



Utilizzo di applicazioni Java nella Telefonia mobile

Dott. Guerriero Raimato

Dip. Informatica ed Applicazioni "R. M. Capocelli"
Università degli Studi di Salerno

J2ME: Introduction



- Primo prototipo *Oak*, 1990
- Home Controller *Star7*
 - *Handuser*
 - *Touchscreen LCD*
 - *Wireless networking integrata*
 - *Porta infrarossi*
 - *flash RAM file system*

Introduzione



- Java 2 Platform, Micro Edition (J2ME)
- Configuration
 - CLDC
- Profiles
 - MIDP
- Midlet
- Networking
 - Http connection
 - Bluetooth
- Applicazioni
 - Emulatore
 - Web Service

Oak Team developers



- Al Frazier, Joe Palrang, Mike Sheridan, Ed Frank, Don Jackson, Faye Baxter, Patrick Naughton, Chris Warth, James Gosling, Bob Weisblatt, David Lavalley and Jon Payne. Missing in action: Cindy Long, Chuck Clanton, Sheueling Chang and Craig Forrest.



Star 7



- Remote controller
 - TV
 - VCR
- Agenda Elettronica
- Gestione delle rubriche
- Precursore degli attuali PDA



Star 7 : Flop

- Prodotto
 - Troppo innovativo
 - Troppo costoso
 - Gravi problemi di stabilità
 - Difficoltà di inserimento sul mercato
 - Diffidenza da parte degli investitori
 - Rapporto qualità prezzo troppo svantaggioso



Star 7: Problems

- Potenza di calcolo limitata
- Memoria limitata
- Errori di programmazione
 - Puntatori
 - Ricorsione
 - Ecc.



Anni '90: nuovi sbocchi

- Espansione di Internet
- Nuovi browser Web
 - **WorldWideWeb**. [Tim Berners-Lee](#)
 - **libwww**. Berners-Lee and Jean-Francois Groff C language in 1991 and 1992
 - **Line-mode**. Nicola Pellow
 - **Erwise**. Un gruppo di studenti della Helsinki University of Technology
 - **ViolaWWW**. Pei Wei. Browser grafico.
 - **Midas**. Estate 1992. Tony Johnson
 - **Samba**. Robert Cailliau sviluppo del primo web browser Macintosh
 - **Alibatic**. Marc Andreessen e Eric Bina
 - **Arena**. In 1993, Dave Raggett, Hewlett-Packard lab
 - **Lynx**. The University of Kansas
 - **Collo**. Tom Bruce
 - **Opera**. Nel 1994 nasce Opera
 - **Internet in a box**. Gennaio 1994, 1994. O'Reilly and Associates
 - **Navipress**. Febbraio, 1994
 - **Mozilla**. Ottobre, 1994, Netscape rilascia Mozilla 0.96b
 - **Internet Explorer**. Agosto, 1995
- Nascita di nuovi paradigmi di comunicazione
- Introduzione di nuovi mercati commerciali
 - Internet commerce



Il browser dinamico: HOT-JAVA



Sotto la spinta del nuovo mercato la Sun trasformò:

- Oak in un prodotto cross-platform browser
- HOTJAVA
 - September 29 1994



Java milestones



- **Autumn, 1994**
 - Van Hoff implementa un compilatore Java *in* Java.
- **May 23, 1995**
 - La Sun annuncia formalmente Java and HotJava al SunWorld '95.
- **May 23, 1995**
 - Netscape annuncia l'intenzione di integrare Java in new web browser Netscape.
- **September 21, 1995**
 - Sun sponsorizza a New York una conferenza per sviluppatori Java
- **September 25, 1995**
 - Sun annuncia l'alleanza con la Toshiba
- **October 30, 1995**
 - Oracle annuncia l'integrazione di java nei suoi software
- **October 30, 1995**
 - Lotus Development Corp., Intuit Inc., Borland International Inc., Macromedia Inc., and Spyllglass Inc. annunciano l'integrazione di java nei proprio prodotti

Hot Java: Caratteristiche



- Capacità Cross-platform
- Innovazione con Java Applets
- Nuove Potenzialità di free-standing applications, "Write once run Anywhere"
- Riduzione dei paradigmi di programmazione pericolosi
- Maggiore facilità di sviluppo

Java milestones-2



- **December 4, 1995**
 - Sun and Netscape annunciano il lancio di un nuovo linguaggio di scripting, Javascript
- **December 4, 1995**
 - Sun, Netscape and Silicon Graphics annunciano la loro intenzione di produrre tools per l'interazione con la rete
- **December 4, 1995**
 - Borland, Mitsubishi Electronics, Sybase and Symatec annunciano il loro accordo con Java
- **December 6, 1995**
 - IBM e Adobe si accordano con la Sun per l'uso di Java
- **December 7, 1995**
 - Microsoft si accorda con la Sun e lancia un nuovo prodotto per la gestione degli script in Visual Basic

Java



- Java suscitò l'interesse di:
 - Netscape
 - Silicon Graphycs
 - Borland
 - Mitsubishi Electronics
 - Sybase and Symatec
 - IBM
 - Adobe
 - Microsoft

Java 2



Con l'introduzione di nuovi package il core si divide in:

- J2SE
 - Applicazioni "standard"
 - Infrastruttura di comunicazione
 - Commercio elettronico
- J2EE
 - Application server
 - Servlet
 - Enterprise Java Beans
 - JavaServer pages
- J2ME
 - Small devices

Java 2 platform



La SUN nel 1996 rilasciò il primo prodotto per il pubblico:

Durante JavaOne, la prima conferenza per gli sviluppatori Java venne rilasciato JDKtm 1.0

