Une introduction à la plate-forme J2ME

J.2.M.E
Java 2 Micro Edition

Michel RIVEILL
Polytech'Nice - Sophia
Email : riveill@unice.fr
Web : http://rainbow.polytech.unice.fr

Objectifs

- Présenter les différentes configurations et profils de J2ME
- Présenter quelques aspects techniques liés à KVM, CLDC et MIDP
- Présenter l'environnement de développement J2ME

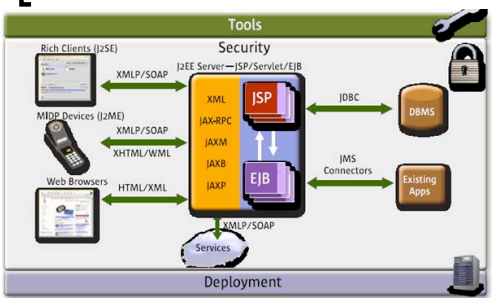
2

Présentation de J2ME

Configurations et profils

3

J2ME et services Web



4

J2ME

- Ubiquitous computing
  - Mainframe : un ordinateur, plusieurs personnes
  - PC : un ordinateur, une personne
  - Ubiquitous computing : plusieurs ordinateurs en réseau, une personne
    - Evolution vers les standards de l'Internet : 'Java' + IP dans chaque équipement + XML
    - Conscience de la localisation de chaque équipement
    - Interoperabilité entre les réseaux, les infrastructures
- Standart pour le développement des applications sur réseaux sans fils, avec des équipements de nature très diverses
  - J2ME = 40% des nouveaux téléphones installés au Japon en 2001
  - J2ME = 18 millions d'équipement en 2001
  - J2ME = 70% des smartphones, PDAs en 2004

5

Diversité des terminaux…
mais un seul langage de programmation : Java

- Une grande variété de terminaux
  - PCs
  - PDAs
  - Téléphones
  - Pageurs
  - Terminaux embarqués
  - Cartes
- Le monde Java est vaste : 1Q04
  - 650 millions de PCs
  - 350 millions de téléphones/PDAs
  - 500 millions de SIM et Smart Cars
- Un seul langage de programmation : Java
  - La seule de manière de programmer sur certains terminaux
  - Partager le même langage, la même conception depuis le terminal jusqu'aux serveurs
  - Permet le chargement dynamique de code
  - Programmes compacts et portables
  - Développement rapide et sûr
  - Atelier et outils homogènes

6

Hétérogénéité des modes de communication (1)

- Equipement radio
  - Fréquence : 3 Khz à 300 Ghz
  - Portée : 3 cm à 300 m, qq km
- La téléphonie
  - 1 G : analogique, défini pour la voix
  - 2 G : numérique, défini pour la voix (GSM) http://www.gsmworld.com
    - SMS http://www.gsmworld.com/technology/sms.html
    - 2.5 G : amélioration pour transfert de données (GPRS) http://www.gsmworld.com/technology/gprs.html http://www.anywhereyougo.com/ayg/yg/wireless/Article.po?page=96110 (article d'introduction sur le GPRS
  - 3 G : défini pour les données (UMTS) http://www.gsmworld.com/technology/3g_future.html
  - 3G+

7

Hétérogénéité des modes de communication (2)

- La convergence des services
  - Java + XML + IP
  - Service basé sur la localisation (E911 aux US : permet de connaître sa position sur un réseau GSM)
  - Interopérabilité entre réseau et zone géographique
- Les communications radio dans la bande de 2.4 GHz
  - Wireless Ethernet (IEEE 802.11b) : http://www.weca.net
    - Optimisé pour transmission de données jusqu'à 100 m, réseau de campus
  - Bluetooth : http://www.bluethooth.com
    - Optimisé pour 30 m ou moins, réseau personnel
    - Java API : http://java.sun.com/jcp/jsr/jsr_082_bluetooth.html
  - HomeRF (combinaison de IEEE 802.11b et de DECT) : http://www.homerf.org
    - Optimisé pour transmission de la voix jusqu'à 50 m

8

Hétérogénéité des langages de description (1)

- Proposition pour la téléphonie
  - Approche 'européenne'
    - HTTP / WAP (Wireless Application Protocol)
    - HTML / WML (Wireless Markup Language)
    - http://www.wapforum.org/what/technical.html
  - Approche 'japonaise'
    - C-HTML (Compact HTML) + HTTP
    - iMode : http://www.mobilemediajapan.com/imodefaq

9

Vers une homogénéité des langages de description (2)

- Aujourd'hui, demain ?
  - XHTML : convergence des différentes approches
  - http://www.w3.org/TR/xhtml1
- Mode déconnecté
  - Réseaux sans fil + équipement nomade
    = fonctionnement en mode déconnecté
    → Nécessité d'avoir un protocole pour décrire les données et gérer leurs cohérences
  - SyncML : exemple de langage
    - Protocole commun utilisé pour la synchronisation des données entre différents équipements
    - Langage construit sur XML
    - http://www.syncml.org
- Gestion de la voix
  - Informatique diffuse = multiples équipements hétérogènes
    → nécessité d'avoir un langage pour décrire les sons
  - VoiceXML : exemple de langage
    - Pour la reconnaissance et la synthèse vocale
    - Langage construit sur XML
    - http://www.voicexml.org

10

SyncML (http://www.syncml.org)

- Message XML comportant des éléments de synchronisation
  - Add, Alert, Atomic, Copy, Delete, Exec, Get, Map, Replace, Search, Sequence, Sync
- 2 représentations
  - Texte WBXML text/vnd.syncml-xml
  - Bytecode application/vnd.syncml-wbxml
- Transport
  - HTTP, WSP, …, JMS

