

VASAVI VIDYA TRUST GROUP OF INSTITUTIONS, SALEM- 103

CLASS: I MCA

SUBJECT CODE: MC5209

SUBJECT: MOBILE APPLICATION DEVELOPMENT

UNIT: I

MC5209

MOBILE APPLICATION DEVELOPMENT

L T P C
2 0 2 3

OBJECTIVES:

- To understand the need and characteristics of mobile applications.
- To design the right user interface for mobile application.
- To understand the design issues in the development of mobile applications.
- To understand the development procedure for mobile application.
- To develop mobile applications using various tools and platforms.

UNIT I INTRODUCTION

Mobile Application Model – Infrastructure and Managing Resources – Mobile Device Profiles – Frameworks and Tools.

12

UNIT II USER INTERFACE

Generic UI Development - Multimodal and Multichannel UI – Gesture Based UI – Screen Elements and Layouts – Voice XML.

12

Lab Component:

- Implement mobile application using UI toolkits and frameworks.
- Design an application that uses Layout Managers and event listeners.

UNIT III APPLICATION DESIGN

Memory Management – Design Patterns for Limited Memory – Work Flow for Application development – Java API – Dynamic Linking – Plugins and rule of thumb for using DLLs – Concurrency and Resource Management.

12

Lab Component:

- Design a mobile application that is aware of the resource constraints of mobile devices.
- Implement an android application that writes data into the SD card.

UNIT IV MOBILE OS**12**

Mobile OS: Android, iOS – Android Application Architecture – Android basic components – Intents and Services – Storing and Retrieving data – Packaging and Deployment – Security and Hacking.

Lab Component:

- i. Develop an application that makes use of mobile database
- ii. Implement an android application that writes data into the SD card.

UNIT V APPLICATION DEVELOPMENT**12**

Communication via the Web – Notification and Alarms – Graphics and Multimedia: Layer Animation, Event handling and Graphics services – Telephony – Location based services

Lab Component:

- i. Develop web based mobile application that accesses internet and location data.
- ii. Develop an android application using telephony to send SMS.

TOTAL: 60 PERIODS

UNIT I

Introduction – Mobile Application Model – Infrastructure and Managing Resources – Mobile Device Profiles – Framework and tools

INTRODUCTION

What is mobile application development?

Mobile application development is the set of processes and procedures involved in writing software for small, wireless computing devices, such as smartphones and other hand-held devices.

MOBILE APPLICATION MODEL

WRITE SHORT NOTES ABOUT MOBILE APPLICATION MODEL.(PART B,C)

- The biggest advance in **mobile phone development was the introduction of Java- hosted MIDlets.**
- MIDlets are executed on a **Java virtual Machine (JVM) a process that** abstract the **underlying hardware and lets developers create application that run** on the wide variety of devices that support the java virtual machine

What is mobile application development?

Mobile application development is the set of processes and procedures involved in writing software for small, wireless computing devices, such as smartphones and other hand-held devices.

TYPES OF MOBILE APPLICATION TECHNOLOGIES

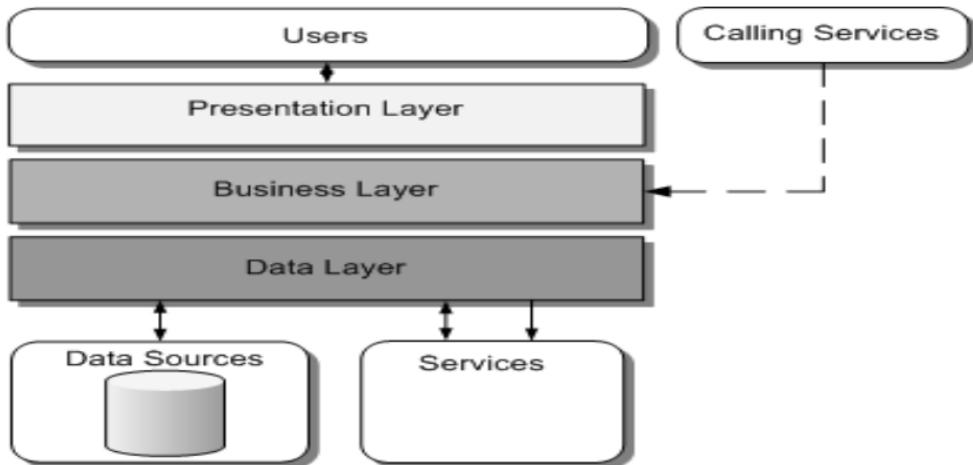
- **Native applications.** These applications are built using **integrated development environments** (IDEs) and languages for mobile OS such as Apple iOS or Google Android.
- **Hybrid apps.** These are web apps that act like native apps. They are developed using technologies such as **HTML, JavaScript and Cascading Style Sheets (CSS)**. Hybrid apps are more **cost-effective to develop than native apps and can be created faster**, but they aren't as feature-rich as native applications.

