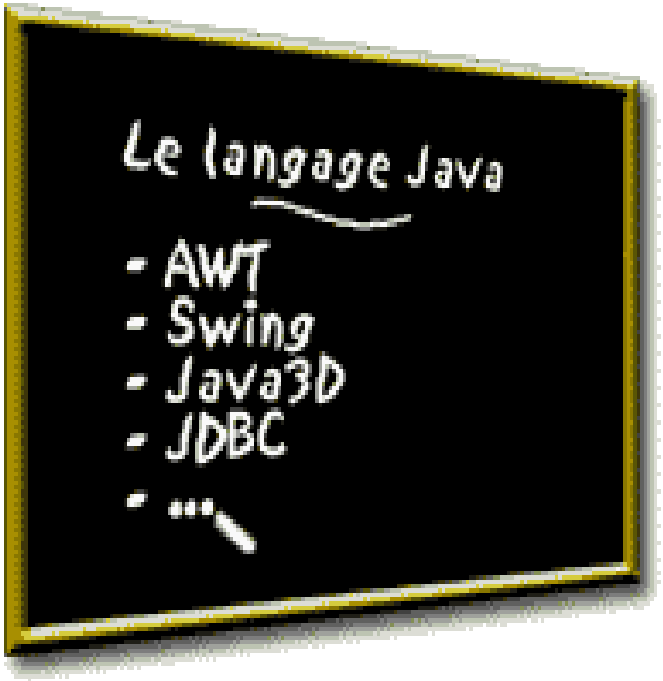
Cours 5 : Les applets java



Applet : programme java inclus dans une page HTML

Applet : technologie internet côté « client »

Architecture du web

- web : architecture client/serveur (requête/réponse)
- Le client, en général un navigateur, envoie à un serveur une requête d'accès à une ressource : programme CGI, fichier (HTML, image,…)
- Le serveur traite la requête et renvoie une réponse au client
- Requête et réponse sont des chaînes de caractères

Rappel : les standards du Web

- langage d'écriture des documents : HTML (HyperText Markup Language)
- adresses des documents sur Internet : URL (Unified Resource Locator)
- protocole de communication entre client et serveur : HTTP (HyperText Transport Protocol)
- typage des documents du Web : MIME (Multipurpose Internet Mail Extensions)
  - exemples : image/gif, text/html

HTML : un langage permettant de décrire des documents circulant sur le Web

- Un exemple : fichier index.html

```
<HTML>
 <HEAD>
 <TITLE> Exemple pour le cours </TITLE>
 </HEAD>
 <BODY>
   C'est ici qu'on &eacute;crit le texte de la page en faisant attention aux accents !!!
 <H1> Bonjour !   </H1>
   <IMG SRC="tomcat.gif">
 </BODY>
</HTML>
```

Communication client/serveur quand la ressource est un fichier html

- Le fichier index.html contient la ressource tomcat.gif
- Les deux fichiers (html et gif) sont présents sur le serveur

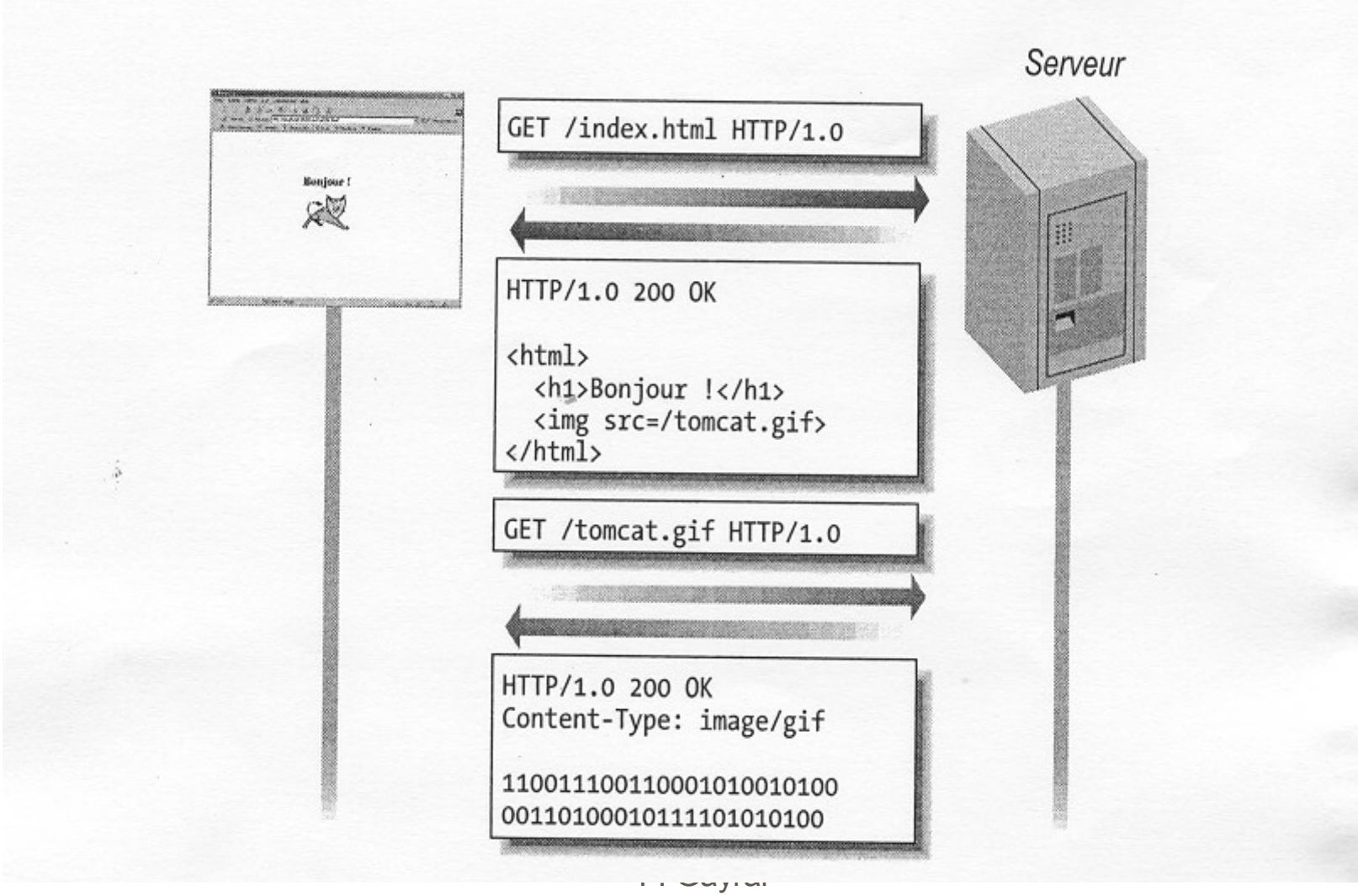
(ex : www.lipn-univ-paris13.fr)

