


J2ME part 2

Magnus Bladh
Streetmedia 7 AB

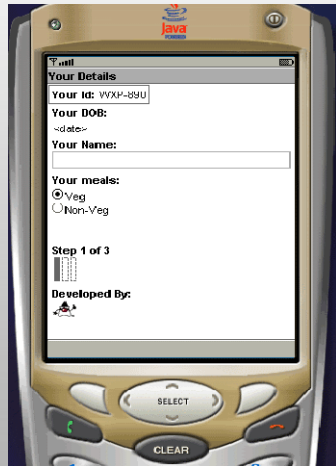
street media 

street media 

User interface

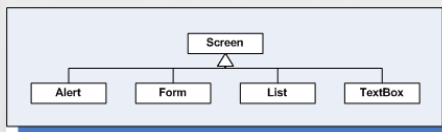
- ◆ The UI classes of MIDP `javax.microedition.lcdui` package can be divided into two logical groups:
 - ◆ High level group
 - ◆ Low level group

High level API



- ◆ The classes of the high-level group are perfect for development of MIDlets that target the maximum number of devices, because these classes do not provide exact control over their display
- ◆ Minimal control over look and feel
- ◆ Look and feel is managed by device
- ◆ Good portability.
- ◆ Used for forms.

High level API Classes



- ◆ Alerts are best used in informational or error messages that stay on the screen for a short period of time.
- ◆ A list contains one or more choices (elements).
- ◆ Text is entered by the user using a textbox.
- ◆ A form is a collections of instances of the Item interface
- ◆ There are eight Item types: *StringItem*, *DateField*, *Textfield*, *ChoiceGroup*, *Spacer*, *Gauge*, *ImageItem*, *CustomItem*.

Low level API



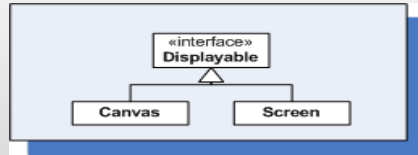
- The classes of the low-level group are perfect for MIDlets where precise control over the location and display of the UI elements is important and required.
- Used for games and other applications
- With more control comes less portability

Low level API classes



- ◆ Good portability.
- ◆ Used for forms.

The “Displayable interface”



- ♦ For you to be able to show a UI element on a device screen, whether high- or low-level, it must implement the Displayable interface.
- ♦ A displayable class may have a title, a ticker, and certain commands associated with it.
- ♦ This implies that both the Screen and Canvas classes and their subclasses implement this interface
- ♦ A Displayable class is a UI element that can be shown on the device's screen. Display class abstracts the display functions of a device's screen and makes them available to you. It provides methods to gain info about the screen and to show or change the UI element that you want displayed.

Handling User Commands

- ♦ A MIDlet interacts with a user through commands. A command is the equivalent of a button or a menu item in a normal application, and can only be associated with a displayable UI element.
- ♦ The Displayable class allows the user to attach a command to it by using the method `addCommand(Command command)`.
- ♦ The Command class holds the information about a command. This information is encapsulated in four properties. These properties are: a short label, an optional long label, a command type, and a priority.

Network Programming with J2ME

- The networking in J2ME has to be very flexible to support a variety of devices and has to be very device specific at the same time.
- To meet these challenges, the Generic Connection framework is first introduced in the CLDC.
- There is 1 Connection class and 7 connection interfaces.
- The 7 connection interfaces define the abstractions of 6 basic types of communications: basic serial input, basic serial output, datagrams communications, sockets communications, notification mechanism in a client-server communication, and basic HTTP communication with a Web server.

Connections!!!

- The 7 connection interfaces define the abstractions of 6 basic types of communications: basic serial input, basic serial output, datagrams communications, sockets communications, notification mechanism in a client-server communication, and basic HTTP communication with a Web server.
- To create a connection you use the following code:
 - `Connector.open(String connect);`
- The parameter connect is a String variable. It has a URL-like format: `{protocol}:[{target}][{params}]`
 - `Connection hc = Connector.open("http://www.test.com");`

Http connection

- The `HttpConnection` class is defined in J2ME MIDP to allow developer to handle http connections in their applications.
- The `HttpConnection` class is guaranteed available on all MIDP devices.

Example

```
HttpConnection hc = (HttpConnection);
Connector.open("http://www.wirelessdevnet.com");

InputStream is = new hc.openInputStream();
int ch; // Check the Content-Length first
long len = hc.getLength();
if(len!=-1) {
    for(int i = 0;i<len;i++)
        if((ch = is.read())!= -1)
            System.out.print((char) ch);
}
else { // if the content-length is not available
    while ((ch = is.read()) != -1)
        System.out.print((char) ch);
}
is.close();
hc.close();
```

SOCKET CONNECTIONS

- New interfaces added to enable low-level IP networking in midp2.0:
- The `SocketConnection` interface defines the socket stream connection. You use it when writing MIDlets that access TCP/IP servers.
- *Socket Connection:* `Connector.open("socket://java.sun.com:port");`

Example

```
SocketConnection client = (SocketConnection) Connector.open("socket://" +
hostname + ":" + port);
// set application-specific options on the socket. Call setSocketOption to set
other options
client.setSocketOption(Delay, 0);
client.setSocketOption(KEEPALIVE, 0);

InputStream is = client.openInputStream();
OutputStream os = client.openOutputStream();

// send something to server
os.write("some string".getBytes());
// read server response
int c = 0;
while((c = is.read()) != -1) {
    // do something with the response
}
is.close(); // close streams and connection
os.close();
client.close();
```