
Java2Me.org


jug
Torino

Sono lieti di Presentarvi

Java 2 Micro Edition:

- MIDP 1.0
- MIDP 2.0
- Optional APIs

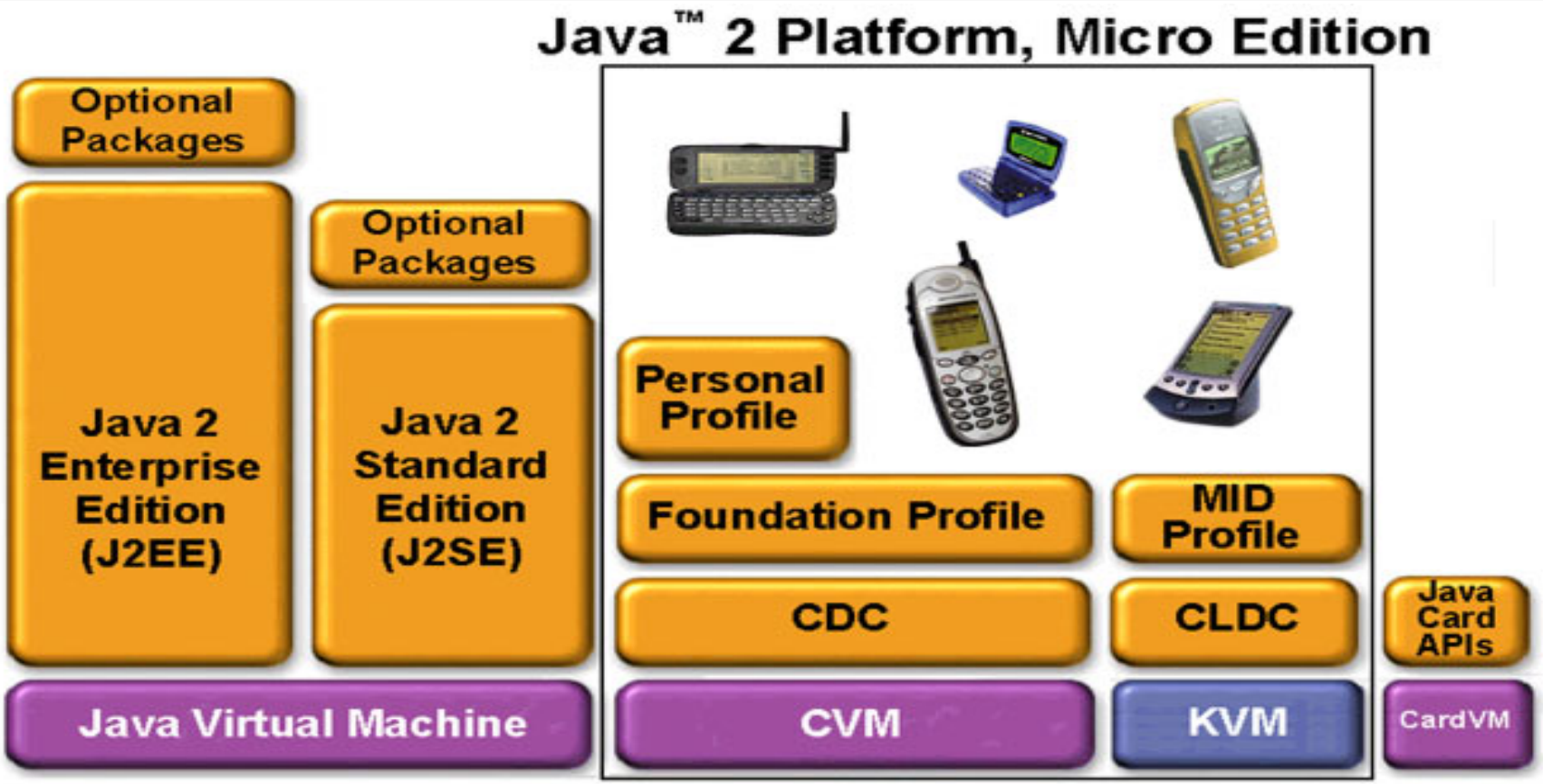
... in 100 Minuti

Non vi preoccupate!