- **Progressive web apps.** A PWA is a website that looks and behaves as if it is a mobile app. These applications are developed with web technologies such as Facebook React.
- **Encapsulated apps.** An encapsulated app runs within a container app. Products such as the Microsoft Power App drag-and-drop app creation tool enable less experienced developers to build a mobile application rapidly. But the lack of isolation from the core OS, OS lock-in and the relative newness could pose problems.
- **Frameworks and libraries.** we can use this reusable code written by someone else to accelerate your development of a mobile app.



What is Mobile App Architecture?

Application architecture is a set of technologies and models for the development of fully-structured mobile programs based on industry and vendor-specific standards.



Mobile app architecture design usually consists of multiple layers, including:

- **Presentation Layer** - contains UI components as well as the components processing them.
- **Business Layer** - composed of workflows, business entities and components.
- **Data layer** - comprises data utilities, data access components and service agents.

THINGS TO CONSIDER BEFORE ATTEMPTING MOBILE APP ARCHITECTURE DEVELOPMENT

Determine the device type

There are different types of smartphones and it is important to evaluate the device type and its characteristics before choosing a specific app architecture.

- **Screen resolution**
- **Screen size**
- **CPU Features**
- **Storage Space**
- **Memory**
- **Availability of the development framework**

Presentation Layer

The main focus of this layer is **how to present the app to the end user**. When designing it, app developers must determine the correct client type for the intended infrastructure. Client deployment restrictions should also be kept in mind.

This layer is **choosing the correct data format and using powerful data validation techniques** to protect your apps from invalid data entry.

Business layer

Caching, logging, authentication, exception management and security are all matters of concern. According to our developers, you need to split tasks into different categories to reduce the complexity of this layer.

For complex rules, app policies, data transformations and validation, you must identify the set of demands separately for each of the categories.

Data Access Layer

This layer complies with the app requirements **to facilitate secure data transactions.** We must design this dimension so that it can be rescaled over time as business needs change.

INFRASTRUCTURE AND MANAGING RESOURCES

EXPLAIN BRIEFLY ABOUT INFRASTRUCTURE AND MANAGING RESOURCES (PART B, C)

ANDROID:AN OPEN PLATFORM FOR MOBILE DEVELOPMENT

More recently, Android has expanded beyond a pure mobile phone platform to provide a development platform for an increasingly wide range of hardware, including tablets and televisions. Put simply, Android is an ecosystem made up of a combination of three components:

- a. **A free, open-source operating system for embedded devices**
- b. **An open-source development platform for creating applications**
- c. **Devices**

More specifically, Android is made up of several necessary and dependent parts, including the following:

- ❖ **A Compatibility Definition Document (CDD) and Compatibility Test Suite (CTS)** that describe the capabilities required for a device to support the software stack.
- ❖ A **linux operating system kernel that provides a low-level interface with the hardware, memory** management, and process control, all optimized for mobile and embedded devices.
- ❖ Open source libraries for application development, including SQLite, WebKit, OpenGL, and a media manager.

- ❖ A runtime used to execute and host Android applications, including the Dalvik Virtual Machine (DVM) and the core libraries that provide Android-specific functionality. The run time is designed to be small and efficient for use on mobile devices.
- ❖ A user interface framework used to stand launch applications.
- ❖ A set of core installed applications.
- ❖ A SDK(software development kit)used to create applications including the related tools, plug ins and documentation.

WHAT DOES ANDROID RUN ON?

- ❖ The first Android mobile handset, the T-Mobile G1, was released in the United States in October 2008.
- ❖ Android is created to support variety of platform and applications.

WHY DEVELOP FOR ANDROID?

Android represents a clean break, a mobile framework based on the reality of modern

mobile devices designed by developers, for developers.