11

Naviguer dans le monde Java

- J2SE : Java 2 plateform, Standard Edition
  - Solutions pour les postes clients : applications autonomes, applets, …
- J2EE : Java 2 platform, Enterprise Edition
  - J2SE+développement d'appications côté serveur (Servlet, EJB, JMS, XA, SOAP, …)
  - Solutions pour les entreprises : E-commerce, E-business
- J2ME : Java 2 plateform, Micro Edition
  - J2SE - nombre réduit de paquetage, machine virtuelle plus 'légère'
  - Solutions pour les terminaux embarqués : téléphones, PDAs, TV box, …
- Les différents environnements sont basés sur le même langage Java mais avec
  - Différentes JVMs et différentes APIs

12

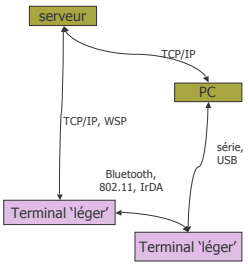
C'est quoi un terminal 'sans fil'

- PCs
- PDAs
- Téléphones
- Pageurs
- Véhicules
- Systèmes embarqués
- Smart carte

13

Architecture d'une application sur terminaux mobiles

	Version 1	Version 2
Serveurs		J2EE
PCs	JDK 1.1.x	J2SE
PDAs	Personnal Java	J2ME/CDC
Téléphones et pageurs		J2ME/CLDC
Terminaux embarqués	Embedded Java	J2ME/CLDC
Cartes	JavaCard	JavaCard



WSP : Wireless Session Protocole (équivalent http pour le WAP : http://www.guill.net/index.php?cat=3&pro=59)

14

Pourquoi J2ME ?

- Le nombre de terminaux 'légers' dépassera largement celui des ordinateurs personnels
  - Ventes PCs : 07 (272,223,000), 06 (239,211,000)
  - Ventes Smartphone : 3T07 (32,853,608), 3T06 (20 867 519)
  - Ventes téléphones mobiles : 07 (1,152,839), 06 (990,862)
    - 30 à 50 % de ces terminaux seront connectés à Internet
    - Plus nombreux que les stations de bureau (320 millions en 2003)
- Il sera nécessaire de télécharger des logiciels sur les terminaux légers
  - Être capable d'adapter les services à la très grande variété des équipements
  - Utilisation des réseaux sans fils (téléphone → informatique)
- J2ME est la plate-forme pour le développement de service sur ces terminaux

15

J2ME Configurations et profiles

- Une seule plate-forme J2ME ne peut couvrir tous les besoins
  - Les terminaux sont trop différents
  - Les besoins applicatifs sont très variables
- Configurations J2ME
  - Bibliothèque minimale et JVM
  - 2 configurations
    - CLDC : connexion limitée (JSR 36)
    - CDC : connecté (JSR 30)
- Profils J2ME
  - Bibliothèques complémentaires
    - Pour des terminaux particuliers
    - Pour un domaine applicatif particulier
  - Exemples :
    - Mobile Information Device Profile (MIDP : http://java.sun.com/products/midp/)
    - Foundation (http://java.sun.com/products/foundation/index.jsp)
      - API proche de J2SE sans IHM
    - Personal (http://java.sun.com/products/personalprofile/index.jsp)
      - API graphique compatible avec AWT
    - Bluetooth, ...
- Configurations et profils sont définis par la 'Java Community Process'

16

Configuration

- Une configuration définie une machine virtuelle et un ensemble de bibliothèques minimaux pour
  - Un ensemble de terminaux qui possèdent des caractéristiques similaires (tailles mémoires, capacité du processeur, etc)
- Elles sont définies par le 'Java Community Process'
  - Test de compatibiltés
  - 2 configurations sont disponibles
    - Connected Limited Device Configuration (CLDC)
    - Connected Device Configuration (CDC)

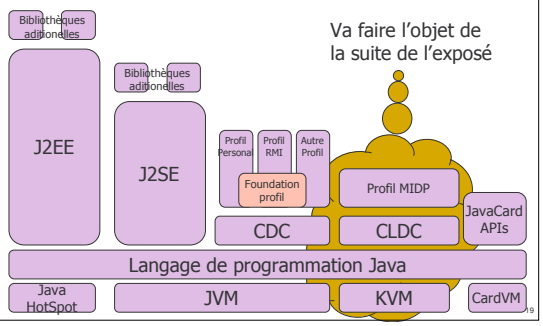
17

Profils

- Un ensemble d'APIs qui complètent la configuration pour définir des bibliothèques pour
  - Un domaine applicatif
  - Un type de terminaux
- L'objectif est de donner plus de flexibilité pour maintenir la portabilité des applications entre les terminaux
- Ils sont définis par le 'Java Community Process'
  - Test de compatibiltés

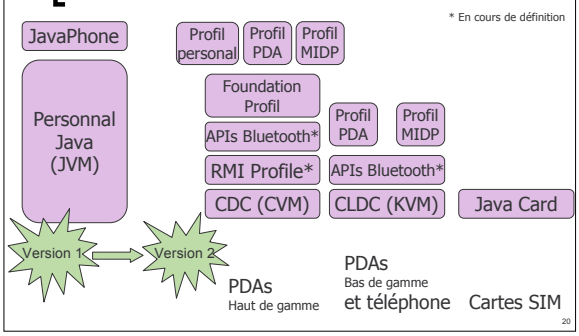
18

Plateforme Java 2



19

Les APIs Java pour les mobiles



20

Les configurations…

- **Connected Limited Device Configuration (CLDC)**
- Spécification définies dans JSR-000030
- Pour les terminaux avec
  - 128 KB – 512 KB de mémoire (RAM+ROM)
  - Batterie limitée
  - Connectivité réseau
  - Interfaces utilisateurs contraintes
- On peut télécharger les spécifications et l'implémentation de référence

