Interface graphique

Responsable : Françoise Gayral

enseignant-chercheur au LIPN (Laboratoire d 'informatique de Paris-Nord) et à l 'IUT (département informatique)

LIPN : http://www.lipn.univ-paris13.fr/~gayral/

IUT : http://www-info.iutv.univ-paris13.fr/~gayral/

Mail : fg@lipn.univ-paris13.fr

Situation du cours d'interface graphique

Dans le PPN, partie algorithmique

- **Algorithmique en 1ère année**
  - algorithmique et programmation impérative : langage C
  - algorithmique et programmation orientée objet : langage Java

- Jusque là :
- Programmation non graphique et séquentielle

En 2 ème année : cours algorithmique en 2 parties

- **Développement d'interface graphique**
  (F. Gayral) : 9 semaines
  - principes généraux indépendants des langages d'implémentation
  - utilisation du langage Java comme support et utilisation de packages prédéfinis dédiés au graphique
- **Algorithmique avancée** : 11 semaines
  - structures de données complexes : piles, files, arbres, graphes...

Déroulement du cours

- **Organisation**
  - équipe : Françoise Gayral (responsable du cours), Dominique Bouthinon, Bouchaïb Khafif, Jean-Michel Barrachina
  - 1 cours d'1h15 obligatoire - 3h de TD/TP
- **Evaluation**
  - 1 contrôle court : coeff 1, sans documents : semaine 5 ou 6
  - 1 contrôle long : coeff 3, une partie sans documents, une partie avec documents : semaine du 12/12/05

- Cours Powerpoint avec photocopies distribuées en début de cours, à compléter et à enrichir durant le cours

Semaine 1	Introduction à la programmation d'interfaces graphiques utilisateur Java comme langage de création d'interface graphique Classe de base pour les fenêtres (avec swing) : premier canevas
Semaine 2	Les composants graphiques Les gestionnaires de mise en page
Semaine 3	Gestion des événements
Semaine 4	fin événement : les classes Adapter Composants particuliers : JList, dialogue avec l'utilisateur, ...
Semaine 5	Le graphisme : classe Graphics, redéfinition de paintComponent
Semaine 6	Java et le web : les applets java
Semaine 7	L'architecture Modèle-Vue-Contrôleur en général L'architecture MVC : un exemple : gestion d'un ensemble d'éléments
Semaine 8	L'architecture MVC : Gestion de données "tabulaires"
Semaine 9	MVC et interaction avec une base de données JDBC

Bibliographie

- Tutoriels de Sun :

  //java.sun.com/docs/books/tutorial/uiswing/

  //java.sun.com/developer/onlineTraining/GUI/Swing1/shortcourse.html#JFCIntro

  //java.sun.com/developer/onlineTraining/GUI/Swing2/shortcourse.html#JFCIntro

- Livres
  - java 1.2 : (s&sM), le programmeur
  - Au cœur de Java 2 (2 volumes), campus press
- Rechercher quelque chose de précis ou pointu:

  //java.sun.com/developer/JDCTechTips/

  //java.sun.com/developer/technicalArticles/

Généralités sur les interfaces graphiques

Bibliographie

- Tutoriels de Sun :

  //java.sun.com/docs/books/tutorial/uiswing/

  //java.sun.com/developer/onlineTraining/GUI/Swing1/shortcourse.html#JFCIntro

  //java.sun.com/developer/onlineTraining/GUI/Swing2/shortcourse.html#JFCIntro

- Livres
  - java 1.2 : (s&sM), le programmeur
  - Au cœur de Java 2 (2 volumes), campus press
- Rechercher quelque chose de précis ou pointu:

  //java.sun.com/developer/JDCTechTips/

  //java.sun.com/developer/technicalArticles/

Historique

- avant 1970 : système à cartes perforées
- années 1970 : introduction du clavier et de l'écran
- années 1980 : introduction de l'écran bitmap (point par point) et de la souris, système de fenêtres et de menus

  ==> IHM tels Smalltalk (76)