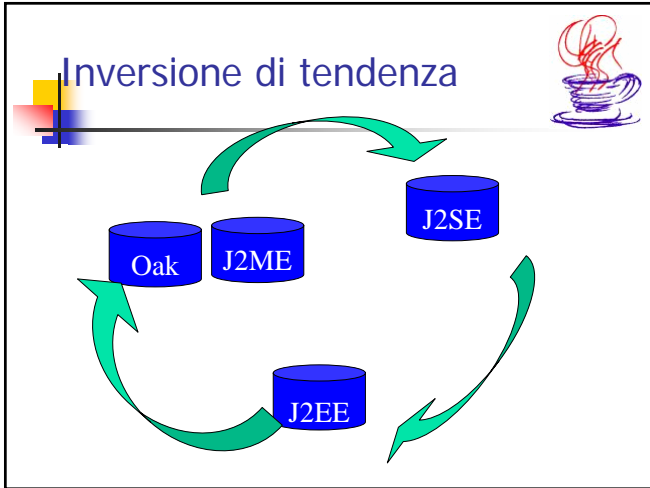
Nel 1997:
Over 220,000 downloads of JDK 1.1

J2ME



Inversione di tendenza:

- Cambiano le politiche di mercato
 - Richiesta di Java su "piccoli device"
 - Smart card
 - Lettori ottici
 - Mobile devices



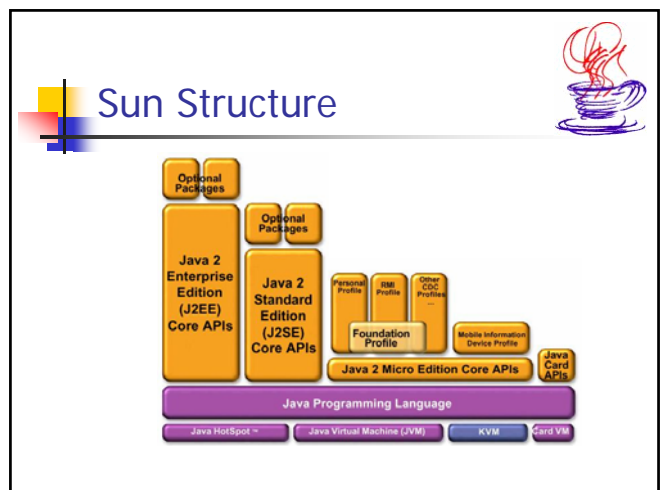
J2ME vs Java "Standard"

Differenti politiche:

- RISC
 - Reduced Instruction Set Computer
 - J2ME
- CISC
 - Complex Instruction Set Computer
 - J2SE/J2EE

J2ME

- J2ME
 - Platform for small devices
 - Interfaccia per numerosi dispositivi differenti
 - Collezione di specifiche
 - Software "cangiante"
 - Il software si modella sul dispositivo



Configuration and Profiles



- Configurazione
 - Stabiliscono le caratteristiche della VM
 - Definisce l'ambiente software per dispositivi con caratteristiche simili
 - Tipo e quantità di memoria disponibile
 - Tipo e velocità del processore
 - Tipo di connessione disponibile
 - Rappresenta la configurazione minima per il device

Configuration and J2ME



- Seconda configurazione di base in J2ME
- **C**onnecte**D**evice **C**onfiguration
 - Microprocessori a 32-bit
 - Dotazione di memoria notevoli (16–32 MB)
 - Tipicamente
 - PDA high-end
 - Dispositivi per ordini commerciali
 - Web-TV phone

Configuration and J2ME



Prima configurazione di base in J2ME:

- **C**onnecte**L**imited **D**evice **C**onfiguration
 - Bassa potenza di calcolo e comunicazione
 - Poca memoria, normalmente 512 kb
- Tipicamente
 - PDA low-end
 - Smartphone low-end

Configuration



- Ciascuna configurazione è costituita da:
- Una Java virtual machine
 - Un core di classi java
- Ciascuna configurazione fornisce:
- Un ambiente di programmazione
 - Una infrastruttura per applicazioni software

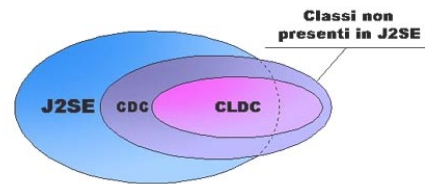
Configuration and J2ME



CLDC:

- Limitate capacità di comunicazione
- Programmazione basata su MidLet
- Possibilità di creare proprie applicazioni
- Possibilità di firmare le applicazioni
- Si basa sulla famosa KVM

CDC - CLDC



CDC



CDC:

- A cavallo tra CDCL e ambienti J2SE
- Maggiore dimensione della memoria (2 Mb)
- Maggiore potenza di calcolo
- Possibilità di supportare ambienti di sviluppo più completi
- i.e. PDA high-end, web telephone ecc.

Configuration



Importante, notare che:

- Grazie alle configurazioni si amplia il mondo delle VM
- Ogni vendor può creare la sua VM
- Ogni vendor può utilizzare la VM di una third-party
- Le VM vengono standardizzate

Configuration-2



Ogni Vendor può creare una propria VM

Tale Virtual Machine deve però soddisfare i requisiti minimi delle specifiche SUN

CLDC



- È il blocco di base su cui sono costruiti i *Profiles* J2ME per piccoli dispositivi
- "Hello world", in J2SE richiede circa 5 Mb di memoria
- Dispositivi mobili hanno
 - 128 KB di ROM
 - 32 KB di memoria disponibile

Why configuration??