The barrier to entry for new Android developers is minimal:

- No certification is required to become an Android developer.
- Google Play provides free, up-front purchase, and in-app billing options for distribution and monetization of your applications.
- There is no approval process for application distribution.
- Developers have total control over their brands.



WHAT ANDROID HAS THAT OTHER PLATFORMS DON'T HAVE

- Google Maps applications
- Background services and applications
- Shared data and inter-process communication
- All application are created equal
- Wi-Fi Direct and Android Beam

INTRODUCING THE DEVELOPMENT FRAMEWORK

- Android applications normally are written using Java as the programming language but executed by means of a custom VM called **Dalvik,rather than a traditional Java VM.**
- Each Android application runs in a separate process with in its own Dalvik instance, relinquishing all responsibility for memory and process management to the Android runtime, which stops and kills processes as necessary to manage resources.

WHAT COMES IN THE BOX

- **The Android APIs**—The core of the SDK is the Android API libraries that provide developer access to the Android stack. These are the same libraries that Google uses to create native Android applications.
- **Development tools** — The SDK includes several development tools that let you compile and debug

your applications so that you can turn Android source code into executable applications.

- **The Android Virtual Device Manager and emulator**—The Android emulator is a fully interactive mobile device emulator featuring several alternative skins. The emulator runs within an Android Virtual Device(AVD)that simulates a device hardware configuration.
- **Fulldocumentation**
- **Samplecode**
- **Onlinesupport.**

UNDERSTANDING THE ANDROID SOFTWARE STACK

TheAndroidsoftwarestackis,putsimply,aLinuxkernelandacollectionofC/C++libraries exposed through an application framework that provides service for, and management of, the runtime and applications.

- **Linuxkernel**

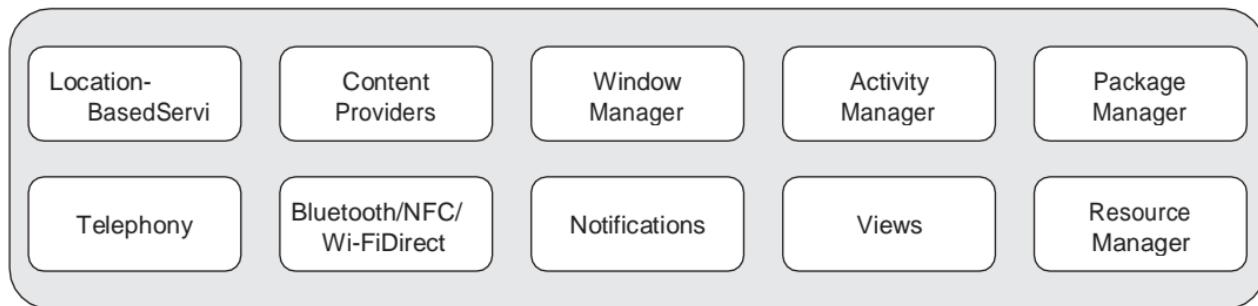
Coreservices(includinghardwaredrivers,processandmemorymanagement,security,network, and powermanagement) are handled by a Linux2.6 kernel.
The kernel also **provides an abstraction layer between the hardware and the remainder of the stack.**
- **Libraries**—Running on top of the kernel, Android includes various C/C++ core libraries such as

libc and SSL.