  Apple avec Macintosh (84), Windows

Changements par rapport à la programmation non graphique

- Tout programme s'exécute dans le cadre d'une fenêtre graphique
  - Mode graphique contre mode texte
  - Programmation événementielle

Fenêtre : élément fondamental d'une interface graphique

- Tout programme qui s'exécute le fait dans le cadre d'une ou plusieurs fenêtres
- Système de fenêtrage permet à plusieurs programmes d'utiliser le même écran graphique
  - A un instant donné, une seule fenêtre est active (voir la barre d'état).
- Fenêtre composée de composants graphiques (ou widgets)
- L'utilisateur communique ses ordres non plus seulement par le clavier, mais en agissant, par l'intermédiaire de la souris, sur les différents composants de le fenêtre

Mode graphique contre mode texte

- Mode texte utilisé jusqu'à présent :
  - informations apparaissent à l'écran uniquement sous forme de caractères
  - la couleur est rare
  - le principal organe de communication est le clavier.

  ==> Possibilités d 'affichage très limitées
- Mode graphique
  - Informations peuvent être affichées d'une multitude de manières différentes :
    - sous forme d'images, de dessins, de courbes ou graphiques, et bien sûr de caractères.
  - utilisation des couleurs (plusieurs millions)

  ==> Plus de convivialité

Affichage graphique

- Plus de mode texte, comme sous DOS où l'affichage était " séquentiel " : chaque information affichée ou lue s'inscrivait à la suite de la précédente.

- Tout programme sous IG s'exécute dans une fenêtre graphique

  ==> Prise en compte des notions de coordonnées, de fonte, d'épaisseur de traits, de couleurs...

  ==> Impossible d'utiliser les fonctions d'affichage standard

Java : System.out.println…
C : printf(…)

Programmation événementielle

- Sans IG ==> la programmation est **séquentielle**

```
public static void main(String[] args) {
    System.out.println("le programme est séquentiel");
    String nom=Console.readString("donner votre nom");
    System.out.println("bonjour "+nom);
    }
```

Le programme termine et rend la main

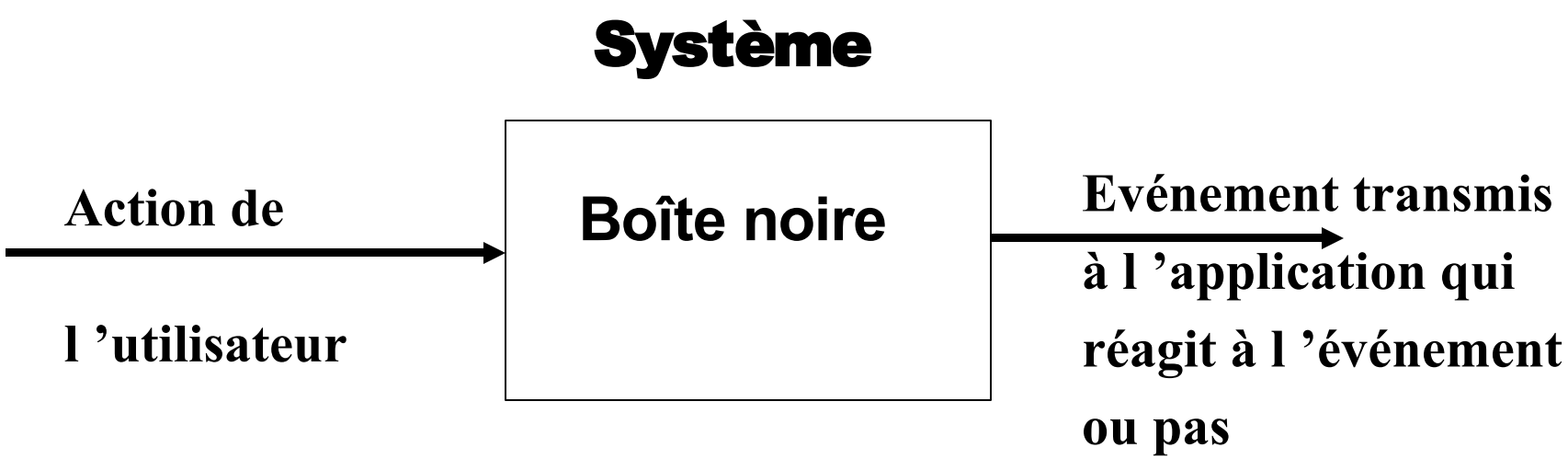
- C'est le programme qui garde le contrôle des opérations
- Les instructions du programme principal sont exécutées les unes après les autres
- L'utilisateur est simplement sollicité pour fournir des informations au moment voulu