"Superare" le limitazioni di:

- Memoria
 - Processore
 - Mediare tra velocità e complessità
 - Fornire un ambiente completo e standard
- Non si possono fare miracoli
- Non sono supportati tutti gli aspetti della J2SE VM
 - Il byte code è limitato
 - Le ottimizzazioni del codice non sono supportate

CLDC: Solution



Per sopperire alla carenza di memoria e potenza:

CLDC limita:

- Le richieste della virtual machine
- Il linguaggio
- L'insieme delle librerie
- No specific input assumption

CLDC



Device host ha le stesse caratteristiche di un OS

- Poche assunzioni sui dispositivi IO
- Possibilità di eseguire virtual machine
- Manage della virtual machine
 - La virtual machine può essere multithread
 - Il file system può non essere multithread
 - Il file system può supportare semplici algoritmi di scheduling

CLDC



Molti processori sviluppati in accordo con CLDC:

- Non hanno supporto per il floating point

Dadd	dload	dsub	fcmpl	frem	i2d
Daload	dload_x	d2f	fconst_0	freturn	i2f
dastore	dmul	d2i	fconst_1	fstore	i2d
dcmpl	dneg	d2l	fdiv	fstore_x	i2f
dcmpl	drem	fadd	float	fsub	newarray (double)
dconst_0	dreturn	faload	float_x	f2d	newarray (float)
dconst_1	dstore	fastore	fmul	f2i	
ddiv	dstore_x	fcmpl	fneg	f2l	

Nessuno vieta di implementare il floating point via software

VM and Language features



- CLDC specifica:
 - Caratteristiche che la VM deve descrivere
- CLDC evidenza:
 - Tutte le specifiche della VM
 - Specifiche non supportabili
 - Specifiche non richieste
 - Le limitazioni
- CLDC Sun:
 - Basato su una versione KVM (Kilobyte Virtual Machine)

No Floating Point



- Variabili di tipo float double non possono essere dichiarate
- Costanti (i.e. 1.0, 2.0F) non possono essere usate
- Le firme dei metodi non possono contenere argomenti float e double
- Oggetti e tipi Float e Double non possono essere istanziati

CLDC: Reflection



- No java.lang.reflect package
 - Constructor getConstructor(Class[] params)
 - Constructor[] getConstructors() ...
- No Weak Reference
- No Java Native Interface
 - Non è possibile il link dinamico al native code
 - Non si può accedere direttamente alle funzionalità della device
 - Si posso estendere le API, con un pre-linking di funzionalità native aggiuntive direttamente nella VM
 - Possibile solo con una custom VM

CLDC: Threading



- I Threads sono complessi da gestire ma indispensabili anche nella CLDC.
- CLDC thread limitation
 - No daemon thread
 - No group thread

CLDC: Object Finalization



Object Finalization

- Effettuare operazioni durante la cancellazione di un oggetto crea grande complessità nella VM e pochi benefici. Non viene implementata nella CLDC
- Nota: java.lang.Object class non ha il metodo finalize()

CLDC: Exception



- Nell'ambiente J2SE c'è una grande varietà di eccezioni possibili:
 - AclNotFoundException
 - ActivationException
 - AlreadyBoundException,
 - ApplicationException, AWTException, BackingStoreException, BadLocationException, CertificateException, ClassNotFoundException, CloneNotSupportedException, DataFormatException, DestroyFailedException, ExpandVetoException, FontFormatException, GeneralSecurityException, GSSException, IllegalAccessException, InstantiationException, InterruptedException, IntrospectionException, InvalidMidiDataException, InvalidPreferencesFormatException, InvocationTargetException, IOException, LastOwnerException, LineUnavailableException, MidiUnavailableException, MimeTypeParseException, NamingException, NoninvertibleTransformException, NoSuchFieldException, NoSuchMethodException, NotBoundException, NotOwnerException, ParseException, ParserConfigurationException, PrinterException, PrintException, PrivilegedActionException, PropertyVetoException, RefreshFailedException, RemarshalException, RuntimeException, SAXException, ServerNotActiveException, SQLException, TooManyListenersException, TransformerException, UnsupportedAudioFileException, UnsupportedCallbackException, UnsupportedFlavorException, UnsupportedLookAndFeelException, URISyntaxException, UserException, XAException

J2ME: Exception CLDC



Si ha a disposizione:

- `Java.lang.error`
- `Java.lang.OutOfMemoryError`
- `Java.lang.VirtualMachineError`

J2ME: Class Loading



- CLDC non permette l'introduzione di meccanismi arbitrati di loading delle classi
- CLDC Specifica:
 - Tutte le VM devono poter caricare application package in formato JAR
 - Il device può trasformare le applicazioni in qualsiasi formato supportato
 - PalmOs accetta applicazioni in formato JAR e le trasforma in un formato PRC proprietario

CLDC: Exception



Nell'ambiente CLDC

- Molte eccezioni non vengono supportate
- Il device è responsabile dell'invocazione
- Il device decide la giusta azione
- Il device fornisce le giuste informazioni all'applicazione

J2ME: Sicurezza



Aspetto delicatissimo è quello della sicurezza:

- Definita per livelli
 - Una applicazione "one-end" ha permessi illimitati
 - unrestricted
 - Una applicazione scaricata da un sito untrusted lavora in un ambiente limitato
- Come una applicazione viene definita sicura??
 - Tramite il certificato
 - Oppure tramite una firma digitale del software

J2ME: Sicurezza (2)



- Gestita mediante privilegi
 - Accesso alle risorse di rete
 - Accesso ai dati dell'utente in lettura/Scrittura
 - Accesso per l'avvio automatico
 - Accesso ai dati multimediali
 - Accesso alla sezione messaggi

J2Se: class verification



- J2SE fornisce sempre una verifica a livello byte-code della classe
 - Tutte le variabili sono state inizializzate
 - Un oggetto è stato creato prima di essere usato
 - Ciascun costruttore deve invocare prima il costruttore della super-classe (eccetto java.lang.Object)
 - Se gli assegnamenti sono fatti verso oggetti compatibili Integer -> char
- Questo lavoro è fatto per tutte le classi che sono state caricate dalla rete ma non per quelle caricate dal file system.