- **Connected Device Configuration (CDC)**
- Spécification définies dans JSR-000036
  - JSR 36 (Personnal)
  - JSR 46 (foundation)
  - JSR 62 (Personnel, nouvelle version du JSR 36)
  - JSR 66 (RMI)
- Pour les terminaux avec
  - 512 KB ROM minimum
  - 256 KB RAM minimum
  - Connectivité réseau
  - Supporte une JVM complète
  - Interfaces utilisateurs contraintes

21

Profil pour les terminaux mobiles (MIDP)

- Le premier profil J2ME défini
- Les terminaux cibles implémentent CLDC
- Le profil défini
  - Affichage
  - Gestion de données persistantes
  - Envoie de message (SMS, email, etc)
  - Sécurité
  - Réseaux sans fils
- Implémentation de référence disponible

22

Exemples

Votre MIDlet — Annuaire, horaires de train, jeux, …

Profil pour terminaux mobile — IHM, HTTP networking, …

J2ME — CLDC = KVM + J2ME

KVM — Threads, pas de float

Processeur DSP — Exemple ARM : 32-bit RISC, 256 K ROM, 256 K Flash, 64 K RAM

23

Profil J2ME

Profil	Configuration	JSR#	JCP état
MID http://java.sun.com/products/midp/	CLDC	37	Version finale
Foundation http://java.sun.com/products/foundation/index.jsp API proche de J2SE sans IHM	CDC	46	Version finale proposée
Personal http://java.sun.com/products/personalprofile/index.jsp API graphique compatible avec AWT	CDC	62	Groupe d'experts formé
PDA	CLDC	75	Groupe d'experts formé
Bluetooth	CLDC	82	Groupe d'experts formé
RMI	CDC	66	Version finale proposée

24

Java technologies

- Téléphones
  - Liste disponible à http://www.javamobiles.com
- PDAs

PalmOS	MIDP pour Palm (Sun)	J2ME MIDP
Windows CE 2.11	Personal Java Runtime (Sun)	PersonalJava
PalmOS, WindowsCE	IBM J9	Différentes bibliothèque
…		

25

CDC et profils de base

Connected Device Configuration
Profil 'foundation' et 'personnal'

26

Pas dans ce cours

- CDC
  - Configuration pour terminaux mobiles permettant d'exécuter une JVM 'standard'
    - SetTopBox, WebPhone, Console de jeux
    - Microprocesseur (32 bits), >256 KB de mémoire volatile, > 512 KB de mémoire non volatile
  - Profils spécifiques permettant de modulariser et de réduire le code embarqué
    - JVM+bibliothèque CDC = 2 MB

27

CLDC et KVM

Connected Limited Device Configuration

K Virtual Machine

28

Composants clés

- Bibliothèques additionnelles (IHM, stockage, transmission, …)
- 'Plus petit commun dénominateur'
- Machine virtuelle allégée

29

Terminaux cibles

- 128 – 512 KB de mémoire
  - > 32 KB de mémoire volatile
  - > 128 KB de mémoire non volatile
- Processeurs ayant 16 ou 32 bits
- Fonctionnement sur batterie
- Communication réseau limitée : 9600 bps
- Très large diffusions des terminaux
  - PDAs, Téléphones, …

30

Objectifs

- Définir un ensemble de technologie Java homogène pour une large gamme de terminaux ayant des contraintes techniques
  - Mémoire, énergie, vitesse processeur, taille écran, connexion réseau
- Permettre la diffusion d'application sur ces terminaux de manière sûres, dynamiques, flexibles, évolutives, …
  - Construction d'application selon des architectures 3 parties
  - Travail en collaboration avec les fournisseurs de terminaux mais aussi les fournisseurs d'applicatifs

31

Domaine de couverture de CLDC

- Pris en charge
  - Machine virtuelle Java et langage Java
  - Modèle de sécurité
  - Entrée / sortie
  - Support réseau
  - Internationalisation
- Pas pris en charge
  - Cycle de vie et installation des applications
  - Support des IHMs
  - Gestion des événements
  - Modèle applicatif de haut niveau
  - Accès aux bases de données
  - ***Ces aspects sont définis dans des profils spécique***
    - ***Exemple : MIDP***

32

KVM

- 'petite' machine virtuelle Java construite pour les terminaux contraint
  - Qq dizaines de kilobytes
    - Implémentation modulaire en C : 24.000 lignes de code
    - 40 à 80 kB (taille statique) selon les options de compilation
      - taille vs vitesse, mise au point
    - Sur PalmOS et Win32 environ 60 kB
    - Fonctionne à 30-80 % de la vitesse du JDK (sans JIT)
  - Comprend les parties communes de J2ME
    - Compatible avec JVM (avec qq restrictions)
      - Pas de gestion des flottants (qui n'existent pas sur les processeurs visés), pas de double non plus
      - Gestion spécifiques des fils d'exécution (pas de groupe de threads)
      - Possibilité de porter facilement l'algorithme de GC
  - CLDC fonctionne sur la KVM

33

Principales restrictions

- JNI
- Réflexion
- Groupe de fils d'exécution
- Référence légère
- Finalisation
- Support incomplet des erreurs
- Nouvelle implémentation du chargeur (classloader), pas de possibilité d'en faire un nouveau
- Nouveau vérificateur de bytecode

- Les sources sont disponibles sous la licence Sun (Sun Community Source Licence)
- Il existe une implémentation de référence du CLDC 1.0 (Version stable complète 1.0.2 en 2001)
  - Win32
  - PalmOS (3.01 ou ultérieur)
  - Solaris
  - Linux
  - De nombreux autres portages de la KVM et de CLDC ont été fait par des partenaires sur des plateformes très diverses.