- Potrete trovare queste slides liberamente scaricabili su

http://java2me.org/

Le versioni di Java



Le configurazioni

Connected Device Configuration (CDC)

- 512 Kb minimo di memoria per l'esecuzione di Java
- 256 Kb minimo per l'allocazione di memoria al momento dell'esecuzione
- Connettività di rete, possibilmente persistente e a banda larga

Connected Limited Device Configuration (CLDC)

- 128 Kb di memoria per l'esecuzione di Java
- 32 Kb per l'allocazione di memoria al momento dell'esecuzione
- Interfaccia utente limitata
- Alimentazione elettrica ridotta, per esempio fornita da batterie
- Connettività di rete, solitamente wireless, a banda stretta e ad accesso discontinuo

I Profili

- Un profilo è un'estensione della Configurazione, ad esempio il MIDP (Mobile Information Device Profile) definisce le API per i componenti dell'interfaccia utente, gli input, la gestione degli eventi, la memoria persistente, le funzionalità di rete e i timer tenendo conto delle limitazioni dello schermo e della memoria degli apparecchi mobili.

K Virtual Machine

- Necessita di soli 40-80 Kb di memoria
- Necessita di soli 20 Kb di memoria dinamica
- Può essere eseguita su microprocessori con un clock di soli 25MHz
- Attualmente i processori di fascia medio alta hanno un clock di 100Mhz...

La configurazione CLDC

Si Occupa di:

- Definire le caratteristiche del linguaggio Java e della Virtual Machine supportate
- Fornire un set minimo di librerie di base
- Gestire gli stream di I/O
- Sicurezza
- Networking
- Internazionalizzazione

Lascia ai profili

- La gestione del ciclo di vita delle applicazioni (installazione, lancio, cancellazione)
- L'implementazione di interfacce utenti
- La "cattura" e la gestione degli eventi
- L'interazione tra l'utente e l'applicazione

Package java.lang

- l'assenza nella CLDC delle classi `Float` e `Double`
- si limita a fornire metodi per ottenere il massimo (`max`) o il minimo (`min`) tra due numeri (`int` o `long`) e il valore assoluto (`abs`).
- La CLDC, inoltre, non supporta i daemon thread e gruppi di thread

Package java.lang

- non è previsto il supporto del codice nativo (Java Native Interface)
- non prevede la possibilità di utilizzare caricatori di classi definiti dall'utente
- Le condizioni di errore che la CLDC si limita a rilevare riguardano mal funzionamenti della macchina virtuale (`VirtualMachineError`) o l'impossibilità di allocare memoria (`OutOfMemoryError`)

Package java.lang

- non supporta la Java Reflection
- mancanza della serializzazione che a sua volta comporta l'impossibilità di supportare RMI

Package java.util

- l'interfaccia Enumeration, Hashtable, Vector e Stack per quanto che riguarda le "collezioni di oggetti"
- Calendar, Date e TimeZone per la gestione e la manipolazione di data e ora
- La classe Random per la generazione di numeri pseudo casuali