J2ME: Sicurezza (3)



- La J2ME in ambito di sicurezza fornisce:
- Device non permette all'utente l'installazione di software
 - Device permette l'installazione di qualsiasi software
 - Device fornisce politiche per l'installazione del software
- Sfortunatamente il device target della CLDC non permette la gestione di una *fine-grained security policy*

J2ME: Verification



- Per i dispositivi mobili sarebbe opportuno fare il lavoro di verifica su tutte le classi caricate
- Problema, troppo carico computazionale così:
- Preverification
 - È fatta sui file prima di installarli sul device, tipicamente in fase di compilazione
 - Viene generato un file che verrà incluso nell'applicativo
 - RunTime
 - Dipende dalla natura del dispositivo
 - Durante il loading della classe
 - E può usare il file creato durante la pre-verifica

CLDC: Other Restriction



- java.lang
 - CLDC circa la metà delle classi J2SE
- Java.lang.object
 - Non supporta finalize()
 - Non supporta clone()
 - Non esiste quindi una maniera standard per clonare un oggetto

CLDC: java.microedition.io



Definisce:

- Un framework Generico di connessione

Fornisce:

- Meccanismo standard per la gestione dell'accesso ai dati locali e/o remoti

Infrastruttura:

- Insieme di interfacce

J2ME: java.io



Per IO sono disponibili:

- Input e output stream verso e da sorgenti "reali" (SIM, memorie)
- Si possono leggere e scrivere dati
 - Come array di byte,
 - ByteArrayInputStream/ByteArrayOutputStream
 - Come oggetti primitivi
 - DataInputStream/DataOutputStream

Threads



Tutte le features:

- Wait(), notify(), notifyAll() sono incluse

Non sono incluse:

- Tutti i costruttori di ThreadGroup
- Non vi sono setName(), getName()
- Resume(), suspend(), stop()
- Destroy(), interrupt(), isInterrupted()
- dumpStack()

J2ME: java.util



Contiene una collezione di classi:

- Date
- Time
- HashTable
- Stack
- Vector
- Enumeration

Profiles



La gestione dei dispositivi basata solo sulle configurazioni è troppo limitata:

Profiles

- Si basano su una configurazione
- Estendono le funzionalità di base
- Forniscono API aggiuntive
- Strumenti di sviluppo per varie categorie di dispositivi

Profili

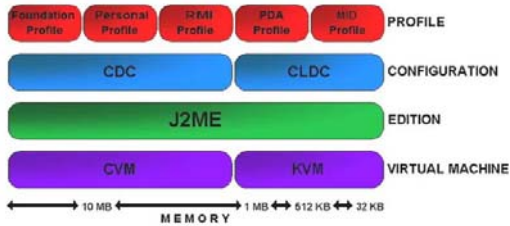
Profiles

Profile



- Profiles
 - Un set di classi
 - Utilizzabili su un insieme di dispositivi
 - SmartPhone
 - Two way pagers
 - PDA
 - KVM environment

Profile – Configuration - VM



MIDP

Il software che implementa le MIDP

- È eseguito nella KVM
- Fornisce servizi aggiuntivi
- Amplia la potenza delle applicazioni
- Usa le MIDP Apis
- Le applicazioni, "Midlets"

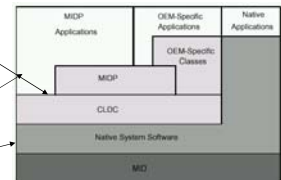
MIDP

Mobile Information Device Profile:

- Sviluppate da Java Community Process
 - <http://jcp.org/en/home/index>
- Disponibile per il download
 - <http://jcp.org/en/jsr/detail?id=37>

MidLets

Accedono direttamente a CLDC



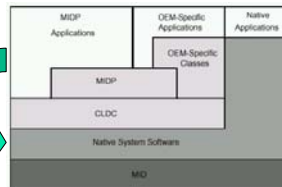
Non accedono direttamente alla piattaforma

Possono accedervi direttamente ma perdendo la portabilità

Midlets



L'unico modo per accedere **direttamente** alla piattaforma è tramite una versione custom della VM
JNI non supportato



MidLets



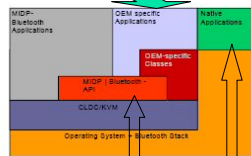
La struttura delle API è complessa a causa dello scenario estremamente variegato e polimorfico dei vari dispositivi su cui ci si deve basare. Quindi c'è la necessità di adattare

Midlets



OEM-Code: Codice Specifico

- Installazione
- Modifica
- Rimozione



Utilizzano contemporaneamente interazioni

- MIDP
- Software Nativo

MIDP hardware: Memoria



- MIDP richiede
 - Almeno 128 KB
 - Implementazione
 - Almeno 32 KB
 - Java Heap
 - Almeno 8KB (optional)
 - Dati persistenti
(le politiche di gestione di questi dati sono "libere")

MIDP hardware: Display



I device MIDP sono caratterizzati:

- Piccoli display
 - 96 x 54 pixel (larghezza, altezza)
- Pixel
 - Inizialmente bianco/nero
 - Adesso a colori

SUN MIDP



Sun MIDP:

- Non è un prodotto commerciale
- È un insieme di specifiche
- I produttori di device devono sviluppare
 - In accordo con le specifiche MIDP per ridurre il gap tra le specifiche SUN e hardware/software dei produttori stessi

MIDP Hardware: Input Device



MIDP Input:

- SmartPhone
 - Tastiera
- Palm OS
 - Handheld
- PDA
 - Input device "Graffiti"
- MIDP base:
 - Numeri da [0 ... 9], caratteri alfanumerici ...