- Le navigateur envoie la première requête

http: //www.lipn-univ-paris13.fr/index.html

- Pour chaque ressource identifiée dans le code HTML, le navigateur envoie une nouvelle requête au serveur
  - ici, une requête pour le fichier tomcat.gif

Communication entre le navigateur et le serveur



Page statiques/Pages dynamiques

- Fichier HTML « pur » : page statique au contenu toujours identique
- Si on veut des pages au contenu dynamique qui dépendent, par exemple, de résultats de programmes lancés sur le serveur comme l'interrogation d'une base de données, il existe d'autres techniques :
  - Embarquer du code dans le fichier HTML (balises encadrée de %)
    - Pages ASP (Active Server Page) : technologie microsoft
    - Pages JSP (Java Server Page) : technologie sun
  - Requête à une servlet (programme java côté serveur)
  - Requête à un script sur le serveur : programme php, perl,…

⇒ Voir le cours Web-BD

Et les applets ?

- Il existe une balise HTML permettant d'intégrer une applet dans une page HTML

```
<APPLET CODE= PremierApplet.class
                WIDTH=200 HEIGHT=100>
</ APPLET >
```

- PremierApplet.class est une ressource (programme java déjà compilé) qui se trouve sur le serveur
- PremierApplet.class correspond à la compilation d'un fichier définissant une sous-classe de la classe javax.swing.JApplet

Applet : technologie internet côté « client »

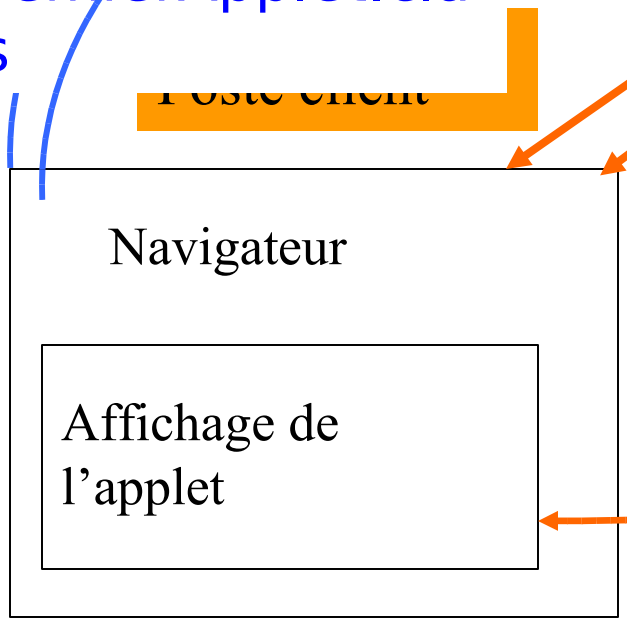
- Applet :
  - code java (.class) stocké sur un serveur web (comme des images ou des fichiers HTML)
  - téléchargé de ce serveur sur le poste client quand le client (un navigateur) demande la page HTML contenant la balise APPLET référant au code inscrit dans l'attribut CODE
  - exécuté ensuite sur le client de façon autonome par la machine virtuelle java incluse dans le navigateur

Schéma client/serveur

1. Requête du navigateur

1.a Requête de la page exemple.html faisant référence à l'applet PremierApplet.class stockée sur le serveur

1.b Requête de la ressource PremierApplet.class



Serveur

contenant des ressources (fichiers html, applet, image,…

2. Réponses du serveur

2.a. Transfert du fichier exemple.html

2.b Transfert du fichier PremierApplet.class

3. Exécution **en local** du fichier .class par l'interpréteur java et insertion de l'affichage de l'applet dans la fenêtre du document html

Comment ça se passe ?