Démo

Programmation événementielle

- Avec une IG ==> la programmation est **événementielle**
  - C'est l'utilisateur, par ses actions, qui contrôle les opérations en cliquant où il veut. C'est l'utilisateur qui "pilote" le programme
  - Le programme ne fait que réagir aux sollicitations de l'utilisateur (pourvu qu'elles aient été prévues)
- Dès que vous lancez un programme avec interface graphique :
  - une boucle infinie est générée
  - le programme attend les événements (enfoncer une touche du clavier, cliquer avec la souris etc.)
  - "attrape" et traite les événements quand ils se produisent.
- Le programme est dirigé par les événements

Comment ça fonctionne ?



- **Evénement** = Objet construit par le système en réponse à une action de l'utilisateur et qui contient toutes les informations concernant cette action

Comment ça fonctionne (détails) ?

- Action de l'utilisateur dans la fenêtre de l'application

  ==> Interruption matérielle

  ==> Récupération de l'interruption par le système

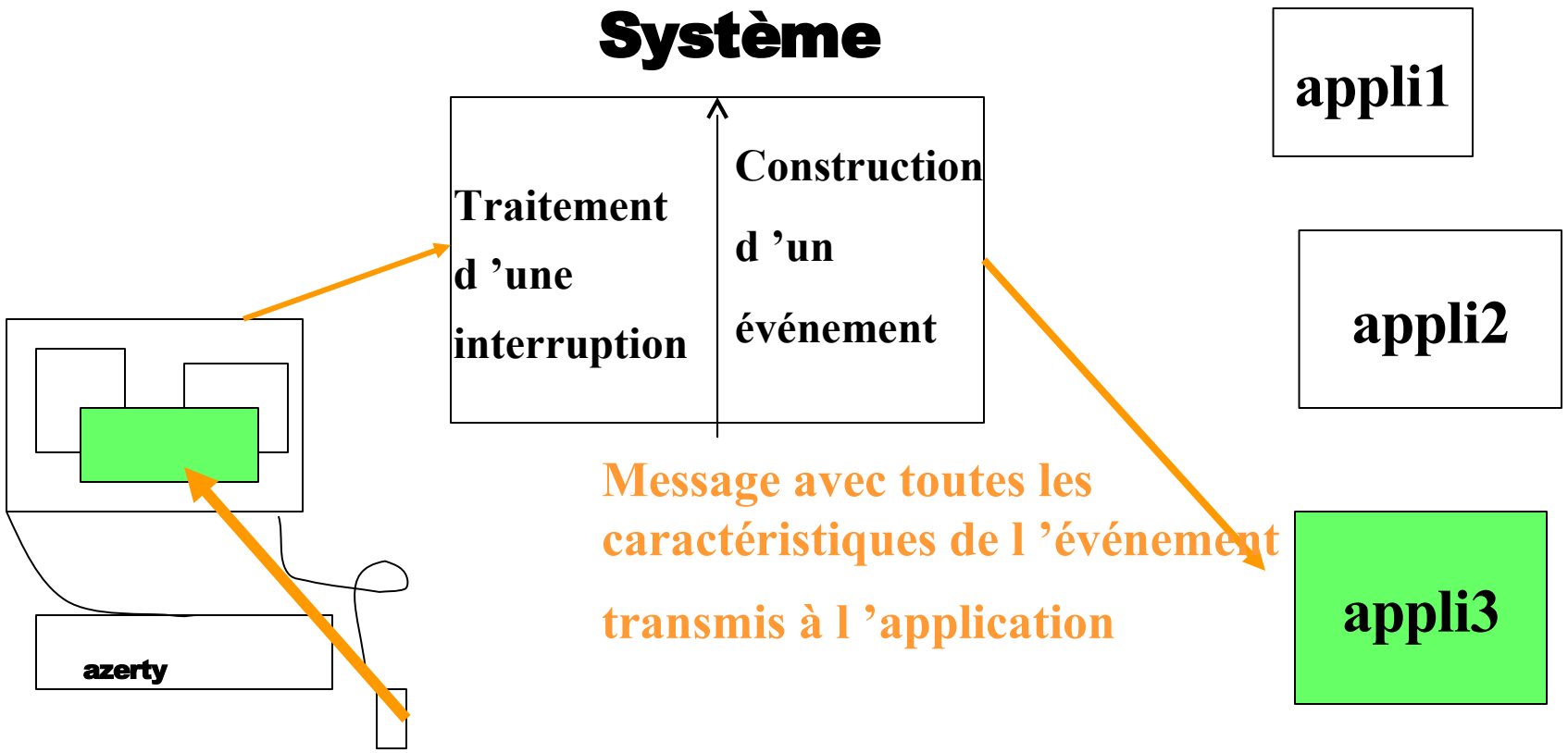
  ==> Construction de l'événement correspondant