MIDP assumption



Un vendor deve fornire un device

- Permetta l'esecuzione in un ambiente protetto
- Fornire l'illusione del multithreading
- Fornire l'accesso a quelli che sono i dispositivi di input (display, memorie, SIM)
- Fornire l'accesso ai dispositivi di output (display, memorie, SIM)
- Fornire una metodologia per la persistenza dei dati (switch off)
- Fornire un sistema di Networking

MIDlets



Le applicazioni eseguite su device MIDP sono dette MIDlets

- Una classe Java derivata da `javax.microedition.midlet.MIDlet`
- È eseguita all'interno di una JVM
- Ha un ciclo di vita ben definito definito da metodi nella classe `midlet`
- Un medesimo ambiente può essere condiviso da più midlet raggruppate in una *MidLet suite*

Esempio di condivisione



```
Public class ContoCorrente{
    private static int NumBankAccount;
    public static synchronized void NewBankAccount(){
        this.NumBankAccount++;
    }
    public static synchronized int GetNumBankAccount(){
        return this.NumBankAccount;
    }
}
```

La variabile `NumBankAccount` è condivisa tra tutte le classi della suite

MidLet Suite



Una Midlet Suite:

- Tutte le midlet all'interno della midlet suite sono installate e gestite come una singola identità
- Al runtime tutte le midlet sono eseguite nella stessa JVM
- Condividono le stesse istanze delle classi
- Condividono le stesse risorse
- Si possono usare le primitive Java di sincronizzazione per evitare accesso a dati inconsistenti
- Possibilità di condivisione di dati persistenti nella stessa suite
- Non è possibile condividere dati tra varie suite, a causa del meccanismo di *naming* delle classi

Midlet Package



Midlet problemi:

- Singola classe Java
 - Codice poco strutturato
 - Difficoltà di leggibilità
 - Perdita di "information hiding"
 - Necessità di ricompilare tutto il codice
 - Difficoltà di testing

Midlet Package: Soluzione



Le midlet supportano un sistema di package:

- Tutte le classi di supporto
- Le immagini
- I file

Possono essere racchiusi in un JAR file, insieme alla midlet

Attributi midlet



Attribute Name	JAR	JAD	Value and Meaning
MIDlet-Name	M	M	The name of the MIDlet suite packaged in the JAR file. This name may be displayed to the user.
MIDlet-Version	M	M	The version number of the MIDlet suite packaged in the JAR file. Version numbers take the form a.b.c (for example 1.2.3), where larger values in each field indicate a newer version, with the leftmost field taking precedence. For example, version 1.2.5 is taken to be more recent than version 1.2.3, and, similarly, version 2.1.5 is newer than 1.3.7.
MIDlet-Vendor	M	M	The name of the MIDlet suite provider. This is free-form text that is intended for display to the user.
MIDlet-n	M	I	Attributes that describe the MIDlet in the MIDlet suite. The value <i>n</i> is replaced by a numeric value starting from 1 to identify individual MIDlets. The format of the value associated with this attribute is described in the text.
MicroEdition-Profile	M	I	The version or versions of the MIDP specification that the MIDlets in this suite can work with. Where more than one version appears, they must be separated by spaces. The versions specified are compared to those listed in the microedition profiles property of the target device to determine whether the MIDlets are compatible with them. MIDP-1.0 is a typical value for this attribute.

Midlet Package



Il JAR file contiene

- "l'occorrente" per il funzionamento della midlet
- Jar file manifest
- Jar file manifest oppure JAD, *Java application descriptor*
- Sia il Jar Manifest che il JAD sono semplici file di testo
 - Attribute-name: attribute-value

JAD



- Velocizzare le operazioni preliminari
 - Taglia Jar File 50Kb
 - Taglia JAD file 50 bytes
- Fornire informazioni all'utente
 - Vendor
 - Versione del software
 - CLDC
- Fornire informazioni al dispositivo
 - Versione CLDC utilizzata
 - Permessi
 - Profilo utilizzato

Esempio JAD file

- MIDlet-1: CityGuide,,examples.cityguide.CityGuideMIDlet
- MIDlet-Jar-Size: 51229 (Non presente nel manifest)
- MIDlet-Jar-URL: CityGuide.jar (Non presente nel manifest)
- MIDlet-Name: CityGuide
- MIDlet-Permissions:
javax.microedition.location.LandmarkStore.read,javax.microedition.location.LandmarkStore.write,javax.microedition.location.LandmarkStore.category,javax.microedition.location.LandmarkStore.management,javax.microedition.location.Location,javax.microedition.location.ProximityListener
- MIDlet-Vendor: SUN Microsystems
- MIDlet-Version: 1.0
- Manifest-Version: 1.0
- MicroEdition-Configuration: CLDC-1.1
- MicroEdition-Profile: MIDP-2.0

Una midlet

```
public class ClientMIDlet extends MIDlet{  
    public ClientMIDlet() {}  
  
    protected void startApp() throws MIDletStateChangeException { }  
  
    protected void pauseApp() {}  
  
    protected void destroyApp(boolean arg0) throws  
        MIDletStateChangeException{}  
}
```

MidLet Lifecycle

Una midlet è derivata dalla classe astratta
Javax.microedition.midlet.MIDlet

- Deve avere un costruttore di default
 - Definito dall'utente
 - Creato automaticamente da java
- Deve implementare i metodi
 - Start
 - Pause
 - Destroy