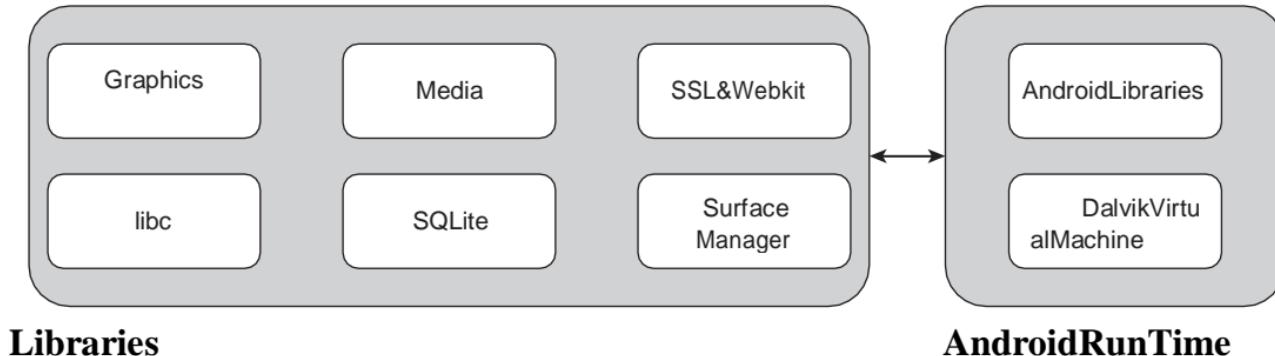
- **Android runtime**—The runtime is what makes an Android phone an Android phone rather than a mobile Linux implementation. Including the core libraries and the DalvikVM, the **Android runtime is the engine that powers your applications and, along with the libraries, forms the basis for the application framework.**
- **Core libraries**—Although most Android application development is **written using the Java language**, Dalvik is not a JavaVM. The core Android libraries provide most of the functionality **available in the core Java libraries, as well as the Android-specific libraries.**
- **Dalvik VM**—Dalvik is a register-based Virtual Machine that's been optimized to ensure that a **device can run multiple instances efficiently.** It relies on the Linux kernel for threading and low-level memory management.
- **Application framework** — The application framework provides the classes used to create Android applications. It also provides a **generic abstraction for hardware access and manages the user interface and application resources.**
- **Application layer**—All applications, both native and third-party, are built on the application layer by means of the same API libraries. The application layer runs within the Android runtime, using the classes and services made available from the application framework.



ApplicationLayer

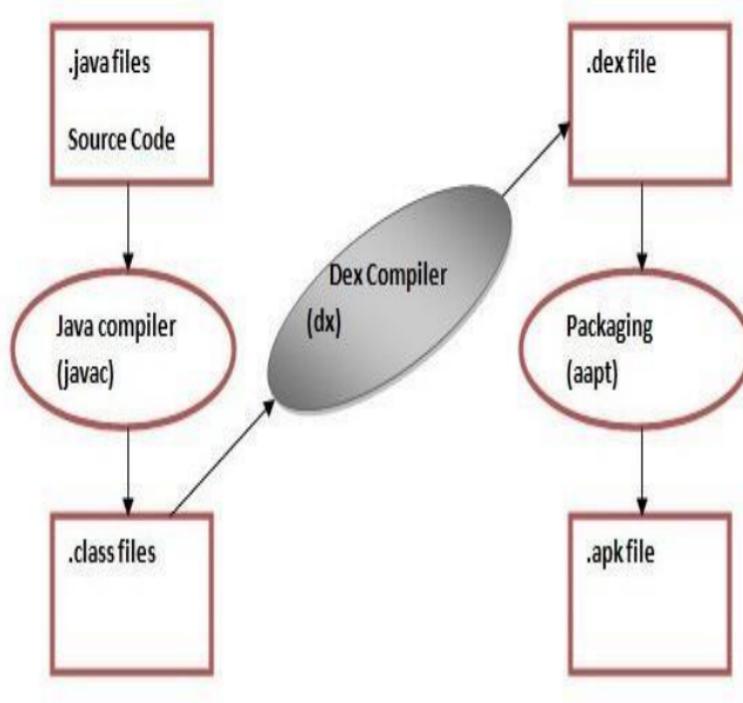


ApplicationFramework



The Dalvik Virtual Machine

- The **Dalvik Virtual Machine (DVM)** is an android virtual machine optimized for mobile devices. It optimizes the virtual machine for memory, battery life and performance.
- Dalvik is a name of a town in Iceland. The Dalvik VM was written by Dan Bornstein.
- The Dex compiler converts the class files into the .dex file that run on the Dalvik VM. Multiple class files are converted into one dex file.
- Let's see the compiling and packaging process from the source file:



Android Application Architecture

- Android's architecture encourages component reuse, enabling you to publish and share Activities, Services, and data with other applications, with access managed by the security restrictions.

Activity Manager and Fragment Manager

- **Views**—Used to construct the user interfaces for your Activities and Fragments.
- **Notification Manager**—provides a consistent and non intrusive mechanism for signaling your users.
- **Content Providers**—Lets your applications share data.
- **Resource Manager**—Enables non-code resources, such as strings and graphics, to be externalized.
- **Intents**—Provides a mechanism for transferring data between applications and their components.

MANAGING RESOURCES

WHAT ARE THE KINDS OF RESOURCES AVAILABLE FOR MOBILE DEVELOPMENT?(PART B, C)

A mobile device is a specialized piece of hardware. It has several different types of resources that may require special monitoring and management activities that require scalable yet uniform designs. Android resources are basically files or external data supporting our app's operation. These files can be images, strings, colours, styles, etc.