  <== Envoi d'une message à l'application concernée avec des informations sur l'événement

  <== L'application réagit à l'événement ou non

- Exemples :
  - clic souris : connaissances des coordonnées du point de clic
  - frappe d'un caractère : connaissance du caractère frappé
  - événement de fenêtre :

Schéma général



Concevoir une application munie d'une interface graphique

- Bien séparer ce qui relève de l'application de ce qui relève de l 'interface graphique

- Exemple d'une banque :
  - applicatif : gérer les clients, les comptes, les transferts,….
  - GUI (graphical user interface) : comment l'applicatif apparaît à l'utilisateur
  - Un même applicatif peut avoir différentes GUI , suivant l 'utilisateur :
    - le guichetier, l 'administrateur,…

Concevoir une interface graphique, c'est:

- définir la fenêtre " cadre " de l'application et les objets graphiques qu'elle contient : menu, composants...

Aspect visuel

- définir les différentes actions que pourra réaliser l'utilisateur dans cette fenêtre (cliquer dans tel ou tel item du menu, dans tel composant : bouton, item d'une liste...)
- écrire le traitement correspondant à chacune des actions prévues.

Aspect événementiel

Remarques

- Toutes les actions de l'utilisateur qu'on veut traiter doivent avoir un traitement prévu
- Mais on ne connaît pas le moment où l 'action est déclenchée
- Le traitement prévu concerne le plus souvent l 'applicatif

API pour le graphique

- API = Application Programing Interface
  - Ensemble d'objets graphiques
  - Ensemble de fonctions permettant l'affichage, l'ouverture de fenêtre

- On peut utiliser :
  - directement l'API (programmation de bas niveau)
  - faire appel à des bibliothèques de haut niveau encapsulant ces fonctions, types...
  - Souvent langages objets avec des bibliothèques de classes organisées en hiérarchie

Différentes API

- XLib : bibliothèque de bas niveau en C utilisant le protocole X Window
- Gtk+ (Gimp tool kit) bibliothèque de haut niveau (widget), surcouche de Xlib
- Qt (société Trolltech) autre bibliothèque (C++). Qt a été choisi par Kde, environnement graphique de bureau Open Source pour les stations de travail Unix/Linux.
- Tcl/Tk
  - Tcl (Tool Command Language) langage de script (langage de programmation interprété)
  - Tk (Tool Kit) : la bibliothèque graphique de widget pour l'écriture de GUI, surcouche de Xlib
- Visual basic (pour windows), C++, bibliothèque java swing