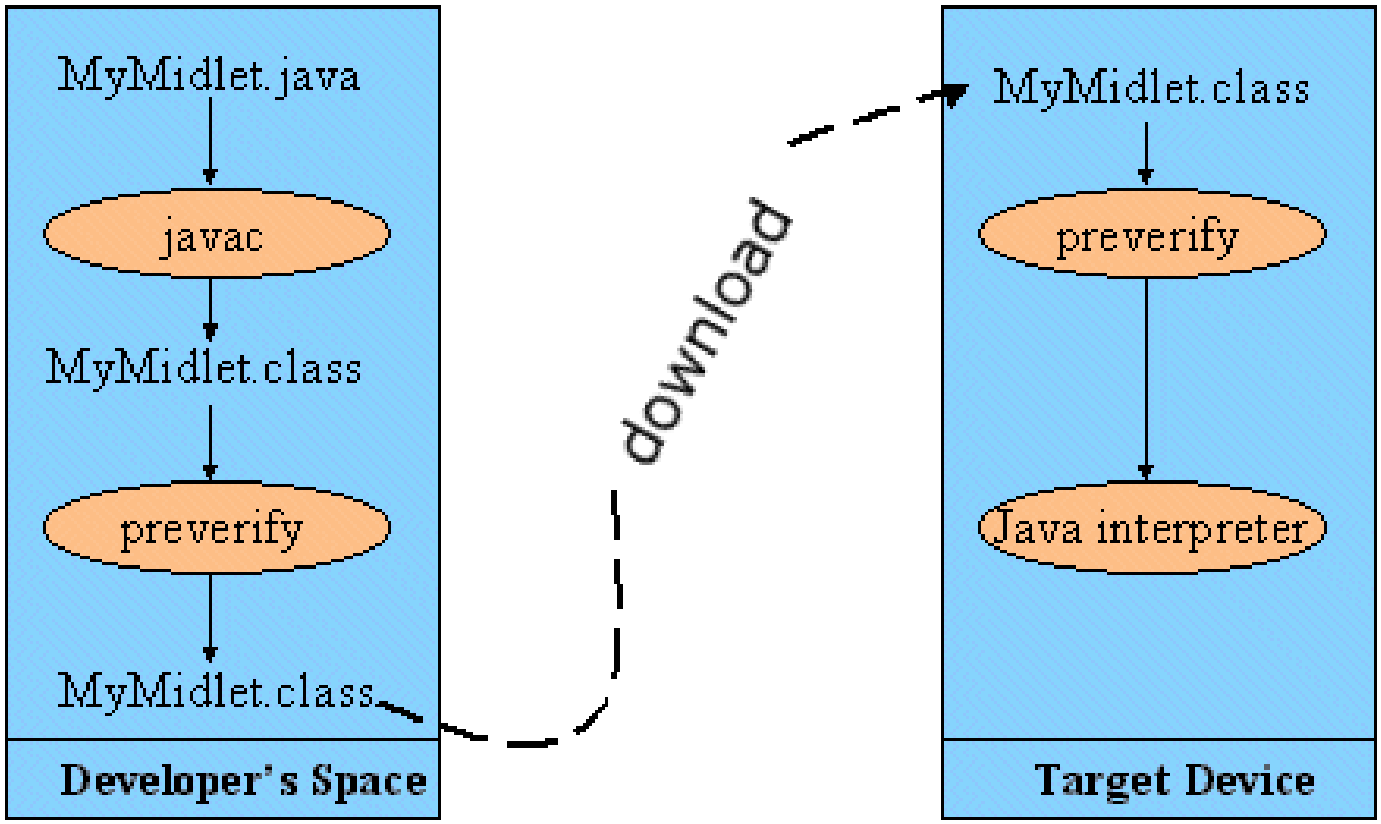
Package java.io

```
- InputStreamReader
(InputStream is,
  String encoding);
- OutputStreamWriter
(OutputStream os,
  String encoding);
```

La sicurezza

- a livello di Virtual Machine
- a livello applicazione
- il processo di verifica avviene in due fasi distinte:
- al di fuori del dispositivo (off-device pre-verification)
- nel dispositivo (in-device verification);

Il processo di verifica



Sandbox model

- ogni applicazione deve aver superato il processo di verifica;
- ogni applicazione a accesso ad un ben determinato set di API (quelle previste dalla CLDC, dal profilo utilizzato e eventuali altri classi);
- non è prevista la possibilità di ridefinire caricatori di classe a livello applicazione. L'unico class loader ammesso è quello standard della virtual machine;
- il codice nativo non è accessibile alle applicazioni