KINDS OF RESOURCES FOR ANDROID APP DEVELOPMENT

1. Animation Resources

- Their purpose is to **set default animations.**
- They are stored in the res/drawable/ folder under the R.drawable identifier. They support bitmap files (.png, .jpg or .gif), PNG files in Nine-patch (.9.png) format and XML files with graphics descriptors.

2. Colour State List Resources

- For determining a **component's colour depending on its state.**
- Stored in res/color/ and accessed through R.color.

3. Drawable Resources

- For **defining bitmap or XML graphics.** They are stored in the res/drawable/ folder under the R.drawable identifier.

4. Layout Resources

- **For defining an app's interfaces. They contain XML files** and are stored in res/layout/ under ID R.layout.

5. Menu resources

- For defining the contents of **an app's menus**. They are stored in res/menu/ and accessed through R.menu.

6. String Resources

- **They contain XML files with data embedded in strings or in string arrays** (including the string's format and style).
- They can be sorted into strings.xml, colors.xml, dimens.xml, styles.xml or arrays.xml and are stored in res/values/ and include R.string, R.array, and R.plurals.

7. Style Resources

- They define a **number of attributes that can be applied to a view or an activity**. Also used for **defining an app's style and design**. They are stored in res/values and accessed through R.style.

8. Other types:

- We'll be defining values as being Boolean, integer, dimensions, colours and other arrangements.
- They are stored in res/values/ but each of them is accessed through unique R subclasses (such as R.bool, R.integer, R.dimen, etc.).

MOBILE DEVICE PROFILES

WHAT DOES MOBILE INFORMATION DEVICE PROFILE (MIDP) MEAN?(PART B,C) DEFINE MIDP (PART A)

What is Java ME?

- **The Java ME stands for Java MicroEdition.**
- It is a development and deployment platform of portable code for embedded and mobile devices (sensors, gateways, mobile phones, printers, TV set-top boxes).
- The Java ME has a **robust user interface, great security, built-in network protocols, and support for applications that can be downloaded dynamically.**
- **Applications which are developed on Java ME are portable** and can run across various devices and can also leverage the native capabilities of the device.

Java ME SDK

- Java ME Software Development Kit (SDK) provides the standalone runtime environment and various utilities required for development Java ME applications.
- It combines the Connected Limited Device Configuration (CLDC) and the Connected Device Configuration (CDC) into one single environment.

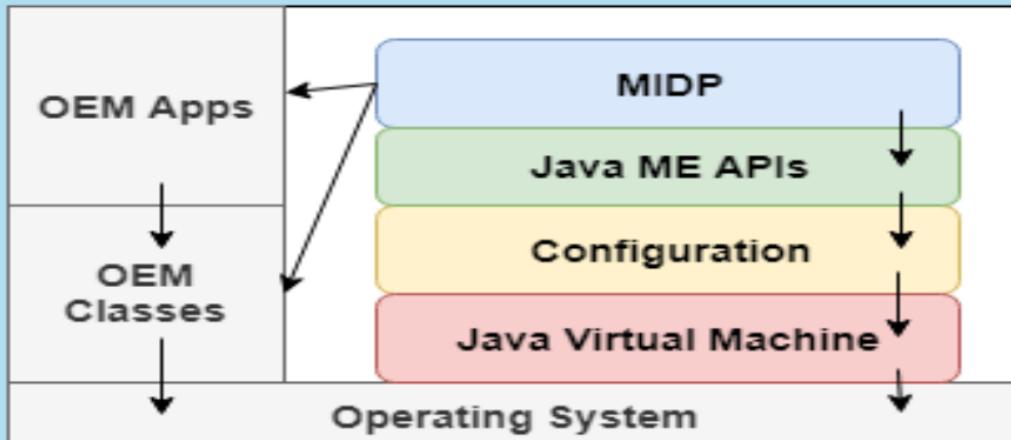
JAVA ME ARCHITECTURE

- Java ME does not simply replace the operating system, rather it stacks up layers on the native operating system and makes an environment for the application to run.
 - These layers are collectively named as **Connected Limited Device Configuration (CLDC)**.
1. The first layer is the configuration layer that includes the Java Virtual Machine. This layer interacts directly with the native operating system and builds the connection between the profile and the JVM.
 2. The second layer is the profile which contains the minimum set of APIs for the small computing device. The profile contains a set of classes which are made to implement the features of a related group of small computing devices.