Les éléments communs à ces API

- **Les widgets** : composants visuels de base généralement implémentés sous forme d'objet
  - posséde des attributs et méthodes par défaut et modifiables
  - définis et organisés selon une hiérarchie de classes fournie par l'API et, par l'héritage, il est possible de définir ses propres widgets dérivés d'une de ces classes

- **La gestion événementielle** par des déclenchements de fonctions ou méthodes

Les environnements de programmation

(IDE Integrated Development Environment)

- Integrated Development Environment (IDE) :

  a programming environment integrated into a software application that provides a GUI builder, a text or code editor, a compiler and/or interpreter and a debugger.
- en français : environnement de développement intégré
- Exemples :
  - Visual Studio (microsoft), Delphi, JBuilder (borland), KDevelop (license GPL) pour KDE, Eclipse**,...**
  - Supportent souvent différents langages de programmation

Environnements de développement pour Java

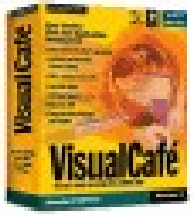
Nombreux IDE commerciaux

Forte for Java
SunSoft

VisualAge
IBM

JBuilder
Inprise

Visual J++
Microsoft



Visual Café
WebGain

...

Des environnements freeware ou shareware

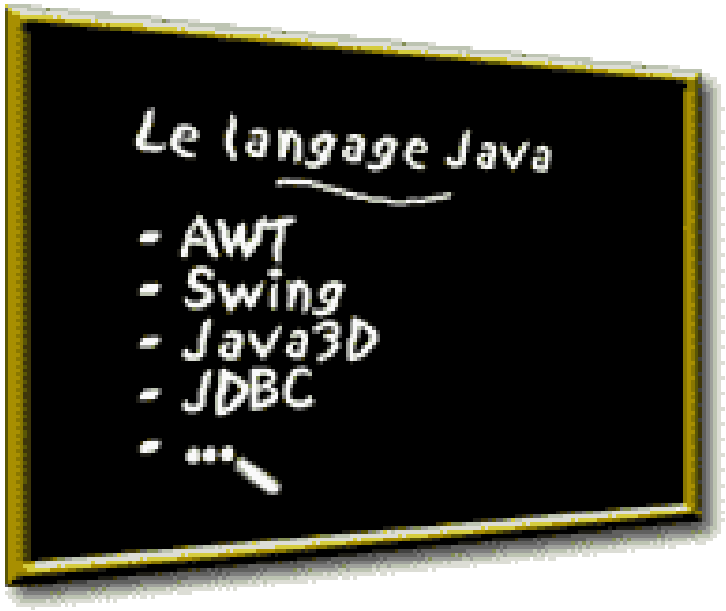
-- Kawa http://www.macromedia.com/software/kawa/

-- Emacs + JDE http://sunsite.auc.dk/jde

Dans ce cours

- Pas d'IDE
- environnement unix simple :
  - éditeur +
  - JRE (Java Runtime Environment) contient uniquement l'environnement d'exécution de programmes Java : javac, java, javadoc, jar…
- API : packages java (objets swing)