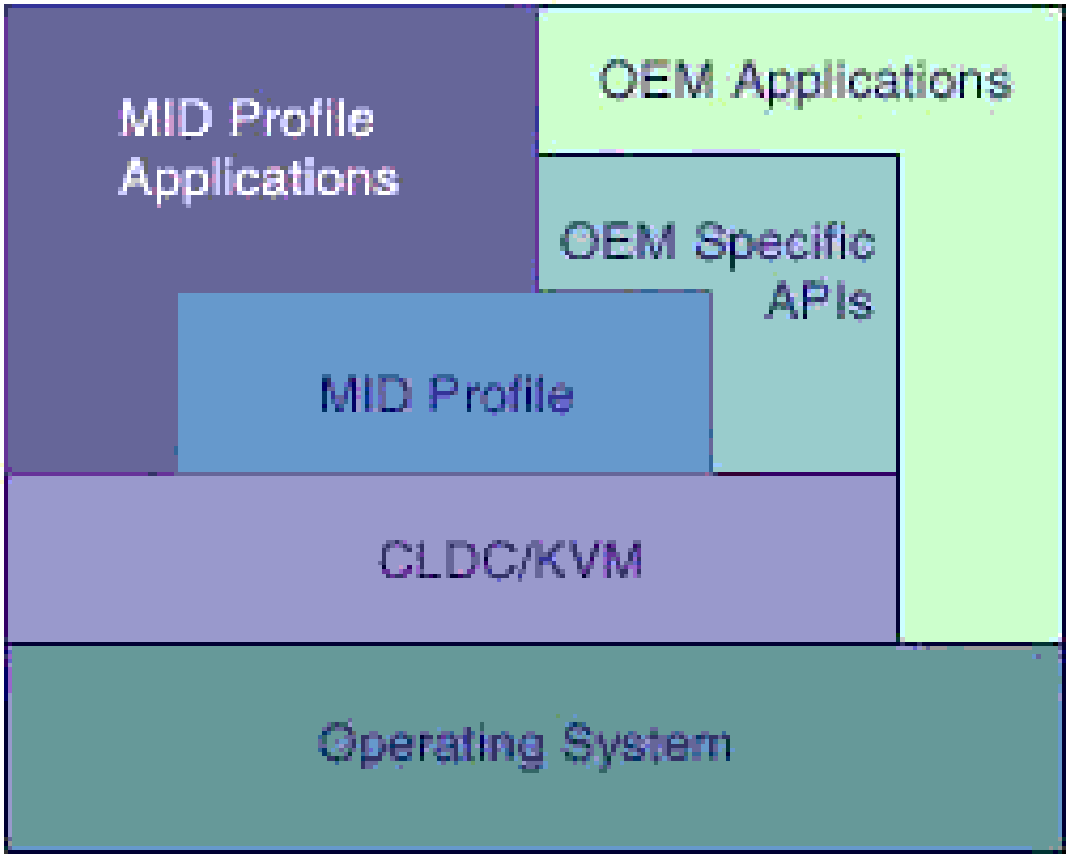
MIDP (Mobile Information Device Profile)

- gestire il ciclo di vita delle applicazioni (caricamento - esecuzione - distruzione delle applicazioni)
- gestire l'interfaccia utente (dispositivi di input/output)
- salvataggio persistente dei dati
- networking (implementazione dei protocolli)

MIDP

- javax.microedition.midlet
- javax.microedition.lcdui
- javax.microedition.rms
- javax.microedition.io

