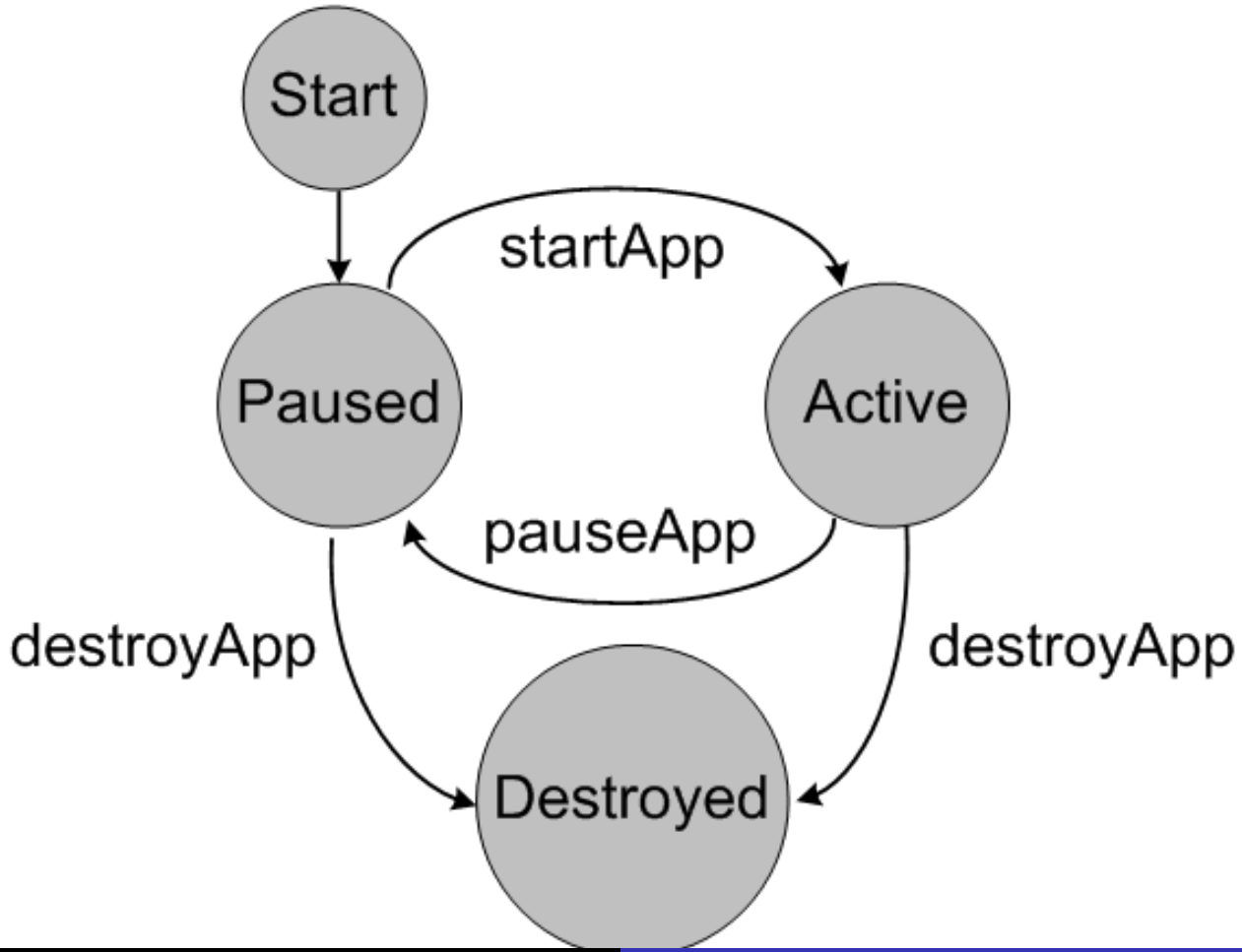
Programmation des Systèmes Mobiles et Sans Fil
Mobile and Wireless Systems Programming

First step

Introduction

- Main Java class for a MIDP based application
- javax.microedition.midlet.MIDlet
- 3 possible states :
  - Active
  - Paused
  - Destroyed

MIDlet lifecycle



Application Management Software

- How to launch a MIDP application ?
- Not part of the Java ME platform
- Start, pause and destroy the MIDlet
- Manage MIDlet states
- MIDlet ineraction with the mobile OS (Connections...)
- ...