Interface graphique avec Java

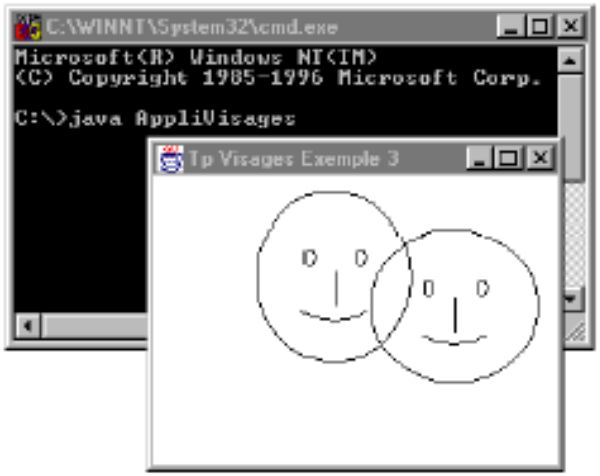


- Historique
- 2 types de programmes graphiques java
- La classe JFrame

Historique

- Java 1.0 en 1996, 1.1,... classes du package awt
- à partir de Java 1.2 (1998) : classes “ Java Foundation Classes ” (JFC)
  javax.swing, javax.swing.border, javax.swing.event,...
- Depuis sa version 1.2, Java a été renommé Java 2.
- Le JDK a été renommé J2SDK (Java 2 Software Development Kit)
  - 1.4 : 2002
  - 1.5 : 2004 désignée sous le nom J2SE (Java 2 Standard Edition)
  - Pour les différences J2SDK1.4 et J2SDK1.5, consulter