Architettura MIDP



javax.microedition.midlet

- `protected abstract void startApp()`

segnala al MIDlet che è stato posto nello stato di attività

- `protected abstract void pauseApp()`

segnala al MIDlet il suo inserimento nello stato di pausa

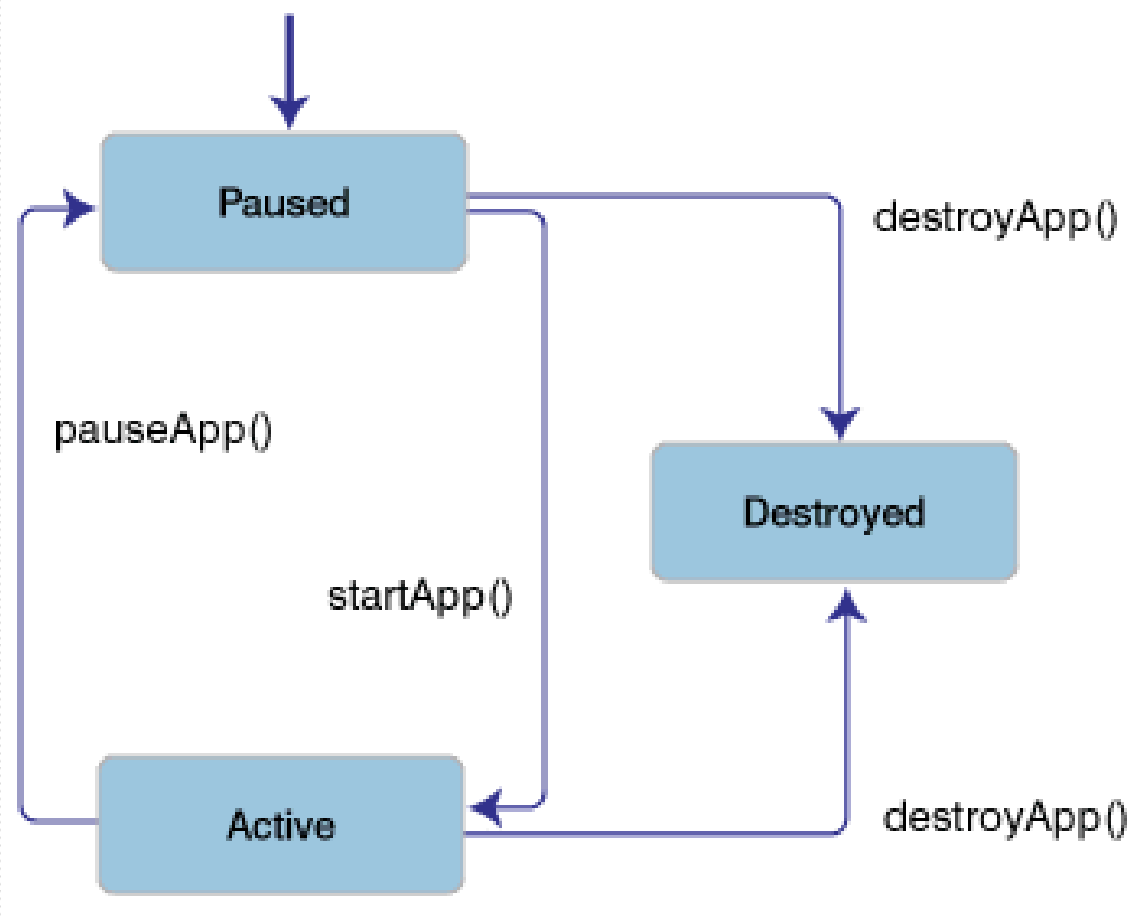
- `protected abstract void destroyApp (boolean unconditional)`

  impone al MIDlet di terminare le proprie attività ed entrare nello stato destroyed.

AMS (Application Management Software)

- garantisce la possibilità di installare e disinstallare MIDlet nel/dal dispositivo
- prepara l'ambiente di esecuzione per il MIDlet (KVM, librerie CLDC-MIDP, librerie di terzi, file JAD, risorse esterne, ecc)
- cattura gli errori che si possono verificare sia in fase di installazione del MIDlet che durante la sua esecuzione