- Le navigateur, quand il rencontre la balise APPLET,
  - vérifie qu'il n'a pas ce code déjà dans le répertoire cache de sa machine
  - Si ce n'est pas le cas, il envoie une nouvelle requête au serveur lui demandant le code indiqué de cette applet
- En réponse, le serveur retourne au client le programme java inscrit dans le paramètre code
  - Le code de l'applet est donc téléchargé du serveur dans le répertoire cache de l'ordinateur client
  - Puis exécuté par le chargeur d'applet sur la machine client

Pour spécifier le répertoire de l ’applet

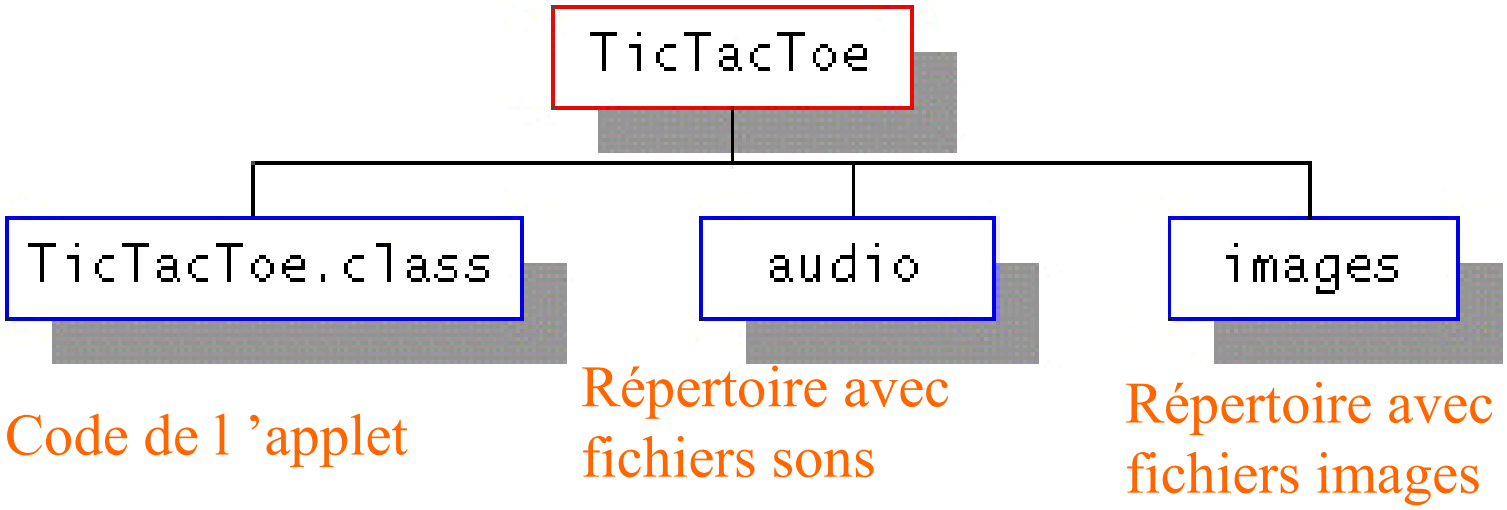
- Par défaut, le code de l ’applet est cherché dans le répertoire contenant le fichier HTML
- S'il est ailleurs, on donne le répertoire dans lequel il est situé grâce à l’attribut CODEBASE
- <APPLET CODE= PremierApplet.class CODEBASE=Exemples/

  WIDTH=200 HEIGHT=100>

  </ APPLET >
- Si c’est une adresse relative : recherche à partir du répertoire contenant le fichier HTML
- Si c’est une adresse absolue, on peut télécharger l ’applet de n ’importe où, voire même d ’un autre serveur web que celui qui contient la page HTML (CODEBASE=http://someServer/…)

Pour utiliser un fichier archive (.jar)

- Si l'applet a besoin de différentes ressources (fichiers images, sons) pour fonctionner et que ces ressources sont présentes dans un fichier jar (tictactoe.jar)
- Exemple



Utiliser l'attribut ARCHIVE de la balise APPLET

- <APPLET CODE= TicTacToe.class

  ARCHIVE="TicTacToe.jar"
  WIDTH=460 HEIGHT=160>
  </APPLET >

- le paramètre CODE est nécessaire pour indiquer où l'exécution commence (où est le code de l 'applet)
- le paramètre ARCHIVE peut comporter plusieurs fichiers jar

<APPLET CODE=Animator.class
ARCHIVE="classes.jar , images.jar , sounds.jar"
WIDTH=460 HEIGHT=160>

Fonctionnement de cette balise

- Quand le fichier archive est identifié, il est téléchargé en une seule transaction HTTP pour toutes les pièces du fichier jar
- Pendant l'exécution de l'applet, chaque fois qu'un fichier est requis (.class ou fichier son ou fichier image) :
  - Le navigateur commence à le chercher dans les fichiers ARCHIVE qui ont été téléchargés
  - S'il n'y est pas, sur le serveur Web servant l'applet dans le répertoire indiqué par CODEBASE

Autres attributs facultatifs de la balise APPLET

- ALIGN=…. justification de l'applet dans la page
- HSPACE = marge horizontale, en pixel, insérée à gauche et à droite de l'applet, entre l'espace de l'applet et le texte adjoint
- VSPACE= marge verticale, en pixel, insérée au dessus et en dessous de l'applet

- Valeurs pour l'attribut ALIGN :
  top, middle, bottom, right, left

Intérêt des applets : technologie « client léger »

- Apporte de l'interaction et de l'animation dans les pages html en ajoutant aux pages web du code java téléchargé
- tout se télécharge :
  - pas de configuration locale
  - plus de déploiement d'applications : évite d ’installer, maintenir, mettre à jour l'application sur **tous** les postes clients mais seulement sur le serveur
  - Pourquoi ?

