Chapter 6: Mobile Operating System

Features of Windows CE OS:
- Similar to windows 95.
- It is a Microsoft’s mobile os used in smart phones and mobile devices.
- It is a 32 bit multitasking, multithreading os that has scalable, open architecture design, providing support for variety of devices. Windows CE is compact, providing high performance in limited memory configurations.
- Standard communication support is built into this OS enabling access of internet.
- Integrated power management enabling long battery life.
- GUI facilitating ease of use for end users.
- Subset of win 32 API: widows CE supports more than 700 of the most frequently used win 32 APIs, enabling developers to take advantage of vast amounts of third party programming resources, tools, documentation for their windows CE based development.
- Low cost, familiar development tools.
- Scalable, full featured OS.
- Extensive and extensible device support- supports keyboard, mouse devices, touch panels, serial ports, Ethernet modems, USB devices, audio devices, parallel port, printer devices, storage devices.

Features of Symbian OS:

User interface
- Symbian has had a native graphics toolkit since its inception, known as AVKON (formerly known as Series 60).
- S60 was designed to be manipulated by a keyboard-like interface metaphor, such as the ~15-key augmented telephone keypad, or the mini-QWERTY keyboards.
- AVKON-based software is binary-compatible with Symbian versions up to and including Symbian^3.

Browser
- Symbian^3 and earlier have a built-in Web Kit based browser. Symbian was the first mobile platform to make use of Web Kit (in June 2005).
- Some older Symbian models have Opera Mobile as their default browser.
- Multiple language support.
- Symbian has strong localization support enabling manufacturers and 3rd party application developers to localize their Symbian based products in order to support global distribution. Current Symbian release (Symbian Belle) has support for 48 languages, which Nokia makes available on device in language packs.

Application development
- Symbian OS previously used a Symbian specific C++ version, along with Carbide .c++ integrated development environment (IDE), as the native application development environment.
- Web Run time (WRT) is a portable application framework that allows creating widgets on the S60 Platform; it is an extension to the S60 Web Kit based browser that allows launching multiple browser instances as separate JavaScript applications.

iOS
iOS (previously iPhone OS) is a mobile operating system developed and distributed by Apple Inc.
- Originally released in 2007 for the iPhone and iPod Touch, it has been extended to support other Apple devices such as the iPad and Apple TV.
- In June 2010, Apple rebranded iPhone OS as "iOS".
- The trademark "IOS" had been used by Cisco for over a decade for its operating system, IOS, used on its routers.
- In 2011, iOS accounted for 60% of the market share for smartphones and tablet computers. By the end of 2012, iOS accounted for 21% of the smartphone OS market and 43.6% of the tablet OS market.

Android Architecture

Wide microprocessor support  Android operating system is a stack of software components which is roughly divided into five sections and four main layers as shown below in the architecture diagram.

![Android Architecture Diagram](image)

**Linux kernel**

At the bottom of the layers is Linux - Linux 2.6 with approximately 115 patches. This provides basic system functionality like process management, memory management, device management like camera, keypad, display etc. Also, the kernel handles all the things that Linux is really good at such as networking and a vast array of device drivers, which take the pain out of interfacing to peripheral hardware.

**Libraries**

On top of Linux kernel there is a set of libraries including open-source Web browser engine WebKit, well known library libc, SQLite database which is a useful repository for storage and sharing of application data, libraries to play and record audio and video, SSL libraries responsible for Internet security etc.
**Android Runtime**
This is the third section of the architecture and available on the second layer from the bottom. This section provides a key component called **Dalvik Virtual Machine** which is a kind of Java Virtual Machine specially designed and optimized for Android. The Dalvik VM makes use of Linux core features like memory management and multi-threading, which is intrinsic in the Java language. The Dalvik VM enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine.

The Android runtime also provides a set of core libraries which enable Android application developers to write Android applications using standard Java programming language.

**Application Framework**
The Application Framework layer provides many higher-level services to applications in the form of Java classes. Application developers are allowed to make use of these services in their applications.

**Applications**
You will find all the Android application at the top layer. You will write your application to be installed on this layer only. Examples of such applications are Contacts Books, Browser, and Games etc.

The steps that an application goes through from starting to finishing Slightly different than normal Java life cycle due to:
- the difference in the way Android application are defined
- the limited resources of the Android hardware platform

  **Lifecycle:**
  - Each application runs in its own process.
  - Each activity of an app is run in the apps process
  - Processes are started and stopped as needed to run apps components.
  - Processes may be killed to reclaim needed resources.
  - Killed apps may be restored to their last state when requested by the user
Life Cycle of Android Activity:
As an activity transitions from state to state, it is notified of the change by calls to the following protected methods:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>onCreate()</td>
<td>This is the first callback and called when the activity is first created.</td>
</tr>
<tr>
<td>onStart()</td>
<td>This callback is called when the activity becomes visible to the user.</td>
</tr>
<tr>
<td>onResume()</td>
<td>This is called when the user starts interacting with the application.</td>
</tr>
<tr>
<td>onPause()</td>
<td>The paused activity does not receive user input and cannot execute any code and called when the current activity is being paused and the previous activity is being resumed.</td>
</tr>
<tr>
<td>onStop()</td>
<td>This callback is called when the activity is no longer visible.</td>
</tr>
<tr>
<td>onDestroy()</td>
<td>This callback is called before the activity is destroyed by the system.</td>
</tr>
<tr>
<td>onRestart()</td>
<td>This callback is called when the activity restarts after stopping it.</td>
</tr>
</tbody>
</table>