Ciclo di vita di un MIDlet



File MANIFEST

MIDlet-1: FluidTime, /FluidTime.png, FluidTime

MIDlet-Jar-Size: 53547

MIDlet-Jar-URL:
http://wap.java2me.org/FluidTime.jar

MIDlet-Name: FluidTime

MIDlet-Vendor: IDI Ivrea

MIDlet-Version: 1.8

File JAD

MIDlet-1: FluidTime, /FluidTime.png, FluidTime

MIDlet-Name: FluidTime

MIDlet-Vendor: IDI Ivrea

MIDlet-Version: 1.8

MicroEdition-Configuration: CLDC-1.0

MicroEdition-Profile: MIDP-1.0

Classi grafiche e interfaccia utente

- `package javax.microedition.lcdui` (`Limited Connection Device User Interface`)
- `Displayable` e `Display`:
  - la prima rappresenta la "massima astrazione" di un oggetto posizionabile e visualizzabile sullo schermo
  - la seconda (Display) costituisce invece il vero e proprio gestore dello schermo.

Displayable

- `addCommand( Command c)`
- `removeCommand( Command c)`
- `isShown()`
- `setCommandListener (CommandListener listener)`

Display

- `getCurrent()`
- `setCurrent(..)`
- `setCurrent`
  `(Displayable nextDisplayable)`
- `setCurrent`
- `(Alert alert,`
  `Displayable nextDisplayable)`

Form

- `Form(String formName);`
- `Form(String formName, Item[] items);`

- `public int append(Image img);`
- `public int append(Item itm);`
- `public int append(String str);`
- `public void insert`
  `( int itemNumber, Item itm);`

Form

- `public void delete(int itemIndex);`
- `public Item get(itemNumber);`
  `public void set(itemNumber, Item itm);`
- `public int size();`
- `setItemStateListener`
  `(ItemStateListener listener);`
- `public void itemStateChanged(Item itm);`

itemStateChanged

- selezione di un determinato valore in una ChoiceGroup;
- modifica del valore di un Gauge;
- inserimento o modifica di un valore in un TextField;
- inserimento o modifica di una data in un DateField;

DateField

- DateField (String label,
  int mode);

```
DateField
   (
      String label,
      int mode
      TimeZone, timeZone
   );
mode := DATE | TIME | DATE_TIME
```

DateField: get&set

- `Date getDate()`
- `void setDate(Date date)`
- `int getInputMode()`
- `void setInputMode(int mode)`

Gauge

- Gauge

```
(
        String label,
        boolean interactive,
        int maxValue,
        int initialValue
);
```

Gauge: get&set

- `int getValue()`
- `void setValue(int value)`
- `int getMaxValue()`
- `int setMaxValue(int maxValue)`
- `boolean isInteractive()`

StringItem

- StringItem
  (
  String label,
  String text
)
- String getText()
- Void setText(String text)

  In alternativa:

  fmMain.append(“Ciao”);

TextField

- TextField
  (
  String label,
  String text,
  int maxSize,
  int costraints
)

  Ci sarebbero parecchi metodi…

  Ma servono così tanto?

TextField: costraints

- `ANY` – Qualsiasi input
- `EMAILADDR` – Input di tipo indirizzo email
- `NUMERIC` – Solo cifre (anche negative)
- `PASSWORD` – Nasconde i caratteri
- `PHONENUMBER` – Numeri di telefono
- `URL` – Caratteri validi per un URL

Choice e ChoiceGroup

- `ChoiceGroup(String label, int choiceType)`
- `ChoiceGroup`

```
(
        String label,
        int choiceType,
        String[] stringElements,
        Image ImageItems
)
```

ChoiceGroup: i metodi

- `int append(String stringPart, Image imagePart)`
- `void delete(int elementNum)`
- `void insert(int elementNum, String stringElement, Image imageElement)`
- `int getSelectedIndex()`
- `boolean isSelected()`

ChoiceGroup

- `int getSelectedFlags (boolean[] selectedArray_return)`
- `int setSelectedFlags (boolean[] selectedArray_return)`

- `EXCLUSIVE` - Una sola selezione x volta
- `MULTIPLE` – Zero o più soluzioni
- `IMPLICIT` – Non disponibile (vedi List)

Image

- `static Image createImage(String name)`
- `static Image createImage(Image source)`
- `Static Image createImage`

```
(
        byte[] imageData,
        int imageOffset,
        int imageLenght
)
```