34

Bibliothèque CLDC

- Classes héritées de J2SE
  - Java.lang.*
  - Java.io.*
  - Java.util.*
- Nouvelles classes
  - Javax.microedition.io.*
  - CLDC Internationalisation
  - E / S
    - InputStreamReader(InputStream);
    - InputStreamReader(InputStream, String);
    - OutputStreamWriter(OutputStream);
    - OutputStreamWriter(OutputStream, String);
- Propriétés CLDC
  - microedition.platform
  - microedition.encoding
  - microedition.configuration
  - microedition.profiles

35

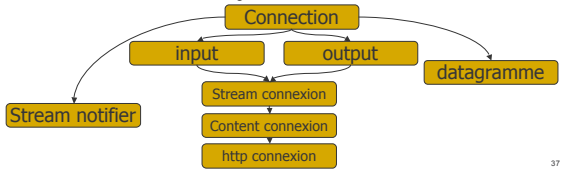
Support réseau

- Les bibliothèques J2SE réseau, IO et stockage sont trop grosses pour les terminaux visés
  - Plus de 100 classes
  - Plus de 200 kB
- Elles n'ont jamais été construites pour ces terminaux
  - TCP/IP doit être disponible
  - Pas ou peu de possibilité pour introduire de nouveaux protocoles : Bluetooth, IrDA
- Introduction de nouvelles classes (Generic Connection Framework)
  - Plus cohérentes
  - Support de différents types de protocoles, extensibles
  - Compatibilité avec les bibliothèques existantes

36

Generic Connection Framework

- 6 interfaces de bases
  - Input série
  - Output série
  - Communication par datagramme
  - Communication par circuit virtuel
    - TCP, connection HTTP légère
- CDLC ne défini pas de protocole de communication
  - C'est le rôle des profils
- L'implémentation de référence possède qq protocoles
  - Peut servir de point de départ



37

Exemple de connection

- Forme générale
  - `Connector.open("<protocol>:<path>:<parameters>");`
- HTTP
  - `Connector.open("http://www.codecamps.com");`
- Sockets
  - `Connector.open("socket://129.144.111.222:9000");`
- Ports séries
  - `Connector.open("comm:0;baudrate=9600");`
- Fichiers
  - `Connector.open("file:codecamps.dat");`

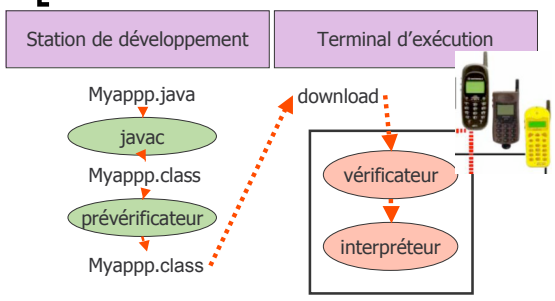
38

Sécurité

- Impossibilité d'implémenter le modèle de sécurité de J2SE
  - Sa taille est supérieur au CDLC
- Modèle de sécurité du CDLC
  - Niveau de la machine virtuelle
    - Une application s'exécutant sur la VM ne doit pas pouvoir causer de dommage au terminal
    - Nouveau Classfile vérificateur certifié
  - Niveau applicatif
    - Une partie seulement de l'API Java est disponible
    - Un programme ne peut pas surcharger certaines classes standards
    - L'accès aux fonctions natives n'est pas autorisé

39

Certification des classes



40

Télécharger le CLDC

- http://java.sun.com/product/cldc
  - Binaires et sources pour KVM et prévérificateur
  - Classes CDLC
  - Documentation
  - Exemples
  - Source pour JAM et Java technology Code Compact
  - Com.sun.kjava
    - bibliothèque non supportée : IHM, cycle de vie
- Les applications utilisent généralement
  - MIDP ou Wireless toolkit

41

MIDP

Mobile Information Device Profil

Pour récupérer des MIDlet
http://midlet.org

42

MIDP
Mobile Information Device Profile

- Le premier profil J2ME, JSR-0037 (1.0) puis JSR-0118 (2.0)
  - Les terminaux visés doivent supporter CLDC
- Objectifs
  - Toolkit de visualisation, input
  - Gestion de données persistantes
  - Messages
    - SMS, email
  - Sécurité
  - Connexion par réseau sans fils
- Sun distribue une version de référence
  - C'est celle que nous allons utiliser en TP ☺

43

Caractéristiques du terminal

- Affichage
  - Taille de l'écran : 96x54 pixels
  - Couleur : 1 bit (noir et blanc)
- Entrée
  - Mini : 1 clavier keypad
  - Possibilité d'avoir des claviers étendus : un ou deux 'clavier', écran tactile optionnel
- Mémoire
  - 128 kb non volatile pour MIDP composants
  - 8 kb non volatile pour les données persistantes
  - 32 kb volatile pour le runtime java (pile, tas, …)
- Réseau
  - Connexion full-duplex, intermittente, sans fils, débit faible

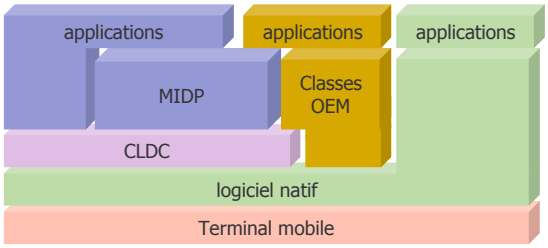
44

objectifs

- Définir une architecture et les APIs associées pour l'écriture d'application pour des terminaux mobiles
  - La simplicité est recherchée, parfois au détriment de la complétude
    - Affichage texte et graphique, orienté formulaire
    - Persistance dans des fichiers structurés en enregistrement
    - Connexion réseau : HttpConnection, UDP, SMS/MMS, ….
- En dehors des objectifs
  - Installation de l'application sur le terminal
  - Modèle de sécurité applicatif
  - Besoins spécifiques des applications
  - Les différents détails d'implémentation