Smallest MIDP application

Listing 1 – SmallestMIDlet.java

```
import javax.microedition.midlet.MIDlet;

public class SmallestMIDlet extends MIDlet {
        public void startApp(){}
        public void pauseApp(){}
        public void destroyApp(boolean b){}
}
```

Main MIDlet's methods

- startApp() : entry point for a MIDP application (main() function)
- pauseApp() : called when the MIDlet enter the paused state (incoming call,...)
- destroyApp() : called when the application is destroyed

Other MIDlet's methods

- int checkPermission(String permission)
- String getAppProperty(String key)
- void notifyDestroyed()
- void notifyPaused()
- boolean platformRequest(String URL)
- void resumeRequest()

Java Application Descriptor

- Simple text file formatted (attribute :attribute value)
- Application description
- Avoid mobile phones and applications incompatibility (CLDC, MIDP, Available memory...)
- Strict rules
- Installation-required, installation-optional, runtime-required, and runtime-optional attributes

Mandatory JAD attributes

- MIDlet-Name
- MIDlet-Version
- MIDlet-Vendor
- MIDlet-Jar-URL
- MIDlet-Jar-Size
- MIDlet-<n>
- MicroEdition-Profile
- MicroEdition-Configuration

MIDlet description

MIDlet-n : MIDletName , [IconPathname] , ClassName

- **n** : the MIDlet number, starting at 1 and incremented by 1 for each MIDlet
- **MIDletName** : the user-visible name of the MIDlet
- **IconPathname** : absolute pathname of a .png image file within the JAR
- **ClassName** : name of a class that extends javax.microedition.midlet.MIDlet and has a public, zero-argument constructor

Some optional JAD attributes

- MIDlet-Description
- MIDlet-Icon
- MIDlet-Info-URL
- MIDlet-Data-Size
- MIDlet-Permissions
- MIDlet-Permissions-Opt
- MIDlet-Push-<n>
- MIDlet-Install-Notify
- MIDlet-Delete-Notify
- MIDlet-Delete-Confirm

Manufacturers optional JAD attributes

- Nokia
  - Nokia-MIDlet-Category
  - Nokia-Scalable-Icon
  - ...
- Vodafone
  - MIDxlet-ScreenSize
  - MIDxlet-Application-Range
  - ...
- ...

Manage attributes

- Get attributes value
  - String getAppProperty(String propertyName) [javax.microedition.midlet.MIDlet]
- Add personal attributes
  - Can't start with MIDlet- nor MicroEdition-
  - Useful to change application properties without code building

JAD file example

MIDlet-<1> : SmallestMIDlet,/res/icon.png,SmallestMIDlet
MIDlet-Description : The smallest MIDlet
MIDlet-Jar-Size : 1234567
MIDlet-Jar-URL : myMidlet.jar
MIDlet-Name : SmallestMIDlet
MIDlet-Vendor : Youssef Ridene
MIDlet-Version : 1.0
MicroEdition-Configuration : CLDC-1.1
MicroEdition-Profile : MIDP-2.0

Tips and tricks

- JAD can't prevent API absence
- Only one attribute per line
- Case sensitive
- Spaces are ignored
- Attribute namespace is read-only ; there is no MIDlet.setAppProperty() method
- Attribute names should appear only once