- `static Image createImage(int width, int height)`

ImageItem

```
Image im = new Image(“/animage.png”);
fm.append(new ImageItem
   (
      null,
      im,
      ImageItem.LAYOUT_CENTER,
      Null
   ));
ImageItem(String label, Image img, int
   layout, String altText);
```

TextBox

```
TextBox
  (
    String title,
    String defaultValue,
    int maxSize,
    int constraints
  );
```

TextBox

ANY - qualunque sequenza di caratteri;

EMAILADDR - i dati inseriti rappresentano un indirizzo e-mail;

NUMERIC - i dati inseriti devono essere di tipo numerico;

PASSWORD - i caratteri inseriti rappresentano una password;

PHONENUMBER - la sequenza rappresenta un numero telefonico;

URL - l'utente può inserire una sequenza di caratteri rappresentanti una URL

List

- `List(String name, int listType)`
- `List`

```
(
    String name,
    int listType,
    String[] elements,
    Image[] icons
)
```

Tipi di List (da Choice)

- `IMPLICIT` - in questo caso la selezione di uno degli elementi della lista genera una chiamata al listener registrato al quale viene passato un comando del tipo `SELECT_COMMAND`;
- `EXCLUSIVE` - prevede la selezione di un unico elemento della lista;
- `MULTIPLE` - rende possibile la selezione multipla.

Alert

- `Alert(String alertName);`
- `Alert`

```
(
    String alertName,
    String alertText,
    Image alertImage,
    AlertType alertType
);
```

Tipi di Alert

- `ALARM` - serve a notificare all'utente una particolare condizione (es. il sopraggiungere di una determinata data);
- `CONFIRMATION` - conferma all'utente l'esecuzione di un'operazione (es. la validità di una password inserita dall'utente);
- `ERROR` - segnala all'utente una condizione d'errore (es. la non validità di una password inserita dall'utente);
- `INFO` - fornisce informazioni all'utente (es. la password scadrà tra 15 giorni);
- `WARNING` - serve a segnalare la potenziale pericolosità di una operazione richiesta dall'utente (es. la richiesta di cancellazione dei dati contenuti in un database);

Alert: modifica

- `public AlertType getType ();`
- `public void setType (int alertType);`
- `public void setTimeout(int timeout);`

- ```
  public void setCurrent
  (
     Alert alert,
     Displayable nextDisplayable
  );
  ```
  ```

I commands

- Command
  (
    String commandLabel,
    int commandType,
    int commandPriority
  );

Command: tipi

- `BACK` - comando di navigazione che permette di ritornare alla schermata precedente;
- `CANCEL` - comando di cancellazione;
- `OK` - comando di conferma;
- `HELP` - comando per il lancio di un eventuale on-line help;
- `EXIT` - comando di uscita dall'applicazione;
- `SCREEN` - indica un comando che appartiene alla screen corrente (ad esempio "load", "save");
- `STOP` - comando per la terminazione di un qualche processo.

Canvas

API grafiche a basso livello

- `public void paint();`
- `public void repaint();`
- `public void repaint`
  `(`
  `int x, int y,`
  `int width, int height`
  `);`
- `public void serviceRepaint();`

Gestione degli eventi

- `public void keyPressed (int keyCode);`
- `public void keyReleased (int keyCode);`
- `public void keyRepeated (int keyCode);`

Azioni di gioco

- `Cavas.FIRE`
- `Canvas.DOWN`
- `Canvas.FIRE`
- …
- `int getKeyCode(int gameAction)`
- `int getGameAction(int keyCode)`
- `String getKeyName(int keyCode)`

Eventi del puntatore

Proviamo ad impostare :