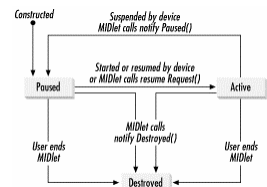
Stato di una Midlet

All'atto del caricamento di una midlet:

- Pause

Il caricamento di una midlet è identico a quello di una qualsiasi classe java:

- la memoria è allocata
- la classe viene caricata
- il costruttore è invocato



- Se in questa fase vi è una eccezione la midlet viene distrutta
- Altrimenti viene schedulata per l'esecuzione e passa nello stato Active

Midlet: Active



- Il metodo `startApp()` viene invocato
 - `Protected void startApp() throws MIDletStateChangeException`
- Il metodo è protetto
- Il metodo deve essere implementato
- È invocato automaticamente da un'altra classe, `Schedule`
- La classe `schedule` può essere ridefinita da qualsiasi vendor, a patto di non intaccare il ciclo di vita della midlet

Midlet Activation



Un midlet può essere "Attivata":

- User Event
 - Pressione di un tasto
 - Puntatore sullo screen
- No user Event
 - Thread in background

StartApp(): problems



- Quando viene avviata la midlet:
- Errore transiente
 - Problema che non esisterà in futuro
 - `MIDletStateChangeException`, la midlet torna in uno stato di Pause e verrà riavviata in seguito
 - Errore non transiente
 - Problema non risolvibile nemmeno in futuro
 - Viene invocato il metodo `notifyDestroyed()`
 - Fatal error
 - Problema grave
 - Viene lanciata una eccezione non catturata da `startApp`, il sistema invoca `destroyApp()`

Midlet: to Pause



From Active to Pause:

Protected abstract void pauseApp()

- Deve essere implementato
- Può essere invocato in qualsiasi momento
- Deve rilasciare ogni risorsa
- Deve salvare lo stato attuale dell'applicazione

Pause to Active



- Il metodo startApp() è invocato
- Implementare StartApp() in double way
- Evitare risorse allocate più volte
- Ricreare lo stato precedente all'interruzione
- Deve riabilitare lo screen per l'utente
- Gestire la persistenza dei dati

Costruttore != startApp()



- Il costruttore non è uguale a startApp()
- La midlet dovrebbe poter accedere al Display
- Si può accedere al display solo dopo aver invocato lo startApp() method
- Se si cerca di accedere al Display dal costruttore si genera una eccezione

Inizializzazione dove??



- Il metodo StartApp() può essere invocato più volte
- Il costruttore solo una volta
- Lo sviluppatore può scegliere dove inizializzare la classe
 - Nel costruttore, quando si necessita di una sola inizializzazione
 - Nello startApp(), quando ad ogni ripristino si necessita di una inizializzazione

Destroy



- Rilascia tutte le risorse
 - Background threads
 - Timers
- Se termina in modalità unconditional (true)
 - Allora la midlet termina
- Se termina in modalità conditional
 - Allora la midlet cerca di continuare lanciando una eccezione MIDletStateChangeException

Self destroy



Ci sono diversi modi per una midlet di terminare

- User destroy
 - Il codice utente invoca il metodo `destroyApp()`
 - Parametro, True
- Self Destroy
 - La midlet invoca il metodo `notifyDestroyed()`
 - Non viene invocato il metodo `destroyApp()`
 - È l'unico metodo per l'applicazione per terminare
 - Non è possibile usare `System`, `RunTime exit()` perché lanciano una eccezione `SecurityException`

Sviluppo di una Midlet



- Strumenti
 - J2sdk
 - J2Me
 - Wireless ToolKit
 - Strumento di sviluppo freeware
 - Multiplatform
 - Target, smartphone e PDA
 - Fornisce ambiente di emulazione

Notifier



Public final void notifyPaused()

- Comunica l'intenzione di passare in uno stato di pause
- Non viene invocato il metodo `pauseApp()`

Public final void resumeRequest()

- Comunica di passare da uno stato pause ad uno Active (thread in background oppure timer)
- Viene invocato il metodo `startApp()`

Java Wireless ToolKit



Fornisce:

- Ambiente di emulazione
- Ambiente di testing
- Emulazione di connessioni
 - Gprs
 - Wireless
 - Bluetooth
- Capacità di interazione con connessioni TCP/IP

Utilizzo di WTK



- Download da
 - <http://java.sun.com/products/sjwtoolkit/download.html>
 - Versione stabile attuale 2.2, 2.3 beta
 - Bastato su MIDP 2.0
 - Supporta CLDC 1.1, 1.2
 - Non è un ambiente di sviluppo

WTK: Nuova midlet



- Nome del progetto
- Nome della classe che fungerà da midlet
 - Il nome deve essere identico a quello della classe



WTK

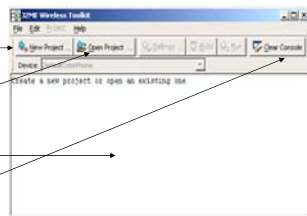


Nuovo Progetto

Apri progetto esistente

Area interazione

Cancella console



Configurazione Midlet



- CLDC, configuration
- Package aggiuntivi JSR 120
- Package addizionali
 - JSR 205
 - JSR 172
 - JSR 75
 - JSR 177
 - JSR 211
 - JSR 82



WTK compile step



- Il Wtk non è un editor java
- Una volta posti i file nelle giuste directory
- Compilazione
- Creazione directory
 - Tmpclasses
 - Tmplib
- Check delle dipendenze
- Creazione dei file JAR e JAD