45

architecture



46

Administration d'application

- Le MIDP permet la gestion des applications
  - En les adaptant aux caractéristiques des terminaux
  - En permettant l'installation, la suppression via un réseau sans fils ou un serveur web
  - En permettant l'évolution d'une MIDlet

47

Hypothèses

- MIDP fait les hypothèses suivantes
  - existence d'un noyau minimal qui gère le matériel avec au moins un fils d'exécution exécutant la machine virtuelle Java
  - existence d'un mécanisme permettant de lire et d'écrire en mémoire non-volatile
  - Capacité minimale pour utiliser un écran bitmap (même réduit)
  - Capacité permettant de lire et d'écrire via un périphérique sans fils
  - existence de dates pouvant être utilisées pour le stockage des données

48

Bibliothèques de classes

- Gestion du cycle de vie de l'application
  - Javax.microedition.midlet
- IHM
  - Javax.microedition.lcdui
- Gestion de la persistance
  - Javax.microedition.rms
- Accès réseau
  - Javax.microedition.io
- Langage et utilitaire
  - Java.lang
  - Java.util

49

Les classes du CLDC

- Java.lang
  - Version allégée du J2SE
    - - de classes, - de méthodes
- Java.util
  - Version allégée du J2SE
    - - de classes, - de méthodes
- Java.io
  - ByteArrayInputStream, ByteArrayOutputStream, DataInputStream, DataOutputStream, InputStream, InputStreamReader, OutputStream, OutputStreamWriter, PrintStream, Reader, Writer
- Javax.microedition.io
  - Connection, ContentConnection, Datagram, DatagramConnection, HttpConnection, InputConnection, OutputConnection, StreamConnection, StreamConnectionNotifier

50

Les classes de MIDP 1.0

- Javax.microedition.midlet
  - Cycle de vie d'une application
- Javax.microedition.lcdui
  - Interface orienté formulaire
    - Alert, AlertType, Canvas, ChoiceGroup, Command, DateField, Display, Displayable, Font, Form, Gauge, Graphics, Image, ImageItem, Item, List, Screen, StringItem, TextBox, TextField, Ticker
- Javax.microedition.rms
  - Orienté persistance
    - RecordStore, RecordEnumeration, RecordComparator, RecordFilter, RecordListener

51

Nouveauté MIDP 2.0

- Sécurité
  - https (au-dessus de TLS, SSL ou WAP WTLS)
  - PKI (certificats)
  - JAR signé
    - La signature et la clé publique sont ajouté au JAD
  - Permissions
    - Nouvelle entrée du JAD : MIDlet-Permissions et MIDlet-Permissions-Opt
- Push de MIDlet
- De nouvelles classes pour de nouveaux domaines
  - Javax.microedition.media
  - Javax.microedition.lcdui.game

52

Nouveauté MIDP 3.0

- Devrait sortir en septembre 2008
  - JSR00271

53

MIDP Application MIDlet

- Une application MIDP est une MIDlet
- Applications centrées sur l'IHM
  - L'écran est l'unité de dialogue
  - Un seul écran visible à un instant donné
  - Les évènements (clavier ou autre) concernent l'écran actif
- MIDlets
  - Ont un cycle de vie bien déterminé
  - Fournissent des informations sur elles-mêmes
  - Étendent javax.microedition.midlet.MIDlet
- MIDlets persistantes
  - Résident, au moins pour partie, dans la mémoire non-volatile (ROM ou EEPROM)
  - Peuvent être téléchargée et recopiée dans la mémoire persistante du MID
  - Une fois installée, elles peuvent exécutées plusieurs fois sans nouvelle installation

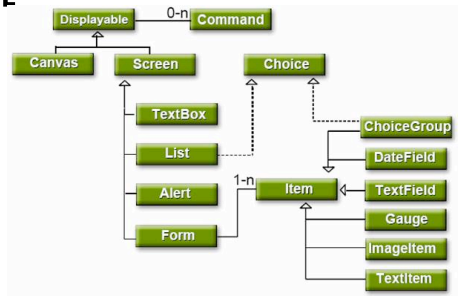
54

Interaction avec les utilisateurs

- Les MIDlets sont généralement des applications dialogants avec l'utilisateur
  - Celui-ci est pris en compte dès le début de la conception
- APIs définies dans javax.microedition.lcdui
- API de haut niveau, portable
  - Orientée écran et widget
  - Une application utilisant cette API pourra s'exécuter sur tous les terminaux
  - Contraintes : ne pas accéder directement aux attributs des terminaux (couleurs, tailles, entrée)
  - Plus simple et moins puissant que AWT
- API de bas-niveau
  - Primitives de dessin
  - Événements claviers
  - L'utilisation compromet la portabilité de l'application : à utiliser uniquement lorsque c'est nécessaire

55

Hiérarchie des classes pour les IHMs



Vous trouverez, dans le répertoire "apps" du Wireless Toolkit, un projet nommé "UIDemo" qui illustre, avec des exemples, l'utilité de chacun de ces composants.