```
touch_screen = true in
x:/WTK_HOME/wtklib/devices
```

E implementiamo:

- `boolean hasPointerEvents();`
- `boolean hasPointerMotionEvents();`
- `void pointerDragged(int x, int y);`
- `void pointerPressed(int x, int y);`
- `void pointerReleased(int x, int y);`

Funzionalità grafiche

- il disegno di testo e immagini:

- `public void drawChar (char c, int x, int y, int anchor);`
- `public void drawChars (char[] c, int offset, int length, int x, int y, int anchor);`
- `public void drawString (String s, int x, int y, int anchor);`
- `public void drawChars (String s, int offset, int length, int x, int y, int anchor);`
- `public void drawImage(Image img, int x, int y, int anchor);`

Gli Anchor Point…

- costanti orizzontali
  - `LEFT, HCENTER, RIGHT`
- verticali
  - `TOP, BASELINE, BOTTOM`

`RIGHT|TOP` :: posiziona l'elemento a destra, sopra l'anchor

Linee, rettangoli, archi

- `public void drawLine (int x1, int y1, int x2, int y2);`
- `public void drawRect (int x, int y, int width, int height);`
- `public void drawRoundRect (int x, int y, int width, int height, int arcW, arcH);`
- `public void drawArc(int x, int y, int width, int height, int startAngle, int arcAngle);`

Riempimento

- `- fillRect(..)`
- `- fillRoundRect(..)`
- `- fillArc(..)`
- Con gli stessi parametri visti prima per la creazione.

Baco MIDP 1.0

Ogni volta che implementiamo il metodo paint dobbiamo fare un “clear screen”: se non lo facessimo, in caso di repaint il canvas rimarebbe disegnato sul display… Baco!

```
g.setColor(255, 255, 255);
g.fillRect(0,0,getWidth(),getHeight());
```

Font

- `static Font getFont (int face, int style, int size);`
- `static Font getDefaultFont();`
- `void setFont(Font font);`

Alcune costanti:

```
Font.FACE_SYSTEM, Font.FACE_PROPORTIONAL
Font.STYLE_PLAIN, Font.STYLE_BOLD
Font.SIZE_SMALL, Font.SIZE_MEDIUM
```

Gestione dei colori

Il MIDP supporta una gestione del colore a 24 bit

- `isColor()`
- `numColors()`

- `public int getColor(); (0xRRGGBB)`
- `public int getRedComponent(); (0-255)`
- `public int getGreenComponent();`
- `public int getBlueComponent();`

Gestione Colori

- `public void setColor(int RGB);`
- `public void setColor`
  `(int red, int green, int blue);`

E per la scala di grigio:

- `public void setGrayScale(int val);`
- `public int getGrayScale();`

Le immagini

- Il formato attualmente supportato per le immagini nel MIDP è il PNG (Portable Network Graphics) nella versione 1.0

- Immutable image
- Mutable image

Creazione di immagini (immutable)

```
static Image createImage
(
   byte data[],
   int imageOffset,
   int imageLength
);
```

```
static Image createImage(Image imageSource);
```

```
static Image createImage(String name);
```

Creazione di immagini (mutable)

- `static Image createImage (int width, int height);`

Con i metodi:

- `public int getWidth();`
- `public int getHeight();`
- `public boolean isMutable();`

Otteniamo informazioni e tipo di un'immagine

Mostrare l'immagine

```
Image im =
Image.createImage("/imageTest.png");
```

```
Protected void paint(Graphics g)
{
   g.drawImage
   (im,10,10,Graphics.LEFT|Graphics.TOP);
}
```

Rendering

- `public Graphics getGraphics();`

- Crea un contesto grafico associato all'immagine, utilizzabile per il rendering.
- Si può richiamare solo su un'immagine mutable.

Tradurre le coordinate

- `void translate(int x, int y);`
- `int getTranslateX();`
- `int getTranslateY();`

Aree di ritaglio

- Per ridipingere solo una parte del display:
  [In questo modo riduciamo il tempo di refresh]

- `void setClip(int x, int y, int w, int h);`
- `void clipRect(int x, int y, int w, int h);`
- `int getClipX();`
- `int getClipY();`
- `int getClipHeight();`
- `int getClipWidth();`