    Car Java est indépendant des plate-formes ; une applet peut donc s’exécuter sur **tout** poste client possèdant la machine virtuelle Java
- Utilisation de l'intranet
- Exécution sur le poste client donc moins d ’encombrement du serveur

  ⇨ Mais, Attention : Problèmes de sécurité (voir à la fin)

Programmation d'une applet

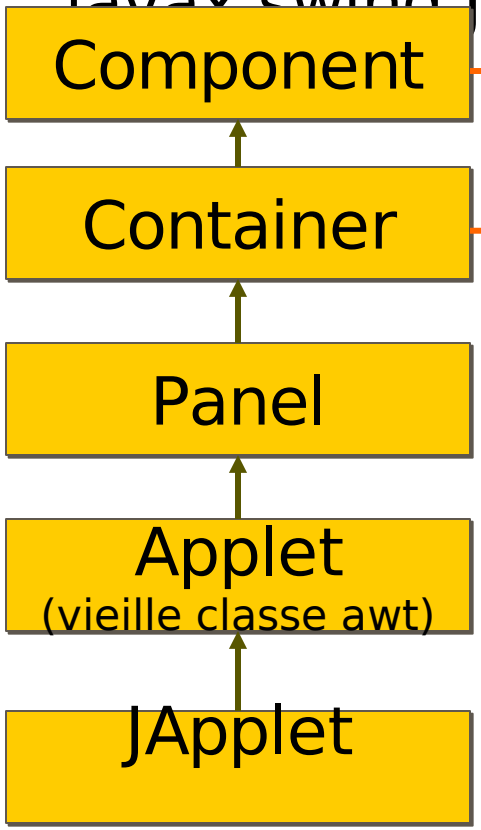
- Une applet ne peut donc être exécutée que si elle est intégrée dans une page HTML et elle s'exécute sous le contrôle d'un navigateur

- Une applet java fait toujours intervenir au moins deux fichiers
  - un fichier java compilé : prog.class
  - un fichier HTML (prog.html) qui contient la balise APPLET avec la référence au code prog.class

- Pour la mise au point d'une applet, on peut utiliser l'outil appletviewer

La classe JApplet : un top-level container

- Une applet est une instance de la classe javax.swing.JApplet



Gestion événementielle

Pour ajouter des composants,
les ajouter à son aire de travail

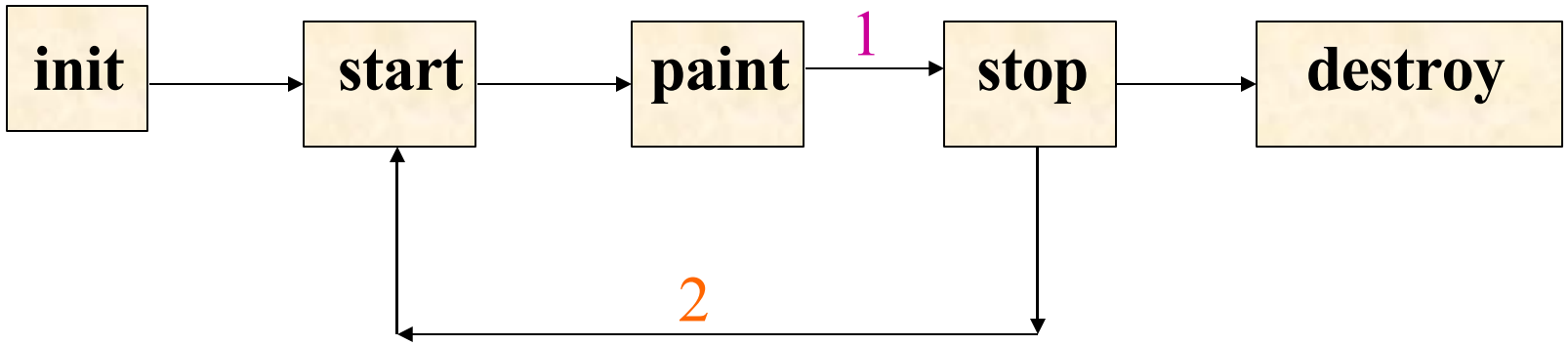
Mais n'hérite pas de JComponent
donc impossible de redéfinir paintComponent.
⇒ Si besoin d'affichage graphique :
Sous-classer JPanel (comme on faisait
pour une JFrame) et y redéfinir paintComponent

par défaut, possède un gestionnaire de
mise en page de type FlowLayout

Méthodes importantes liées au cycle de vie

Méthodes (public)	Appelée quand ?
void init()	**Initialisation :** quand l'applet est chargée Ne se produit qu'une fois dans la vie de l'applet
void start()	**Démarrage :** lors du lancement ou de la reprise de l'applet
void stop()	**Arrêt :** quand on quitte la page contenant l'applet
void destroy()	**Destruction :** quand on quitte le navigateur Ne se produit qu'une fois dans la vie de l'applet
void paint(Graphics g)	**Affichage**

