation Orientée Objet : Ramasse-miette

## Le destructeur :

Il permet de définir les actions à faire lorsque l'objet est détruit.

## Le ramasse-miette : Garbage Collector

Le ramasse-miettes (ou GC [Garbage Collector]) se charge de repérer ces objets inutiles et de libérer cette mémoire inaccessible.

Pour les plus curieux, sachez qu'en fait le GC fonctionne en permanence dans un thread de faible priorité.

# Les packages (paquetage)

Regroupement de classes

Ce regroupement suit le placement des fichiers sur le système de fichier.

- un dossier = un package

Librairie standard

- exemples :
  - java.lang.String  java/lang/String.java

exemple ?

# Les Librairies : fichiers jar

Un fichier JAR (Java ARchive) est un fichier ZIP utilisé pour distribuer un ensemble de classes Java.

Les fichiers JAR sont créés et extraits à l'aide de la commande jar inclue dans le JDK.

Les fichiers JAR sont destinés à être exécutés comme des programmes indépendants, dont une des classes est la classe principale.(contient le main)

exemple ?

jar -tvf rt.jar java/lang/String.class

# Petit point

Ce que l'on vient de voir :

- création de son premier petit programme java
  Hello Xavier
- Définitions d'une classe et d'un objet
- Création d'une classe et instanciation de deux objets.
- Du vocabulaire
- compiler, exécuter, archiver, documenter son programme.

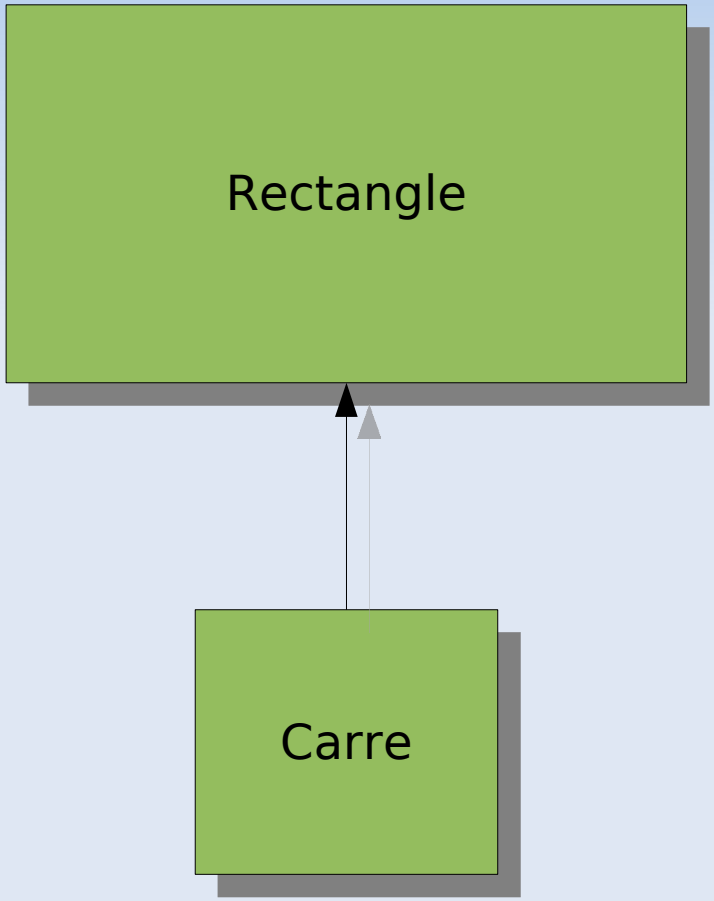
# Programmation Orientée Objet : héritage

## héritage

- L'idée principale consiste à définir une classe à partir d'une autre.
- La classe définie à partie d'une autre sera nommée classe fille.
- Celle qui sert à définir un classe fille sera nommée classe mère.
- On dit alors que la classe fille hérite (ou dérive) de la classe mère.
- Une fois que l'héritage est spécifié, la classe fille possède aussi l'ensemble des attributs et des méthodes de sa classe mère.