56

Gestion de la persistence

- Permet uniquement l'enregistrement de structure (RMS)
- API indépendante des terminaux
- API définie dans **javax.microedition.rms**
- Un enregistrement est un tableau de bytes
- Les enregistrements de l'application peuvent être sauvegarder en mémoire
- Les enregistrements en mémoire peuvent être partagés entre les applications
- Support pour les énumérations et les ensembles (sorting, filtering)
- Mise à jour atomique des enregistrements simples

57

Entrée-sortie

- Implémente les spécifications du CLDC défini dans **javax.microedition.io**
- Peut ouvrir des connections HTTP (RFC2616)

58

Utilitaires MIDP

- Classes et interfaces très utiles
  - Implémente les spécifications du CLDC spécifiée dans **java.util**
    - Calendrier
    - Date
    - Enumération
    - Table de hachage
    - Tirage aléatoire
    - Pile
    - alarme (timer,timerTask)
    - vecteur

59

internationalisation

- Support des caractères ISO Latin 1 (ISO8859-1)
- Peut être adapté en fonction du terminal, défini dans microedition.locale
  - Chaque terminal doit fournir ses propres classes
  - Les applications sont constituées d'un ensemble de fichiers jar

60

Pour télécharger le MIDP

- http://java.sun.com/products.midp
  - Émulateur sous forme binaire (windows)
  - Classes et sources pour bibliothèques (java & C)
  - Documentation (pdf et javadoc)
  - Examples
- J2ME Wireless toolkit
  - http://java.sun.com/products/j2mewtoolkit

61

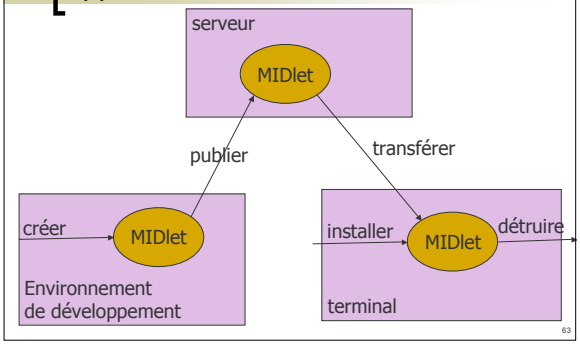
Cycle de vie d'une MIDlet

- Recherche de l'appli
- Installation
- Chargement
  - Création d'une instance
  - Initialisation
  - Terminaison
- Gestion de version
- Suppression par l'utilisateur

- Le logiciel d'administration
  - installe les MIDlets sur les terminaux
  - vérifie la sécurité
  - met à jour la version de l'appli si nécessaire
- MIDP défini les mécanismes nécessaire
  - Formats des Jar et Jad, MIDlet descripteurs dans **javax. microedition. midlet**

62

MIDP cycle de vie des applications



63

Transfert de l'application

- Le MID transfère la MIDlet depuis un serveur
- Le médium de transfert doit être identifier
  - Cable série
  - Port Infra Rouge(IrDA)
  - Réseau sans fils
- Une négociation à lieu entre le MID et le serveur
  - Caractéristiques du terminal
  - Taille de l'application
  - Coût du transfert
- Le transfert recopie la MIDlet dans la mémoire du terminal
  - Vérification de la sécurité
  - Transformation du code de la MIDlet du format public au format du terminal

64

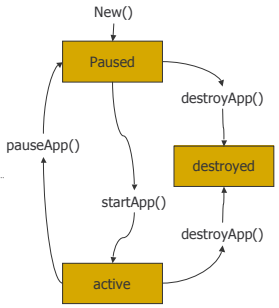
exécution d'une MIDlet

- Chargement
  - Installe la MIDlet dans la KVM
- Nouvelle version
  - Une nouvelle version d'une MIDlet est disponible dès son installation
  - Le logiciel d'administration contrôle les MIDlets installées et leur numéro de version
  - Cette information est utilisée pour changer de version, l'installation d'une nouvelle version provoque la désinstallation de la version précédente
  - Les attributs des MIDlets (versions, dates, etc.) sont contenus dans le descripteur (fichier .jad) et dans le manifest inclus dans le fichier .jar.
- Suppression d'une application
  - La suppression de l'image de la MIDlet provoque aussi, si possible la suppression des ressources liées et des données écrites en mémoire persistantes

65

Cycle d'exécution d'une MIDlet

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class HelloMIDlet extends MIDlet {

    public HelloMIDlet() {
    }

    public void startApp() {
    }

    public void pauseApp() {
    // On va suspendre l'exécution de la MIDlet…
    // Que faut-il faire
    }

    public void destroyApp() {
    // La MIDlet ne va plus être en exécution…
    // Que faut-il faire…
    }

}
```



66

MIDlet jar

Spécifié dans **javax.microedition.midlet**

- Un jar contient un manifest et peut contenir plusieurs MIDlets
  - Le jar contient pour chaque MIDlet les ressources nécessaires
    - Bitmap, son, etc
- Chaque fichier jar est accompagné d'un descripteur
  - Fichier .jad
  - Il permet de vérifier que la suite de MIDlet peut être téléchargée
  - Défini les attributs de configuration
  - Permet le transfert simultané de plusieurs MIDlet contenues dans le même fichier jar
  - Le type MIME d'un fichier .jad est
    - Text/vnd.sun.j2me.app-descriptor
  - Pour configurer un navigateur