Cycle de vie géré par le navigateur



1: quand l'utilisateur quitte la page html qui contient la balise de l'applet

2: quand l'utilisateur, revient sur la page html qui contient la balise de l'applet

Autres méthodes de JApplet

- URL getCodeBase()

  retourne l 'URL du répertoire à partir duquel l'applet est chargée

- URL getDocumentBase()

  retourne l 'URL (en absolu) du répertoire où se trouve le document HTML qui contient l'applet

  - Exemple, si l'applet est dans le document

    http://java.sun.com/products/jdk/1.2/index.html

    l 'URL retournée est : http://java.sun.com/products/jdk/1.2/

- Ces 2 méthodes retournent la même URL (même serveur, même répertoire) s'il n'y a pas d'attribut CODEBASE

Autres méthodes de JApplet

- Image getImage(URL url)

  retourne une instance d 'image correspondant au fichier image placé dans l 'URL (absolue) url

Exemple :

- si l'applet est dans le document

  http://java.sun.com/products/jdk/1.2/index.html
- si le fichier image, Terre.gif, est dans le répertoire

  http://java.sun.com/products/jdk/1.2/images
- Pour récupérer l'image correspondant (si on est dans une méthode de l'applet correspondant)

```
String nomFic=this.getCodeBase()+"images/Terre.gif ";
Image img= this.getImage(nomFic);
```

Exemple
Premier fichier : définit l'applet

```
public class AppletBonjour extends JApplet  {

public void init() {
    JLabel label = new JLabel("Bonjour",JLabel.CENTER);

    label.setBorder(BorderFactory.createMatteBorder(1,1,
    2,2,Color.black));
    Container c=this.getContentPane();
    c.add(label, "Center");
     }
}
```

Deuxième fichier : définit la page html qui intégrera l'applet

Fichier PremierApplet.html

```
<HTML> .....
<APPLET      CODE= AppletBonjour.class
             WIDTH=200          HEIGHT=100>
</ APPLET >
.....
</HTML>
```

pour tester l'applet, taper la commande :

appletviewer PremierApplet.html

Fonctionnement

- Une fois que le code de l'applet a été téléchargée et est en mémoire sur le poste client, comment s'exécute l'applet ?
  - le navigateur crée une instance de la classe AppletBonjour en appelant le constructeur sans argument de la classe
  - puis adresse à cet objet les méthodes init(), start() et paint()
  - puis le cycle de vie continue en suivant les actions de l'utilisateur jusqu'à l'exécution de destroy lors de l'arrêt de la machine virtuelle

- Grande différence avec une application locale autonome :
  - On n'écrit pas de méthode main.
  - C'est le navigateur qui gère le cycle de vie de l'applet

Autres attributs de la balise APPLET : PARAM pour passer des informations à une applet

- Pour définir des paramètres : attribut PARAM

```
<APPLET
CODE= "AppletAvecParam.class"
WIDTH=200 HEIGHT=100>
<PARAM NAME=nom VALUE=  " Joseph Cerrato">
</ APPLET >
```

- Pour récupérer des paramètres : appeler dans la méthode init de l'applet la méthode de **JApplet**:

```
String getParameter(String name)
```

Code de l ’applet

```
public class AppletBonjourParam extends JApplet  {
public void init() {

String nomPers= this.getParameter("nom");

if (nomPers==null)
        label = new JLabel("Bonjour", JLabel.CENTER);
else    label = new JLabel("Bonjour "+nomPers ,
JLabel.CENTER);

Container c=this.getContentPane();
c.add(label, BorderLayout.CENTER);
}
}    // fin de l'applet
```

Sécurité : problème crucial dans la programmation web

- Téléchargement de l ’applet du serveur vers le client doit être sûr : le poste client doit être protégé contre l'exécution d'une applet qui violerait l'intégrité de son environnement

- exemples d’actions dangereuses :
  - ouvrir une connexion réseau quelconque
  - accéder aux fichiers locaux du client

- Plus de détails dans la doc :

  …Java/docs/tooldocs/windows/policytool.html
  http://java.sun.com/docs/books/tutorial/security1.2/index.html

Évolution du système de sécurité

- Modèle de sécurité du JDK 1.0 : applet fonctionne dans un environnement très restreint, appelé bac à sable (sand box)
  - Interdiction d'ouvrir une connexion réseau sauf vers le serveur d'où l 'applet provient
  - Interdiction d'accéder aux fichiers locaux du client
- Modèle de sécurité du JDK 1.1 : concept d'applet signée
  - La signature est une clé numérique fonctionnant comme une marque de confiance
- Modèle de sécurité du JDK 1.2 : paramétrable ; le code (local ou téléchargé) peut être muni de permissions selon les besoins et l'origine certifiée de l'applet

Signature d'un message : principe

- La signature d'un message consiste à envoyer 2 informations :
  - le message en clair
  - le même message signé avec la clé privée de l'auteur du message.
- Ensuite le destinataire :
  - décrypte le message signé à l'aide de la clé publique de l'émetteur
  - vérifie si le résultat correspond au message en clair.