Semplice midlet



```
public class PrintMidlet extends MIDlet {
    private Form mainForm = new Form ("Prima Midlet");

    protected void startApp() throws
        MIDletStateChangeException {
        Display.getDisplay(this).setCurrent(
            mainForm);
        mainForm.append("Prima midlet: ");
        System.out.println("Prima midlet:
        ");
    }

    protected void pauseApp() {}
    protected void destroyApp(boolean
        arg0) throws
        MIDletStateChangeException {}
}
```



Una semplice midlet



```
public class PrintMidlet extends MIDlet {
    private Form mainForm = new Form ("Prima Midlet");

    protected void startApp() throws
        MIDletStateChangeException {
        Display.getDisplay(this).setCurrent(mainForm);
        mainForm.append("Prima midlet: ");
        System.out.println("Prima midlet: ");
    }

    protected void pauseApp() {}
    protected void destroyApp(boolean arg0) throws
        MIDletStateChangeException {}
}
```

- È obbligatorio implementare le classi
 - startApp
 - pauseApp
 - destroyApp
- Il costruttore di default è inutile (in questo caso)
- Per poter visualizzare dati bisogna utilizzare una Form
- Il valore restituito dalla System
 - Non è visualizzato sul dispositivo, ma nella console
 - L'append nella form è visualizzato nel dispositivo
- Da notare che Display è gestito all'interno di startApp

Midlet Calcolatrice



- Costruiamo un midlet calcolatrice
- Tre parametri in input
 - Due operandi, interi
 - Un operatore, +, -, *, /
- Due comandi
 - Risultato
 - Exit
- Un output
 - Risultato operazione

Midlet: Calcolatrice



```
private Form mainForm = new Form ("Calcolatrice");
private TextField symbolField = new TextField ("Primo operando:", "", 20, TextField.ANY);
private TextField symbolField_2 = new TextField ("Secondo operando:", "", 20, TextField.ANY);
private TextField symbolField_operaz = new TextField ("+-*/", "", 20, TextField.ANY);
private StringItem resultItem = new StringItem ("", "");
private Command registerCommand = new Command ("=", Command.SCREEN, 1);
private Command exitCommand = new Command ("Exit", Command.EXIT, 5);

public ClientMIDlet() {
    mainForm.append(symbolField);
    mainForm.append(symbolField_2);
    mainForm.append(symbolField_operaz);
    mainForm.append(resultItem);
    mainForm.addCommand (registerCommand);
    mainForm.addCommand (exitCommand);
    mainForm.setCommandListener (this);
}
}
```

Midlet: Calcolatrice



Midlet: Calcolatrice



```
protected void destroyApp(boolean arg0){}
public void run() {
    int op1, op2;
    try {
        op1 = Integer.parseInt(symbolField.getString());
        op2 = Integer.parseInt(symbolField_2.getString());
    } catch (NumberFormatException e) {
        resultItem.setText("Operando non riconosciuto");
        return;
    }
    String operaz = symbolField_operaz.getString();
    resultItem.setLabel("Risultato: ");
    if (operaz.equals("+")) resultItem.setText(Integer.toString(op1+op2));
    else if (operaz.equals("-")) resultItem.setText(Integer.toString(op1-op2));
    else if (operaz.equals("*")) resultItem.setText(Integer.toString(op1*op2));
    else if (operaz.equals("/")) try {
        resultItem.setText(Integer.toString(op1/op2));
    } catch (ArithmeticException e) {resultItem.setText("Possibile divisione per 0");}
    else resultItem.setText("Operatore non riconosciuto");
}
```

Input

Operatore

Calcolo Output

Midlet: Cycle Life



Sviluppiamo una midlet per visualizzare il ciclo di vita di una midlet

- Utilizziamo un thread in background controllato da un Timer
- Il thread viene messo in pause se è attivo
- Viene attivato se è in pause
- Dopo un "n" volte viene ucciso

Thread:

```
synchronized (this) {
    if (thread == null) {
        thread = new Thread() {
            public void run() {
                System.out.println("Thread
running");
                while (thread == this) {
                    try {
                        Thread.sleep(1000);
                    } catch (InterruptedException ex) {
                    }
                }
                System.out.println("Thread
terminating");
            }
        };
        thread.start();
    }
}

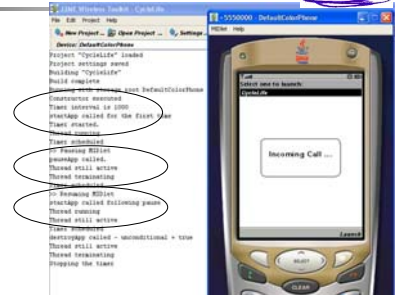
protected void pauseApp() {
    System.out.println("pauseApp called.");
    synchronized (this) {
        if (thread != null) {
            thread = null;
        }
    }
}
```

Midlet CycleLife

Viene invocato il costruttore, startApp

Il thread è in pause, l'emulatore simula una chiamata in ingresso

Il thread riparte



Timer

```
private void startTimer() {
    // Create a task to be run
    task = new TimerTask() {
        private boolean isPaused;
        private int count;

        public void run() {
            // Pause or resume the MIDlet.
            System.out.println("Timer scheduled");
            if (count++ == 2) {
                // Terminate the MIDlet
                try {
                    ExampleMIDlet.this.destroyApp(true);
                } catch (MIDletStateException ex) {}
                ExampleMIDlet.this.notifyDestroyed();
                return;
            }
        }
    };
}

if (isPaused) {
    System.out.println(">> Resuming
MIDlet");
    ExampleMIDlet.this.resumeRequest();
    isPaused = false;
} else {
    System.out.println(">> Pausing
MIDlet");
    isPaused = true;
    ExampleMIDlet.this.pauseApp();
    ExampleMIDlet.this.notifyPaused();
}

// Create a timer and schedule it to run
timer = new Timer();
timer.schedule(task, timerInterval,
timerInterval);
System.out.println("Timer started.");
}
```