```
Description: MIDP app descriptor MIDP app descriptor
MIME Type: text/vnd.sun.j2me.app-descriptor
Extension: jad
Application: d:\midp\midp.exe -transient file://%1
```

67

Exemple de descripteurs

- MIDlet-Name: SunSamples
- MIDlet-Version: 1.0
- MIDlet-Vendor: Sun Microsystems, Inc.
- MicroEdition-Profile: MIDP-1.0
- MicroEdition-Configuration: CLDC-1.0
- MIDlet- 1: Sokoban, /icons/Sokoban.gif, example.sokoban. Sokoban
- MIDlet- 2: Tickets, /icons/Auction.gif, TicketAuction
- MIDlet- 3: Colors, /icons/ColorChooser.gif, example.Color
- MIDlet- 4: Stock, /icons/Stock.gif, example.stock.StockMIDlet
- MIDlet- 5: Tiles, /icons/Tiles.gif, example.tiles.Tiles
- MIDlet- 6: ManyBalls, /icons/ManyBalls.gif, example.ManyBalls
- MIDlet- 7: Sampler, /icons/App.gif, Sampler
- MIDlet- 8: Properties, /icons/App.gif, example.PropExample
- MIDlet- 9: HttpTest, /icons/App.gif, example.HttpTest

68

Attributs des descripteurs

Attribute Name	File	Description
MIDlet-Name	*Req*	Name of the MIDlet suite.
MIDlet-Version	*Req*	Version number of the MIDlet suite.
MIDlet-Vendor	*Req*	Organization that provides the MIDlet suite.
MIDlet-Description	*Opt*	Description of this MIDlet suite.
MIDlet-Info-URL	*Opt*	URL for further information on the MIDlet suit
MIDlet-< n>		Name, icon, and class of the <n> th MIDlet in JAR file, separated by ",".
MIDlet-Jar-URL	*Req*	URL from which to download the JAR file.
MIDlet-Jar-Size	*Req*	Number of bytes in the JAR file.
MIDlet-Data-Size	*Opt*	Minimum number of bytes of persistent data required by the MIDlet.
MicroEdition-Profile		The required J2ME Profile.
MicroEdition-Configuration		The required J2ME Configuration.

69

Le paquetage javax.microedition.midlet (1)

- Défini les interfaces et les classes nécessaires
  - Pour écrire des MIDlet
  - Pour le dialogue MIDlet – machine d'exécution
- Classes abstraites pour les MIDlets
  - Permet le respect du cycle de vie précédemment défini

70

Le paquetage javax.microedition.midlet (2)

```
protected MIDlet()
        // constructeur sans paramètre

        // modifiable
        protected abstract void startApp() throws MIDletStateChangeException
                // appelée au démarrage de la MIDlet (paused -> active state)
        protected abstract void pauseApp()
                // appelée avant de suspendre la MIDlet (active -> paused state)
        protected abstract void destroyApp( boolean unconditional)
                throws MIDletStateChangeException
                // appelé avant d'arrêté l'exécution de la MIDlet (* -> removed state)

        // non modfiable
        // à appeler après l'appel à la primitive de changement d'état
        public final void notifyDestroyed()
                // demande au gestionnaire d'arrêté l'application
        public final void notifyPaused()
                // demande au gestionnaire de mettre en pause l'application
        public final String getAppProperty( String key )
                // demande de retourner des informations concernant l'application
```

71

C'est ma première MIDlet

- Ecrire le code Java
- Compiler le code Java
- Prévérifier le code
- Construire le fichier Jar
- Créer le descripteur de la MIDlet
- Exécuter la MIDlet sur l'émulateur (ou sur le terminal)

http://java.sun.com/products/j2mewtoolkit

72

Ecrire le code Java

```
import javax.microedition.midlet.*;
Import javax.microedition.lcdui.*;

public class HelloMIDlet extends MIDlet {
    private Display display;          // Display pour ma MIDlet
    public HelloMIDlet () {
        display = Display.getDisplay (this);
    }
    public void startApp() {
        TextBox t = new TextBox("Hello MIDlet", "Hello World ! It's
        my first MIDlet", 256, 0);
        display.setCurrent (t);
    }
    public void pauseApp() {
        System.out.println("In pauseApp…");
    }
    public void destroyApp( boolean unconditional) {
        System.out.println("In destroyApp…");
    }
}
```

73

Compiler et 'vérifier' le code

- javac –g:none
  –bootclasspath c:\midp-fcs\classes
  HelloMIDlet.java –d tmp
- preverify
  -classpath .;c:\midp-fcs\classes
  HelloMIDlet –d verified tmp
  - Le répertoire verified contient le code vérifié
  - Le répertoire tmp contient le code non vérifié

74

Construire le fichier Jar et le decripteur 'Hello.jad'

- jar cf Hello. jar *. class
  - Le fichier jar doit contenir toutes les ressources
- Le descripteur (fichier Hello.jad) doit contenir :
  - MIDlet-1: HelloMIDlet
  - MIDlet-Description: Ma première MIDlet
  - MIDlet-Jar-Size: 922
  - MIDlet-Jar-URL: Hello.jar
  - MIDlet-Name: FirstMIDlet
  - MIDlet-Vendor: Sun Microsystems
  - MIDlet-Version: 1.0

75

Excuter la MIDlet

- **midp**
  –classpath Hello.jar
  –descriptor Hello.Jad
- **Les paramètres de la commande**