3. The third layer is the Mobile Information Device Profile (MIDP). The MIDP layer consists of APIs which are for user network connections, persistence storage, and the user interface.

It also has access to Connected Language Device Configuration (CLDC) and Mobile Information Device Profile (MIDP) libraries.

- A small computing device has two components supplied by the Original Equipment Manufacturer (OEM). They are namely OEM apps and OEM classes.
- The MIDP communicates with the OEM classes to gain access to features like sending and receiving messages and accessing device-specific persistent data.
- OEM applications are small programs such as address book etc.

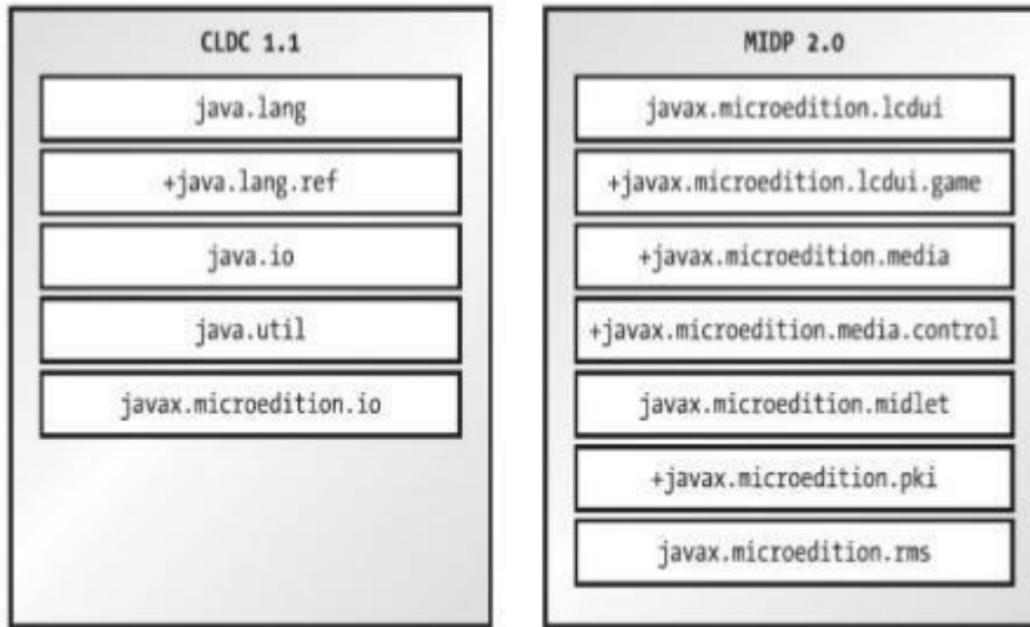


Layers of the Java ME architecture

MIDI

- Mobile Information Device Profile (MIDP) is a specification for the use of Java technology for mobile devices.
- MIDP sits on top of the Connected Limited Device Configuration (CLDC).
- Because MIDP is primarily used with CLDC, which is designed for highly constrained devices with limited CPUs, screen size, RAM, battery power and user interface, midlets are ideal for low-end cell phones.
- MIDP is part of the Java™ 2 Platform, Micro Edition (J2ME™).
- An application that runs in the MIDP environment is called a MIDlet.
- The first MIDP devices were launched in April 2001.

Anatomy of MIDP Apps



GENERAL API

- The core application programming interfaces are defined by the underlying Connected Limited Device Configuration system.
- **javax.microedition.io**→Contains the **Java ME-specific classes used for I/O operations.**
- **javax.microedition.lcdui**→Contains the **Java ME-specific classes used for the GUI.**
- LCDUI has a simple screen based approach where a single Displayable is always active at a time in the application user interface. **LCDUI API provides a small set of displayable common in mobile device user interfaces: List, Alert, Textbox, Form and Canvas.**
- LCDUI also has a **quite unique approach of abstract operations, called Commands.** The placement of commands added to a displayable is completely up to the device implementation of this toolkit. The application programmer uses API specified command types to indicate the usage or purpose of the command in an application user interface. Common types are BACK, EXIT, ITEM, SCREEN.