Midlet and Internet

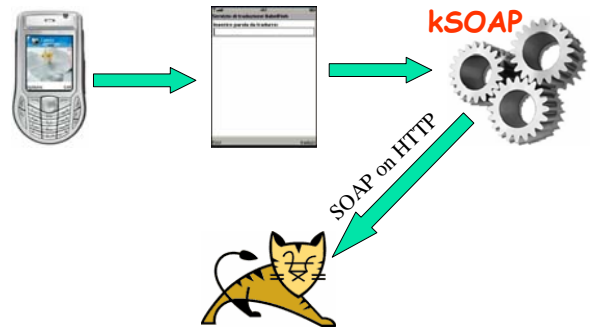
- Non solo applicazioni standalone
- Possibilità di interazione con la rete
- Invocazioni http
- Invio di chiamate voce
- Invio di SMS
- Invio di MMS
- Invocazioni di Web Service

Traduttore



- Midlet per l'invocazione di un servizio di traduzione
- Client è una midlet
- Il canale di comunicazione è un canale HTTP
- I dati sono due stringhe una di input una di output
- I messaggi sono "SOAP standard", xml

Invocation Schema



Traduttore (2)



- Architettura
- Client smartphone
 - Canale di comunicazione http
 - Server, Tomcat
 - Application Server for web service
 - Axis 1.2 RC2
 - Web Service, JWS standard
 - Serializzazione dati lato client kSOAP

Midlet Traduttore: Costruttore



```
public ClientMIDlet () {  
    mainForm.append (symbolField);  
    mainForm.append (resultItem);  
    mainForm.addCommand (doCommand);  
    mainForm.addCommand  
    (exitCommand);  
    mainForm.setCommandListener (this);  
}
```

Midlet Traduttore



```
public void startApp () {
    Display.getDisplay (this).setCurrent
    (mainForm);
}

public void pauseApp () {
}

public void destroyApp (boolean
unconditional) {
}
```

Request Soap Messages



```
POST /axis/my_services/Translator.jws HTTP/1.1
SOAPAction: urn:TranslatorService#translator
Content-Type: text/xml
Content-Length: 515
User-Agent: kSOAP/1.0
Host: 127.0.0.1
```

```
<SOAP-ENV:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" >
<SOAP-ENV:Body SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" >
<Traduci xmlns="urn:TranslatorService" id="o0" SOAP-ENC:root="1">
<sourcedata xmlns="" xsi:type="xsd:string">hello</sourcedata>
</Traduci>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Midlet Traduttore: RUN



```
public void run() {
try {
    String symbol = symbolField.getString ();
    resultItem.setLabel ("Result: ");
    SoapObject rpc = new SoapObject("urn:TranslatorService", "Traduci");
    rpc.addProperty ("sourcedata", symbol);
    resultItem.setText (""+new Http Transport
("http://localhost:8083/axis/my_services/Translator.jws",
"urn:TranslatorService#translator").call (rpc));
} catch (Exception e) {
    e.printStackTrace ();
    resultItem.setLabel ("Error:");
    resultItem.setText (e.toString ());
}
}
}
```

Namespace
Nome servizio
Protocollo trasporto
URL servizio
Invocazione servizio
Porta Servizio

Response SOAP Message



```
HTTP/1.1 200 OK
Set-Cookie: JSESSIONID=C888B3E6B082CE8B1608A32E860FEAE9; Path=/axis
Content-Type: text/xml; charset=utf-8
Transfer-Encoding: chunked
Date: Mon, 05 Dec 2005 16:24:46 GMT
Server: Apache-Coyote/1.1
```

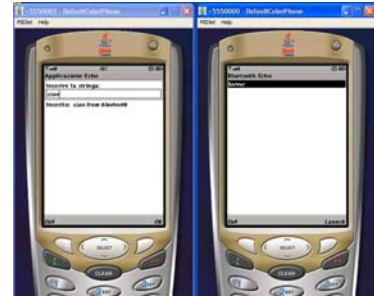
```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soapenv:Body>
<ns1:TraduciResponse
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="urn:TranslatorService">
<TraduciReturn xsi:type="xsd:string">clao</TraduciReturn>
</ns1:TraduciResponse>
</soapenv:Body>
</soapenv:Envelope>
```

Midlet Dati su Bluetooth

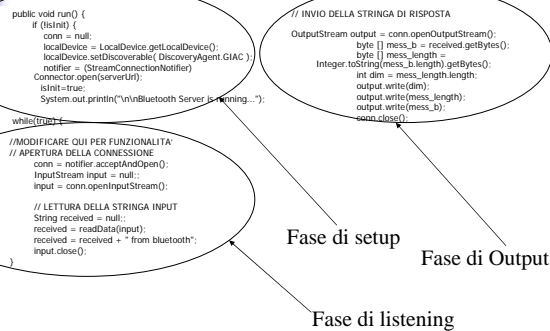
- BlueMidlet
 - Input, semplice stringa
 - Creazione Canale Bluetooth
 - Invio dati sul canale bluetooth
 - Ricezione dati da parte del server
 - Invio dei dati al client
- Server Bluetooth (Echo Server)
 - Resta semplicemente in attesa
 - Restituisce i dati che gli vengono passati



Echo Over Bluetooth



Echo Over Bluetooth channel



Ringraziamenti

- Dipartimento Informatica e Applicazioni
- Associazione Studentesca "Archimede"