    http://perso.wanadoo.fr/jm.doudoux/java/tutorial/chap009.htm

2 types de programmes graphiques java



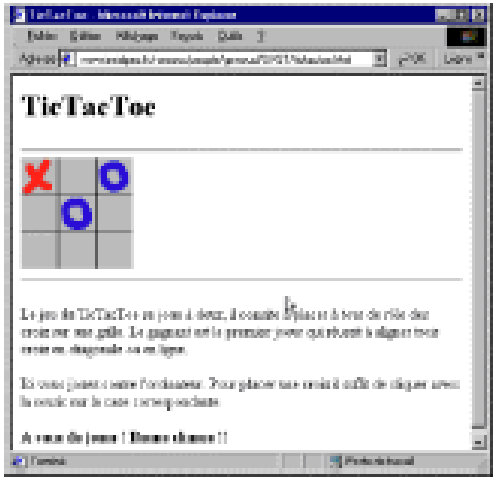
- **Les Applications indépendantes :**

– Programmes autonomes (stand-alone)

- **Les Applets**

– Programmes exécutés dans l'environnement d'un navigateur Web et chargés au travers de pages HTML

- **Différences :**
  - les contextes d'invocation et d'exécution
  - Les classes concernées



2 types de programmes graphiques java

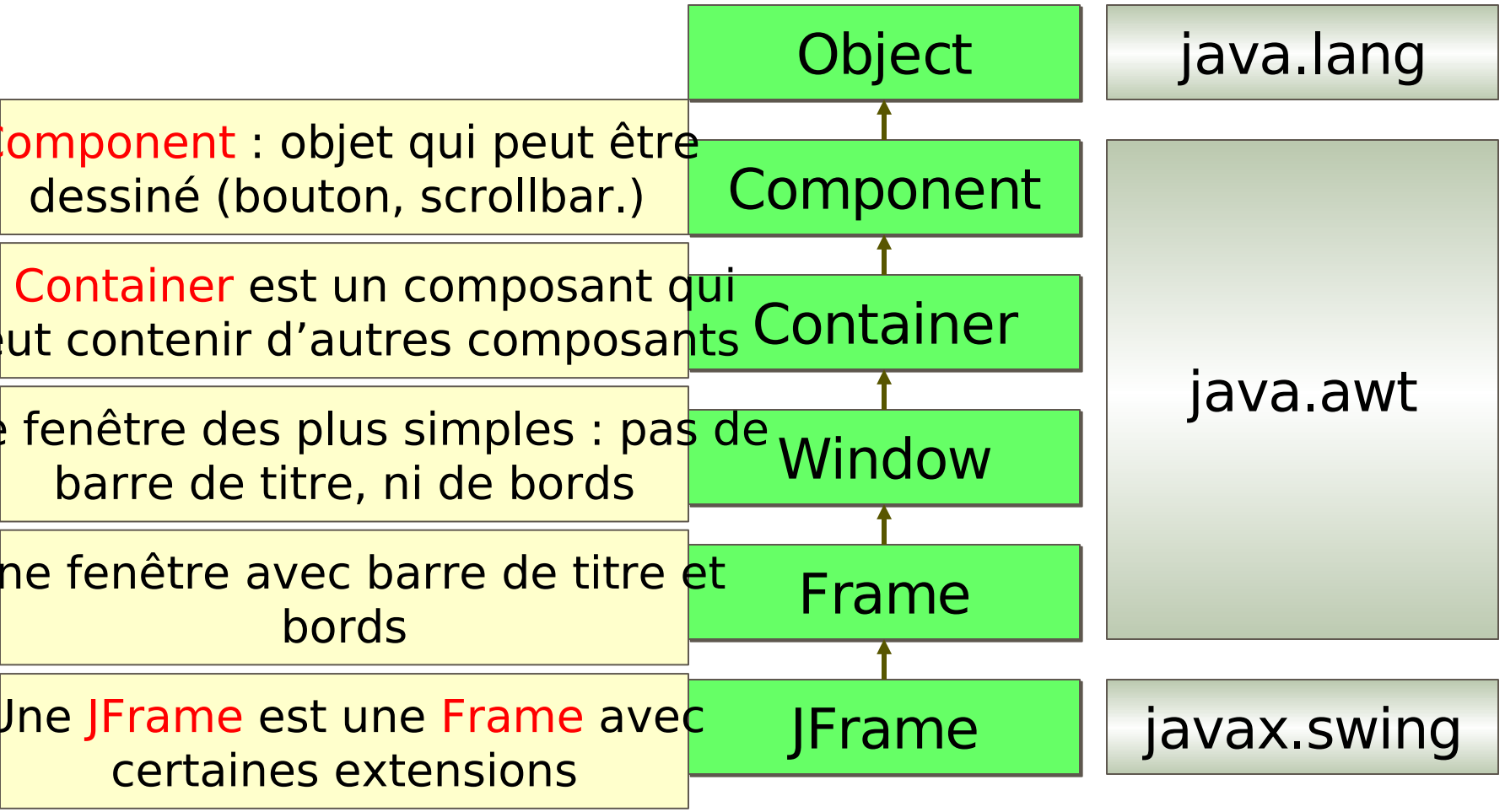
- Indépendante du web : sur la machine locale