- LCDUI → Limited Capability Device User Interface
- javax.microedition.rms → The Record Management System provides a form of persistent storage for Java ME; a database for the mobile device.
- javax.microedition.midlet → Contains the base classes for Java ME applications, and allows applications to be notified of changes to their state.

OPTIONAL JSR

javax.microedition.messaging → Wireless messaging API (optional), for sending SMS and MMS messages.

javax.microedition.pim → Personal information management API (optional), access the device's Address Book, to-do List, Calendar.

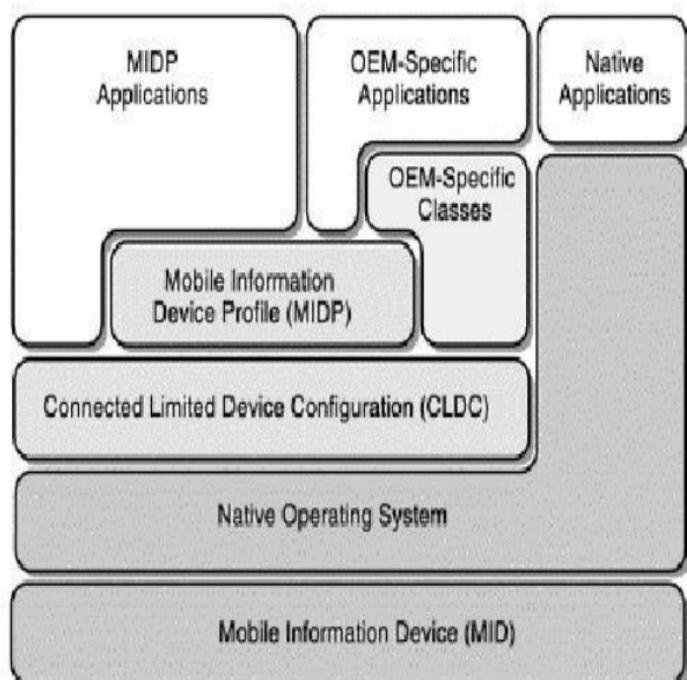
javax.microedition.io.file → The File Connection Optional Package (FCOP) is one of two optional packages defined by JSR 75 through the Java Community Process. The FileConnection API specified in JSR 75 gives access to the local file systems on devices like PDA. In order to overcome security issues MIDlet needs to include requested file permission in its JAD file under MIDlet-Permission property.

Development tools → There are several different ways to create MIDP applications: code can be written in a plain text editor, or one can use a more advanced IDE such as NetBeans, IntelliJ , or Eclipse

HISTORY

- MIDP was developed under the Java Community Process.
- MIDP 1.0 (JSR 37) - Approved on September 19, 2000
- MIDP 2.0 (JSR 118) - Approved on November 20, 2002
- MIDP 3.0 (JSR 271) - Approved on December 9, 2009
 - A. Shared libraries for midlets
 - B. Improved cross-device interoperability
 - C. Improved UI expressability and extensibility
 - D. Support of devices with larger displays

MID Profile Architecture



- E. High-performance games
- F. Auto-launched midlets
- G. Inter-midlet communications
- MIDP is now succeeded by MEEP as of Java ME 8.

FRAME WORK AND TOOLS

WHAT IS MEANT BY FRAMEWORK? (PART A)

DEFINE THE USAGE OF FRAME WORK. (PART B)

EXPLAIN THE TYPES OF FRAME WORKS AVAILABLE (PART B, C)

- A framework, or software framework, is a platform for developing software applications. It provides a foundation on which software developers can build programs for a specific platform.

- For example, a framework may include predefined classes and functions that can be used to process input, manage hardware devices, and interact with system software.
- **A framework is similar to an application programming interface (API), though technically a framework includes an API.**
- Depending on the operating system that is selected, the developer would then create the app using a corresponding language.
- All you need is to code your app using the languages you are comfortable with and the framework will render a hybrid version of your app that works across multiple platforms.

1. Cordova/PhoneGap

- **Designed by Adobe, PhoneGap is a combination that not only allows developers to create their apps using HTML, CSS and JavaScript, but also package the app, make it portable and release it to multiple app stores.**
- It can also extend the features of HTML and JavaScript, to create apps with a Native look and feel.
- Cordova **provides a set of JavaScript APIs** that allow the apps to use native functions such as Camera, Compass, Contacts, and Geo location. And it supports multiple operating systems such as iOS, Android, Windows Phone, Web OS, Blackberry, etc.