→ Chaque auteur doit avoir une signature composée d'une clé publique et d'une clé privée

Signature d'une applet : technique étape 1 : se créer une signature

- Pour se créer une signature, utiliser l'outil keytool :

keytool -genkey -alias gayral -keypass kpi135 -keystore gayralstore -storepass ab987c

- Paramètres :
  - genkey : paramètre indiquant que l'on souhaite générer un certificat
  - alias : alias identifiant l'auteur(identifiant la paire de clés dans la base de clés qu'il examine)
  - keypass : mot de passe limitant l'accès à la paire de clés qui va être générée
  - keystore : nom de la base de clés
  - storepass : mot de passe protégeant la base de clés
- Crée une base de clés appelé gayralstore et génère une nouvelle entrée composée de la paire clé publique/clé privée de l'auteur ainsi qu'un certificat auto-signé (signé par vous-même) et l'insère dans la base de clés sous l'alias spécifié
- Le mot de passe kpi135 pdevra toujours être utilisé pour accéder à l'entrée contenant cette clé.

Signature d'une applet : technique
étape 2 : signer l'applet avec cette signature

- Signer une applet, c'es signer le fichier jar contenant cette applet
  - Créer le fichier jar : **jar cvf applet.jar AppletBonjour.class**
- Signer le fichier jar avec l'outil jarsigner fourni dans le jdk
  - va utiliser le certificat créé précédemment

jarsigner -keystore gayralstore -storepass ab987c applet.jar gayral

Que fait jarsigner ?

- Ajoute 2 fichiers dans le répertoire META-INF de l'archive
  - un fichier de signature de type alias.sf contenant les signatures pour chaque fichier contenu dans le JAR (chaque entrée du fichier manifest).
  - un fichier bloc de signature de type alias.dsa ou alias.rsa contenant une signature pour le fichier alias.sf ainsi que le certificat (et la clé publique donc) du signataire.

- Un fichier JAR peut être signé par plusieurs entités (en lançant jarsigner plusieurs fois sur ce fichier), afin de permettre différents profils d'autorisations pour la même applet ;

JDK 1.2 : le gestionnaire de sécurité

- L'environnement java utilise un gestionnaire de sécurité pour vérifier les accès aux ressources
- Un gestionnaire de sécurité est systématiquement chargé par les navigateurs : les applets tournent donc toujours sous son contrôle
  - Une applet n'aura pas l'accès aux ressources tant qu'on ne lui aura pas explicitement donné les permissions
  - les permissions seront données dans par une entrée dans un "fichier de police" (policy file)
  - Chaque fois que l'applet effectue une action susceptible de faillir à la sécurité, l 'environnement demande au gestionnaire de sécurité de vérifier les permissions.

Utilisation d'un Policy file

- C'est un fichier texte particulier qu'on crée soit "à la main", soit en utilisant un outil graphique appelée par la commande : policytool
- Il existe un policy file par défaut nommé .java.policy dans le "home directory"
- Exemple de fichier :

```
grant codebase "http://www.games.com",
signedBy "Duke",
 { permission java.io.FilePermission "/tmp/games", "read, write"; };
```

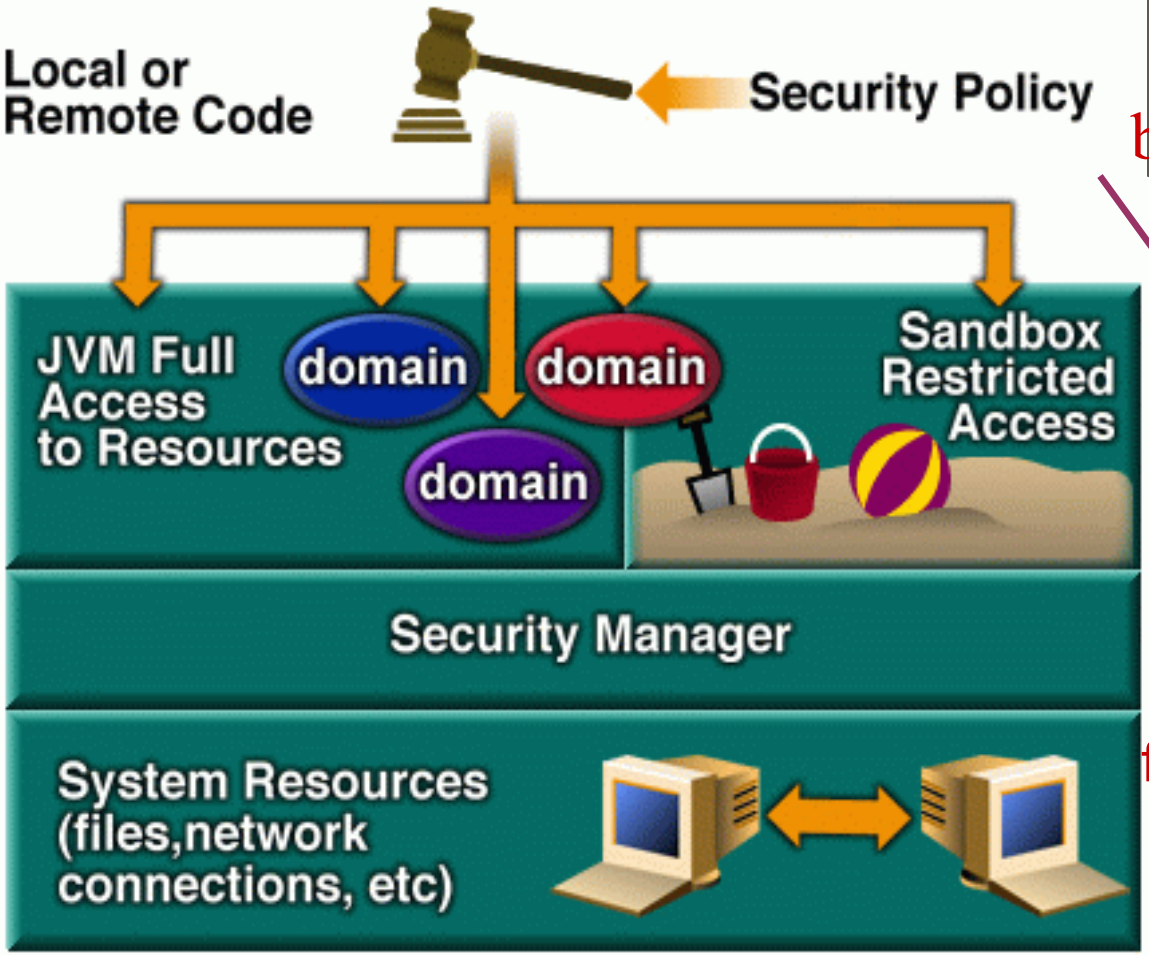
Cela permet au code téléchargé de "www.games.com", signé par "Duke", de lire et d'écrire dans le répertoire "/tmp/games