  ⇒ classe javax.swing.JFrame

- Dans une application pour le web : applet placée sur un serveur et téléchargée sur le poste client

  ⇒ classe javax.swing.JApplet

JFrame et ses superclasses



Premier exemple : ouvrir une simple fenêtre

Démo

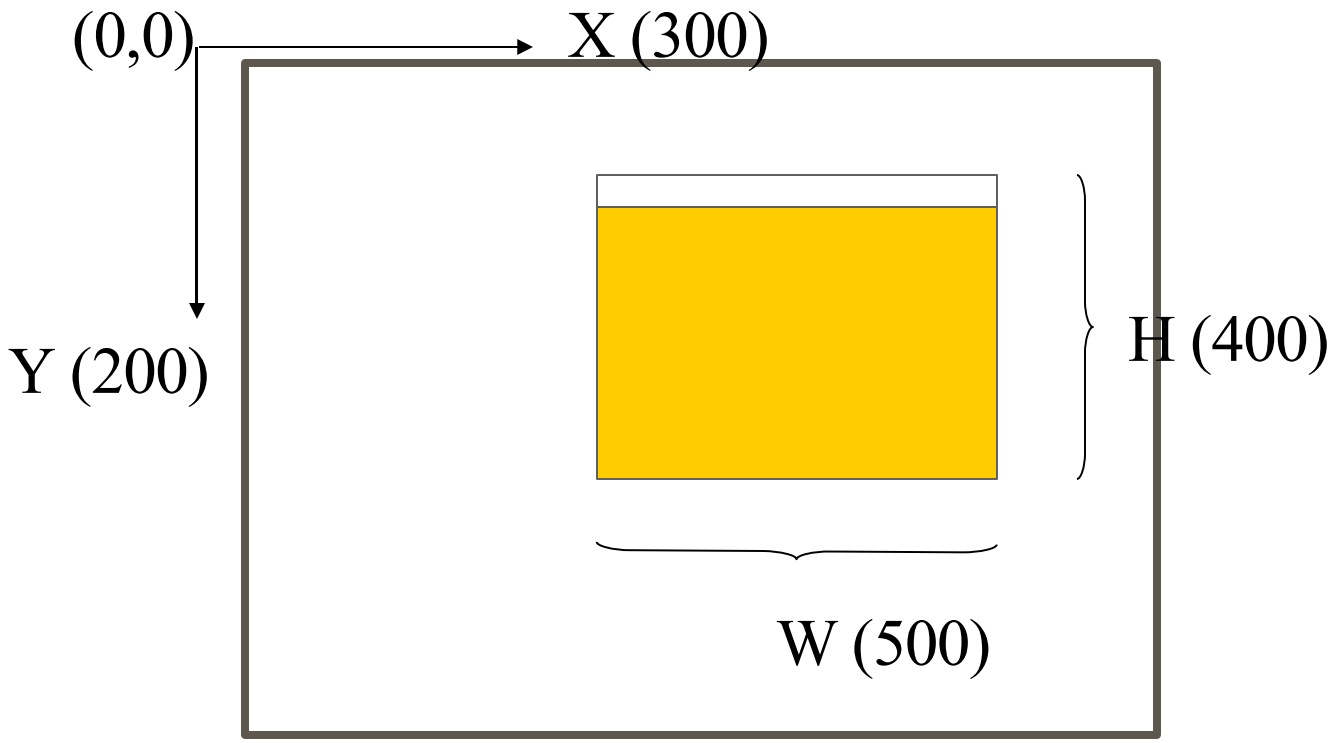
```
import javax.swing.*;
public class FenSimple extends JFrame {


public FenSimple (String titre, int x, int y, int w, int h)  {
        super(titre);
        this.setDefaultCloseOperation(EXIT_ON_CLOSE);
        this.setBounds(x,y,w,h);
        this.setVisible(true);
 }

public static void main(String[] args) {
        new FenSimple (  "Ma première fenêtre ",
300,200,500,400);
 }
}
```

Remarques sur le programme

- Ce programme se contente d'afficher une fenêtre a l'écran
- Une fois qu'arrive la dernière instruction de la fonction main() :
  - le programme ne s'arrête pas : il attend les événements
  - la fenêtre est toujours à l'écran
  - si l'utilisateur agit, rien ne se passe car rien n'a été prévu
  - le programme tourne tant que la fenêtre n'est pas fermée
- Fenêtre pas du tout paramétrée : position et dimensions "en dur"

Positionnement et taille

this.setBounds(300,200,500,400); //Position et taille



Détails sur les méthodes employées

- setVisible : Dessine la fenêtre au premier plan, par dessus toute fenêtre déjà visible à l'écran
- Les méthodes setBounds() et setVisible() sont héritées de Component, et sont donc disponibles pour tout Component
- setDefaultCloseOperation(EXIT_ON_CLOSE) permet de fermer la fenêtre quand l 'utilisateur cliquera sur la croix
- Autres méthodes :
  - setSize(int w, int h)
  - setLocation (int x, int y)
  - show() équivalente à setVisible(true)

Comment pallier l'indépendance de Java par rapport aux plate-formes ?

- Par exemple : si on veut connaître des informations sur la machine sur laquelle s 'exécute le programme ?
- La classe Toolkit
  - fait le lien entre les classes java indépendantes de toute plate-forme et la plate-forme sur laquelle tourne le programme
  - On obtient une instance de Toolkit par la méthode de classe de Toolkit : public static Toolkit getDefaultToolkit()
    - Exemple de code :

```
Toolkit aTK= Toolkit.getDefaultToolkit();
Dimension dim = aTK.getScreenSize();
```

et si on veut connaître des informations sur la machine sur laquelle s 'exécute le programme ?

- La classe System
  - permet d 'accéder à des informations liées au système ou à l 'utilisateur
  - Variables de classe connues
    - in, out, err (dans System.out)
  - Méthode importante
    - public static String getProperty(String key) : retourne la valeur de la propriété repérée par la clé key
    - Exemple de code :

```
System.getProperty("user.dir");
```