Taken together, these seven methods define the entire lifecycle of an activity.
Features of Android

• Popularity (High market share)
• Enabled SMS
• Enabled GPS
• Cameras & microphones
• Open source apps
• Development with Java
• Linux based
• Short learning curve with IDE Java Eclipse IDE for Android app development
• Easy deployment to mobile device
• Easy to distribute apps via Android Market
  Share app, Sell app,
  Download & reuse open source apps

Limitations

• Screen size
• Battery life time
• Processor capacity
• Slow network access and Web browser
• Input: soft keyboard, phone keypad, touch screen, or stylus

Security

Write steps to develop a User interface with two text box and two buttons.

Creating a User Interface

The graphical user interface for an Android app is built using a hierarchy of View and ViewGroup Objects. View objects are usually UI widgets such as buttons or textfields. ViewGroup objects are invisible view containers that define how the child views are laid out, such as in a grid or a vertical list.

Android provides an XML vocabulary that corresponds to the subclasses of View and ViewGroup so you can define your UI in XML using a hierarchy of UI elements.

Create a Linear Layout

1. In Android Studio, from the res/layout directory, open the activity_my.xml file. The BlankActivity template you chose when you created this project includes the activity_my.xml file with a RelativeLayout root view and a TextView child view.
2. In the Preview pane, click the Hide icon to close the Preview pane. In Android Studio, when you open a layout file, you’re first shown the Preview pane. Clicking elements in this pane opens the WYSIWYG tools in the Design pane. For this lesson, you’re going to work directly with the XML.
3. Delete the <TextView> element.
4. Change the <RelativeLayout> element to <LinearLayout>.
5. Add the android:orientation attribute and set it to "horizontal".
6. Remove the android:padding attributes and the tools:context attribute.

**Creating a Text Field**

A text field allows the user to type text into your app. It can be either single line or multi-line. Touching a text field places the cursor and automatically displays the keyboard. In addition to typing, text fields allow for a variety of other activities, such as text selection (cut, copy, paste) and data look-up via auto-completion.

Text fields can have different input types, such as number, date, password, or email address. The type determines what kinds of characters are allowed inside the field, and may prompt the virtual keyboard to optimize its layout for frequently used characters.

You can add a text field to your layout with the EditText object. You should usually do so in your XML layout with a <EditText> element.

![Fig: Text field with keyboard](image)

As with every View object, you must define certain XML attributes to specify the EditText object's properties.

1. In the activity_my.xml file, within the <LinearLayout> element, define an <EditText> element with the id attribute set to @+id/edit_message.
2. Define the layout_width and layout_height attributes as wrap_content.
3. Define a hint attribute as a string object named edit_message.

**Add a Button**

A button consists of text or an icon (or both text and an icon) that communicates what action occurs when the user touches it. Depending
on whether you want a button with text, an icon, or both, you can create the button in your layout in three ways:

1. **With text, using the Button class:**
   ```xml
   <Button
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="@string/button_text"
   />
   ...
   ```

2. **With an icon, using the ImageButton class:**
   ```xml
   <ImageButton
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:src="@drawable/button_icon"
   />
   ...
   ```

3. **With text and an icon, using Button class with the android:drawableLeft attribute:**
   ```xml
   <Button
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:drawableLeft="@drawable/button_icon"
       android:text="@string/button_text"
   />
   ...
   ```

1. In Android Studio, from the `res/layout` directory, edit the `activity_my.xml` file.

2. Within the `<LinearLayout>` element, define a `<Button>` element immediately following the `<EditText>` element.
3. Set the button's width and height attributes to "wrap_content" so the button is only as big as necessary to fit the button's text label.

4. Define the button's text label with the android:text attribute; set its value to the button_send string resource you defined in the previous section.

**Note:** This button doesn't need the android:id attribute, because it won't be referenced from the activity code.

The layout is currently designed so that both the EditText and Button widgets are only as big as necessary to fit their content, as shown in figure.

To configure how the Button control looks, adjust the control’s properties by selecting the control (either in the Outline tab or the Preview window) and changing its properties in the Properties Tab.

Specific properties you will want to be aware of:

- Give the Button or ImageButton control a unique name using the id property.
- Set the text displayed on the Button control using the text property; set the image displayed on the ImageButton control using the src property.
- Set the layout height and layout width properties of the control to wrap_content.
- Set any other properties you desire to adjust the control’s appearance.
  
  For example, adjust the font of a Button using the Text color, text size, and text style properties.