Poubelle

Sécurité : problème crucial dans la programmation web

- 3 aspects importants :
  - Authentification : être capable de vérifier l ’identité des parties impliquées
  - Confidentialité : garantir que seules les parties impliquées comprennent la communication
  - Intégrité : vérifier que le contenu de la communication n ’a pas changé pendant la transmission

modèle de sécurité du JDK 1.0



Environnement très restreint : bac à sable (sand box)

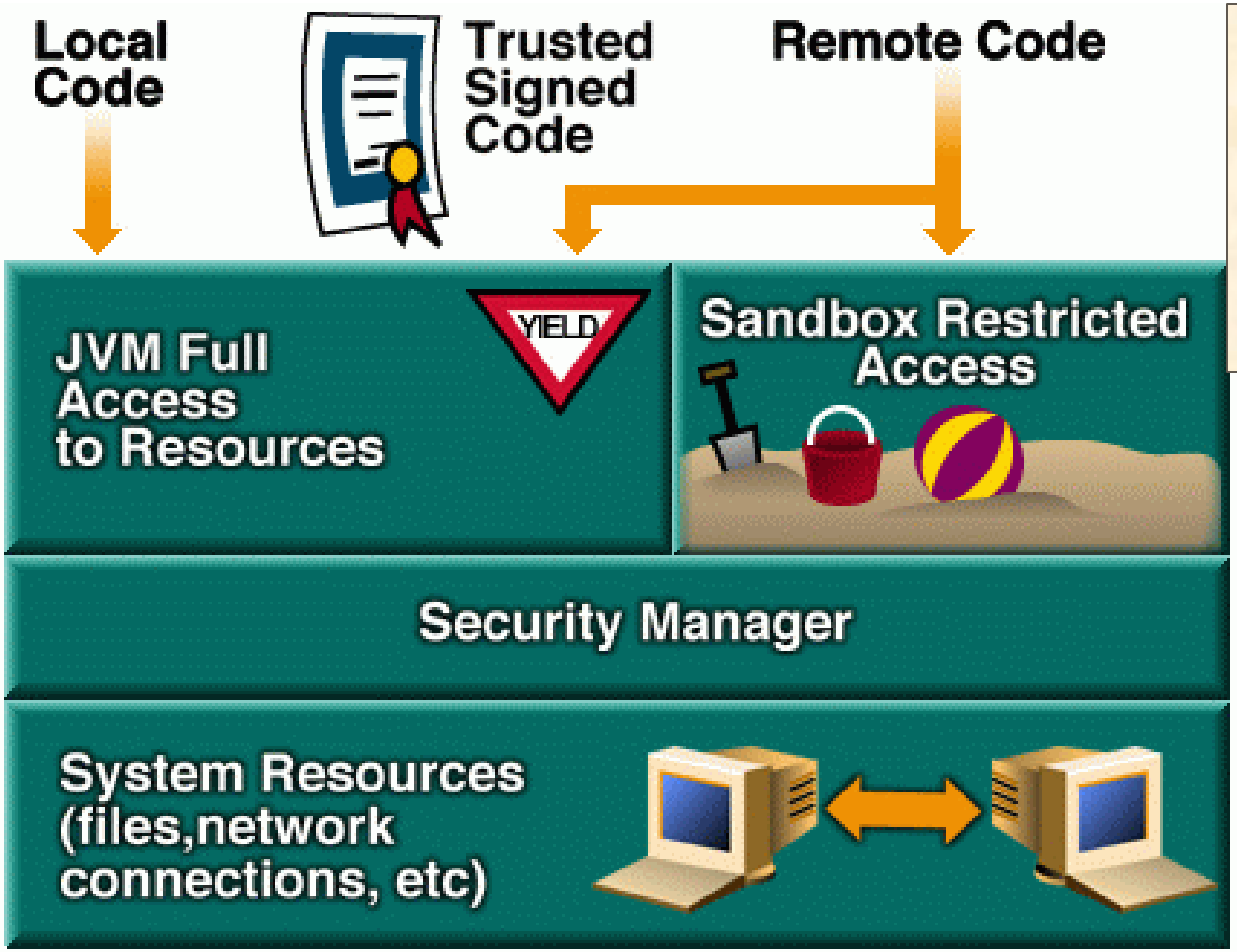
~~Toucher au disque de la machine client~~