Remarque : une classe fille dérive d'une *unique* classe mère, l'héritage multiple n'étant pas supporté par le langage Java

# Programmation Orientée Objet : héritage




Classe mère

Classe fille

exemple ?

# Programmation Orientée Objet : héritage

```
package Geometry;
public class Rectangle {
    private static final String type ="Rectangle";
    private int largeur,longueur;
    /** Creates a new instance of Rectangle */
    public Rectangle(int largeur,int longueur) {
         System.out.println("Constructeur "+type);
        this.largeur=largeur;
        this.longueur=longueur;
    }
    public int perimetre(){return (largeur+longueur)*2;}
    public int surface(){return largeur*longueur;}
}
```

```
package Geometry;
public class Carre extends Rectangle {
    private static final String type ="Carré";
    /** Constructeur */
    public Carre(int cote) {
        super(cote,cote);
        System.out.println("Constructeur "+type);
    }
}
```

# Programmation orientée objet : héritage

## L'intérêt de l'héritage

L'héritage supprime, en grande partie, les réécriture du code.

Une fois la hiérarchie de classes bien établie, on localise en un point unique les sections de code.

On peut rajouter une classe à partir d'une classes parentes.

Si vous n'aviez pas encore modélisé un comportement dans une classe donnée, et que vous vouliez maintenant le rajouter, une fois l'opération terminée, ce comportement sera alors directement utilisable dans l'ensemble des sous-classes de celle considérée.

# Programmation Orientée Objet : héritage

Régles de l'héritage :

• Règle 1 : si vous invoquez super(), cela signifie que le constructeur en cours d'éxécution passe la main au constructeur de la classe parente pour commencer à initialiser les attributs définis dans cette dernière. Ensuite le premier constructeur continuera son exécution.

• Règle 2 : un appel de constructeur de la classe mère peut uniquement se faire qu'en première instruction d'une définition de constructeur. Une conséquence évidente est qu'on ne peut utiliser qu'un seul appel au constructeur de la classe mère.

• Règle 3 : si la première instruction d'un constructeur ne commence pas par le mot clé super le constructeur par défaut de la classe mère est appelé. Dit autrement, l'appel à super() est implicite. Dans ce cas, faites bien attention à ce qu'un contructeur à zéro paramètre soit définit au sein de la classe parente.

• Règle 4 : si vous invoquez this(), le constructeur considéré passe la main à un autre constructeur de la classe considérée. Encore une fois, cela doit être la première instruction du bloc.

exemple ?

# Programmation Orientée Objet : polymorphisme

## Polymorphisme

Un langage orienté objet est dit polymorphique, s'il offre la possibilité de pouvoir percevoir un objet en tant qu'instance de classes variées, selon les besoins.

Le langage Java est polymorphique.

exemple ?

# Programmation Orientée Objet : abstraction

Le mécanisme des classes abstraites permet de définir des comportements (méthodes) qui devront être implémentés dans les classes filles

Une classe abstraite ne peut pas être instanciée

# Assez parlé => Action

Un petit exemple de développement

# Quelques liens utiles :

Quelques pointeurs utiles

* Le site de Sun Microsystems :
  http://java.sun.com

* Les tutoriaux de Sun Microsystems (en anglais) :
  http://java.sun.com/docs/books/tutorial/

* Les spécifications des différentes versions de l'API Java :
  http://java.sun.com/products/jdk/1.1/docs/api/index.html
  http://java.sun.com/products/jdk/1.2/docs/api/index.html
  http://java.sun.com/products/jdk/1.3/docs/api/index.html
  http://java.sun.com/products/jdk/1.4/docs/api/index.html

* Cours en ligne :
  http://www.infini-fr.com/Sciences/Informatique/Langages/Imperatifs/Java/java.htm
  ...