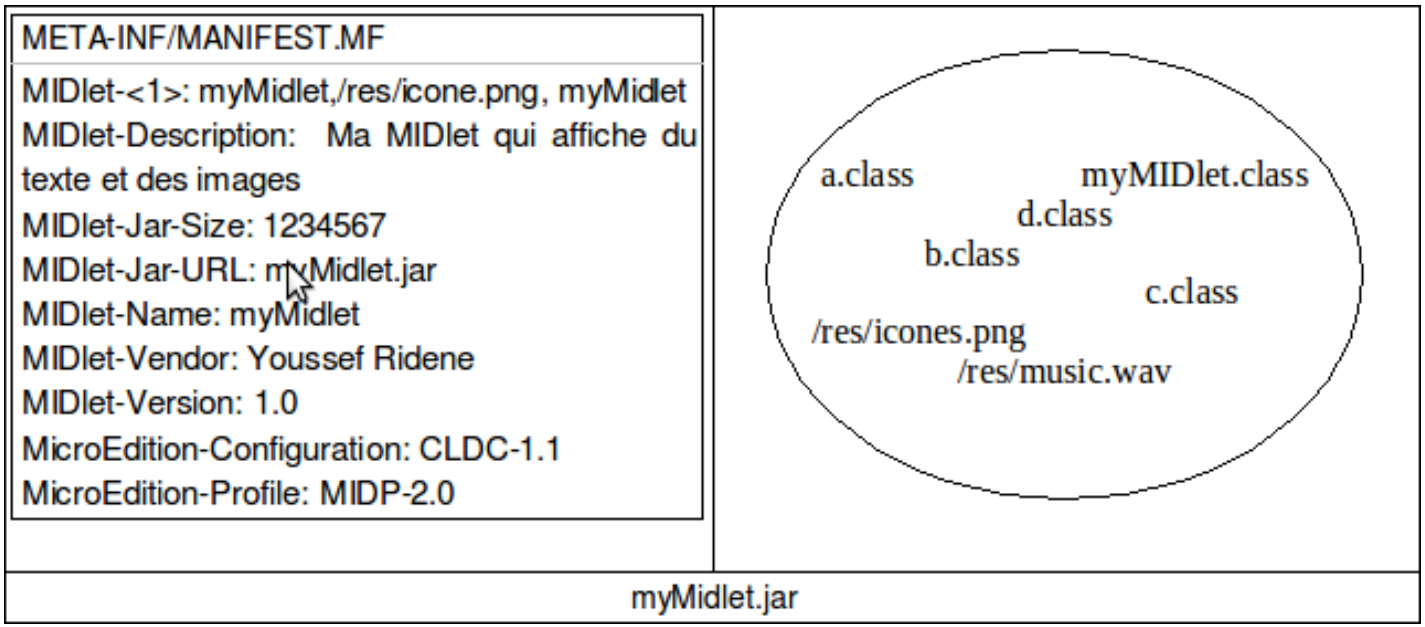
Java ARchive contents

- Normal Java archive
  - .class files and ressources (audio, images, video...)
  - MANIFEST.MF
- Optimized for Java ME platforms

Manifest file

- /META-INF/MANIFEST.MF
- same syntax as the JAD file and may share the same attributes
- attributes in the JAD must agree with those in the manifest
- Important for the security model introduced in MIDP 2.0
- Mandatory attributes
  - MIDlet-Name
  - MIDlet-Version
  - MIDlet-Vendor
  - MicroEdition-Profile
  - MicroEdition-Configuration
  - MIDlet-<m>

JAR example


META-INF/MANIFEST.MF
MIDlet-<1>: myMidlet,/res/icone.png, myMidlet
MIDlet-Description: Ma MIDlet qui affiche du texte et des images
MIDlet-Jar-Size: 1234567
MIDlet-Jar-URL: myMidlet.jar
MIDlet-Name: myMidlet
MIDlet-Vendor: Youssef Ridene
MIDlet-Version: 1.0
MicroEdition-Configuration: CLDC-1.1
MicroEdition-Profile: MIDP-2.0
a.class
myMIDlet.class
d.class
b.class
c.class
/res/icones.png
/res/music.wav
myMidlet.jar

MIDlet suites

- A group of MIDlets distributed in a single JAR file
- MIDlet-1, MIDlet-2,. . . , MIDlet-n
- Installed onto a device as a single entity
- MIDlet suite can not launch another MIDlet suite

Pre-processing !