~~Lancer un programme du poste client~~

~~Ouvrir des connexions réseaux vers d'autres serveurs que celui qui a fourni le code de l'applet~~

~~Lire certaines propriétés du système client : user name, user home, classpath,...~~

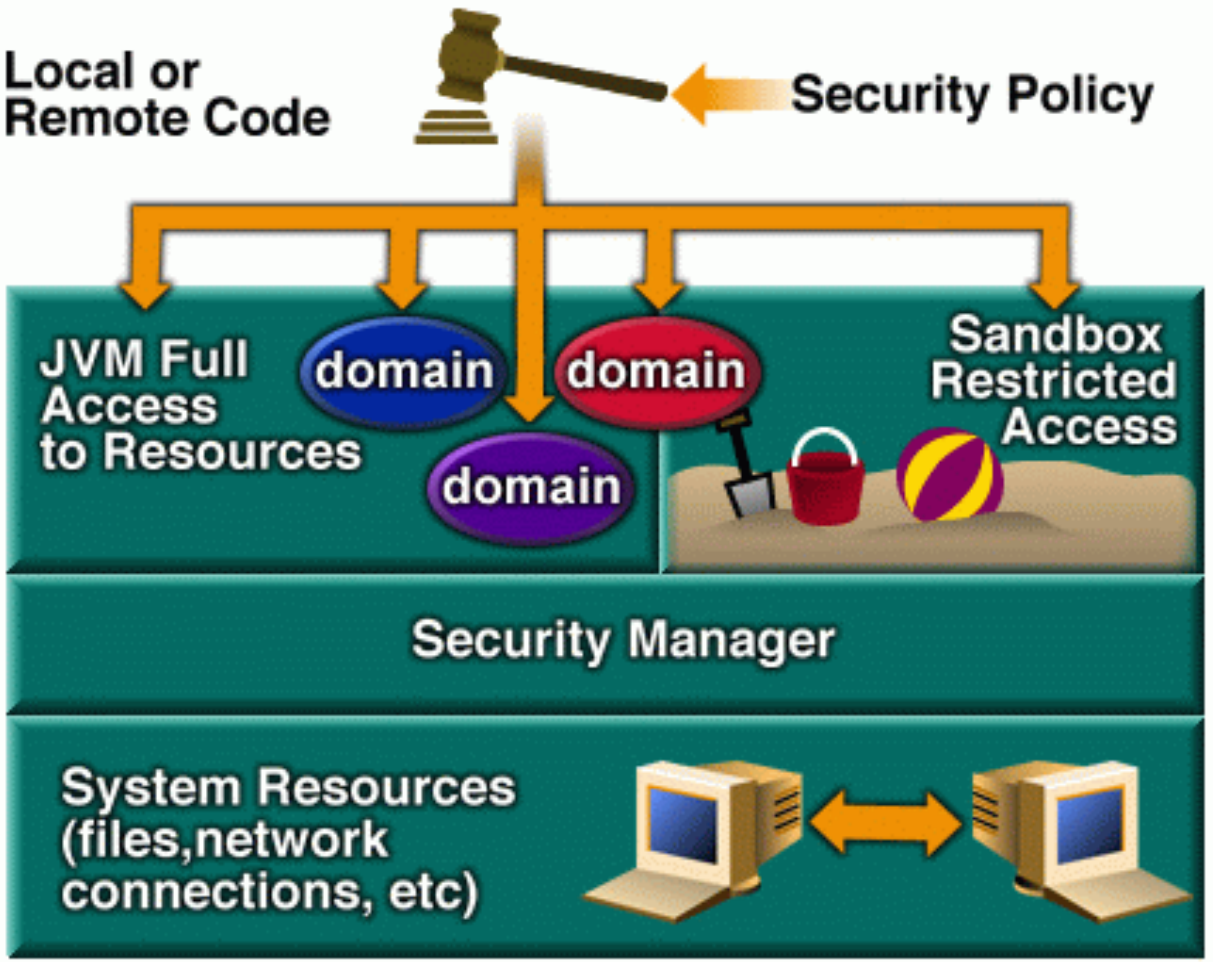
Modèle de sécurité du JDK 1.1
Notion de signature (ou certification)



- Concept d'applet signée
La signature est une clé numérique fonctionnant comme une marque de confiance

Si signature de l'applet est jugée de confiance (marquée dans le navigateur client)
⇒ traitée comme une appli locale
Sinon
⇒ restreinte au bac à sable

Modèle de sécurité du JDK 1.2
paramétrable



Le code, local ou téléchargé,
peut être muni de permissions
selon les besoins et l'origine certifiée de l'applet

Définition de **domaines**
ensembles de classes
dont les instances
ont le même ensemble de permissions