```
-classpath <path>      // directories and zip files to search
-descriptor <jad file> //MIDlet app descriptor file to use
-transient <url> <app_ name>
                       // run app_ name from descriptor at url
-autotest <url> <app_ name>
                       // repeatedly run application
-verbose               // enable classloading information
-debug                 //enable debugging info
-help show this message
```

76

Variable d'exécution d'une MIDlet

Option	Défaut	Description
HTTP_PROXY	Null	URL sur serveur proxy
CLASSPATH		Classes utilisateurs
ENCODING	Null	Codage des caractères
DOUBLE_BUFFERING	True	Utilisation du double tampon pour l'affichage
SCREEN_DEPTH	1 (b/w)	2, 4, 8 pour 4, 16 ou 256 couleurs

77

Les différents émulateurs

- Ktoolbar
  - http://java.sun.com/products/j2menewtoolkit
- PalmOS 3.5
  - http://java.sun.com/products/midp4palm
- autres :
  - Motorola i1000 téléphone
    - http://www.idendev.com
  - Siemens

78

Une demo...

pour vous montrer que ca marche ☺
- en ligne de commande
- sur Palm OS
- avec ktoolbar

79

Développement MIDP

IHM
Persistance
Réseau

80

Principales bibliothèques

- UI (User Interface)
- Réseau : connexion HTTP
- Persistance : orientée enregistrement
- Dans : java.util
  - Classes: **Timer, TimerTask, Calendar, Date, HashTable, Random, Stack, TimeZone, Vector**
  - Interfaces : **Enumeration**
  - Exceptions : **EmptyStackException, NoSuchElementException**

81

Recommandations pour un code Java performant et compact

- Oublier les bonnes méthodes de programmation ☺
  - Mais continuer à écrire du code clair, commenté, lisible
  - Pensez à la maintenabilité et à la testabilité
- Si vous avez des problèmes de performances
  - Analysez (profiler) votre code pour identifier les goulot d'étranglement (bottlenecks)
  - Optimisez les goulots en conservant la qualité

82

Recommandations pour un code Java performant et compact

- Construire de petites applications, les plus simples possibles
  - Ofusquer le source
    - Mettre les méthodes en « private »
    - Compacter les identifiants de méthodes
      - http://www.alphaworks.ibm.com/tech/JAX
      - http://www.retrologic.com/retroguard-main/html
    - Dégraisser les jarfiles des classes non chargées
      - Outils calculant la fermeture transitive du graphe de chargement
  - JAR non compressé pour accélérer le démarrage
    - Décompression
    - Mapping mémoire virtuelle ou Flah ou ROM
  - Simplifier votre application

83

Recommandations pour un code Java performant et compact

- Internationalisation à l'installation
  - Une version de l'application (JAR) par langue, pays, …
  - Précompilation avec des #ifdef avec cpp (ou ses versions Java)

84

Economie de mémoire à l'exécution

- Utiliser les types scalaires plutôt que des objets
  - void setSize(int width, int height);
  - void setSize(Dimension size);
- Aider le Ramasse-Miette
  - Mettre les références qui ne seront plus utilisées à null
  - Reutiliser les objets instanciés
    - Pool d'objets, …
  - Prévoyer la capacité initiale des collections
    - Constructeur Vector (int initialCapacity, int capacityIncrement)
  - Instancier à la demande (lazy ou on-demand)
    - Public class LazyClass {
      private Vector v;
      public Vector getVector() {
      if (v==null) v=new Vector();
      return v;
      } }

85

Economie de mémoire à l'exécution

- Evitez la récursion
  - Élégant mais lent et gourmant en pile
  - Dérécusivez quand c'est possible
- Evitez les exceptions
  - Car les exceptions sont des objets et leur déclenchement augmente la taille du code

86

Code performant

- Utiliser des variables locales
  - for(i=0;i<buf.length;i++) { if(buf[i]>='0' && buf[i]<='9' && …) }
    for(i=0;i<buf.length;i++) {Char c=buf[i]; if(ch>='0' && ch<='9' && …) }
- Eviter la concatenation de chaines et utiliser StringBuffer
- Utiliser les threads mais limiter la synchronisation
  - public synchronsized Vector getVector() {
    // peu performant
    if (v==null) v=new Vector();
    return v;
    }
  - public Vector getVector() {
    // synchronisation circonscrite
    if(v==null) {
    synchronized(this) { **if(v==null)** v=new Vector();}
    }
    return v;
    }
- Eviter de faire des appels à chaque boucle
  - A chaque optimisation on gagne
    - For (**int** i=0; i<**s.length()**; i++) { … }
    - Int len = **s.length()**;
      for (**int** i=0; i<s; i++) { … }
    - Int len = **s.length()**;
      while (**len** > 0)

87

Graphisme

- Ne charger que les images dont vous avez besoin
  - Pour MIDP utilisez :
    - Image createImage(String ressource)
- Adapter la taille / nb couleur des images
- Utiliser le clipping et les images Buffer pour les animations
  - En programmation graphique 3D, la méthode du ***clipping*** consiste à ne pas calculer les objets extérieurs au cône de vision d'une scène afin d'optimiser le temps de calcul.
  - Eviter le scintillement (flicking)

88

Utilisez les outils d'analyse

- Déboggage
- Profilage
- Analyse de la pile d'appel et du tas
  - Fuite mémoire, …

89

Le résumé du cours

- Comprendre les principaux aspects de la spécification J2ME :
  - Machine virtuelle : KVM
  - Configurations : CLDC
    - Machine virtuelle et biblikothèque Java minimales pour une large gamme de terminaux
    - Risque de s'aligner le terminal le moins perfectionné de la gamme…
  - Profils : MIDP
    - fournit un environnement complet (API) pour un ensemble spécifique de terminaux
      - javax.microedition.midlet (cycle de vie)
      - javax.microedition.lcdui (IHM)
      - javax.microedition.rms (persistance)
      - javax.microedition.io (entrée sortie)
      - java.lang and java.util
- Présenter la chaîne de développement et quelques émulateurs

90

Quelques ressources

- **J2ME Codecamp:** www.sun.com/developers/edu/camps
- http://java.sun.com/j2me
- J2ME site: http://java.sun.com/products/j2me
- J2ME Wireless Toolkit: http://java.sun.com/products/j2mewtoolkit
- http://www.billday.com/j2me
- Pour s'abonner à la liste **kvm-interest**
  mail listserv@java.sun.com
  subscribe kvm-interest
- Archives de la liste : http://archives.java.sun.com
  http://www.kvmworld.com
- Site JCP: http://java.sun.com/jcp

91

Bibliographie

- C'est un domaine encore peu stabilisé…

- Eric Giguère, Java 2 Micro Edition, Ed Wiley, 2000, ISBN 0-471-39065-8
- Yu Feng, Dr. Jun Zhu, Wireless Java Programming with Java 2 Micro, 1st edition (May 24, 2001), Ed Sams; ISBN: 0672321351
- Wong, Java 2 Micro Edition, Ed Addison Wesley. ISBN 0-201-70244-4 (06/2001)

92