- Optional but often used => Fragmentation issue
- #if, #else, #endif...
- Dummy example : French or English

```
        String message = null;
#if FR
        message = "Bonjour le monde!";
#elif EN
        message = "Hello World!";
#endif
```

Pre-processing !

- > cpp -P -DEN MyClass.java.c -o MyClass.java
- Input : Java class with preprocessing directives
- Output : Java class
- Not only cpp but better : Ant and Antenna
- Preprocessing and porting
  - Knowledge of the specific behavior of each device model
  - Java features supported by each device model
  - Manage code but also ressources
  - Invest in the development of an automated compilation and packaging process

Porting solutions for Java ME developers ?

- Automated tools : Neomades, Mobile Distillery, Javaground
- Manual tools : Off-shore (India, Romania...)

Compiling

- Like Java SE but for Java ME platform
- Input : .java file(s)
- Output : .class file(s)
- Example :

```
$javac -bootclasspath myMidlet.java
/home/ry/lib/midpapi10.jar :/home/ry/lib/cldcapi10.jar
```

What is obfuscating ?

- detecting and removing unused classes, fields, methods, and attributes
- optimizing bytecode and removing unused instructions
- renaming the remaining classes, fields, and methods using short meaningless names
- preverifying the processed code

Why obfuscating ?

- creating more compact code, for smaller code archives, faster transfer across networks, faster loading, and smaller memory footprints
- Making programs and libraries harder to reverse-engineer
- Listing dead code, so it can be removed from the source code
- Retargeting and preverifying existing class files for Java Micro-Edition

Obfuscating in practice

- Proguard : http ://proguard.sourceforge.net/)
- .jar program
- Options :

-injars in.jar
-outjars out.jar
-libraryjars /usr/local/java/wtk2.1/lib/midpapi20.jar
-libraryjars /usr/local/java/wtk2.1/lib/cldcapi11.jar
-overloadaggressively
-repackageclasses ”
-allowaccessmodification
-microedition
-keep public class mypackage.MyMIDlet

Obfuscating tips

- Avoid obfuscating while developing (debugging)
- Some issues with BlackBerry
- MIDlet class isn't obfuscated => Lesser important code
- Unused ressources aren't managed by obfuscators

Pre-verifying

- verify well-formness of classes
- less sofisticated then Java SE preverifier
- Example :

```
$preverify -classpath
/home/ry/lib/midpapi10.jar :/home/ry/lib/cldcapi10.jar -d .
myMidlet
```

Packaging

- jar tool
- Example :

```
$jar cvmf MANIFEST.MF myJar.jar myMIDlet.class
```

Over The Air

- Large scale deployment
- Device's built-in browse and remote server
  1. The client device sends an HTTP GET request to the server for the given URL.
  2. The server sends an HTTP response with the suite's JAD file as the message body
  3. The client verifies the HTTP response and extracts the suite's MIDlet-Jar-File and MIDlet-Jar-Size attributes.
  4. The device sends an HTTP GET request for the JAR file.
  5. The server sends an HTTP response with the JAR file as the message body.
  6. The device verifies the message and the JAR file.
  7. The device asks the user to verify installation.

Domains

- Identified 3rd party protection domain Ű signed by or for a party which is known (formerly known as the Trusted third party domain)
- Operator domain Ű signed by an operator or a carrier
- Manufacturer domain Ű signed by a device manufacturer

Unsigned applications will be assigned to the unidentified third party protection domain (formerly known as the Untrusted third party domain)

Application signing

- 2 additionnals fields
  1. MIDlet-Certificate-1-1
  2. MIDlet-Jar-RSA-SHA1
- Keytool
- Permissions :
  - MIDlet-Permissions : javax.microedition.io.Connector.http
  - MIDlet-Permissions-Opt : javax.microedition.io.Connector.https

Unsigned applications will be assigned to the unidentified third party protection domain (formerly known as the Untrusted third party domain)

Conclusion

- MIDlet
- JAR and JAD
- Pre-processing for fragmentation issue
- Compiling, Obfuscation, Preverifying and packaging