Benefits of Cordova/Phone Gap: -

- **Fast:** The strong and robust back-end ensures that the apps created are super fast and do not have any language.
- **Cheap:** Since, Cordova is open source, this means that you don't have to pay any money to download and run the software.
- **Flexible:** PhoneGap eliminates the need for multiple languages, allowing developers with even limited knowledge of HTML, CSS and JavaScript to build some pretty amazing languages.
- **Cross-platform compatibility:** PhoneGap is compatible with multiple operating systems including iOS, Android, Windows, etc.
- **Native Hardware:** It allows access to native hardware components, which means that the apps run with a native look and feel.
- **Built-in Compiler:** In addition to creating the apps, PhoneGap also comes with a built-in compiler, which takes the pain out of separate compiling.
- **Collaboration:** Similar to GitHub, PhoneGap now allows developers to share their coding and get help from other other developers.

2. Ionic/Ionic 2

- **Ionic is one of the most popular hybrid frameworks that is currently on the market.**
- It uses the JavaScript MVVM framework and AngularJS, which power the framework, making it faster and better than the other frameworks on the list.
- **The best part about Ionic is that it provides all the functionality that can be found in native mobile development SDKs,** which means that when users build their apps on Ionic, they can access all the native functions of the device, including camera, contacts, etc.
- Additionally, by using Angular, Ionic provides custom components and methods for interacting with them.

Benefits of Ionic/Ionic 2:

- **Free:** Ionic is an open source project under MIT and has been designed by a worldwide community.
- **Fully Cross-Platform:** Ionic follows the write once, run anywhere philosophy, which means that apps written on Ionic can be run on multiple operating systems including iOS, Android, Windows Phone and Blackberry.
- **Premier Native Plugins:** Ionic offers over 120 native device features that can be used within your app.

- **Performance:** Ionic places a special focus on performance, by leveraging hardware and requires no additional third party JS applications.
- **Native and web optimized:** Emulates native app UI guidelines and uses native SDKs, combining the UI standards and device features of native apps together with the full power and flexibility of the open web.

3. jQuery Mobile

- **jQuery Mobile is a great mobile framework to create cross-platform apps for platforms** such as regular desktop, smart phone, tablet, or an e-reader device like Nook or Kindle.
- Built on top of solid jQuery and jQuery UI foundation, jQuery Mobile is a great framework to develop apps, mobile and web, to run seamlessly with unique user experience across mobiles, tablets and desktops. Rather than focus on providing the native look, jQuery focuses on performance.

Benefits of jQuery Mobile:

- **Cross-Platform and Cross-Device:** With a focus on ‘write-less, do more’ mantra, jQuery is not only a great framework for creating apps on multiple operating systems, but also for multiple devices including smartphones, desktops, tablets, etc.
- **Easy to Learn:** If you already know jQuery and JavaScript, you don’t have to learn any new syntax.

- **ThemeRoller:** jQuery is designed as a theming framework, which allows developers to create customized themes for their apps.
- **Download Builder:** jQuery's download builder allows building a custom bundle that contains only the components that are needed. The builder also generates a custom JavaScript file and full and structure-only stylesheets for production use.
- **Layout Grid:** jQuery's layout grids make it easy to create client's product page, result page, and other custom pages.

4. React Native

- React Native separates itself from the rest of the frameworks on the list by shifting its focus from creating hybrid apps to creating actual real native apps.
- **ReactNative was designed on the heels of React, by Facebook** and it stemmed from their Ads Manager app.
- **React Native comes with fast development cycles and declarative self-contained UI components, while retaining the speed and feel of native apps.**
- The application logic is written and run in JavaScript, while the UI is fully native. This offers developers a chance to maintain the native look and feel of the app, without having to use traditional languages.

Benefits of React Native:

- **Parallelize Work:** Facebook's Async display kit allows rendering off the main thread, which results in super smooth animations.
- **Declarative Style:** The codes can now be written in a declarative format, which means the codes are now readable, flexible and not manipulative.
- **Sophisticated Gesture Handling:** Reactive Native allows access to native gesture functionality, so it results in better gestures on the app.
- **Native Capabilities:** Allows access to platform specific capabilities and components, including widgets.
- **Faster Iterations:** React Native allows Hot Reloading and Live Reloading, reducing feedback loop and also offers over-the-air code updates to your app.
- **Cross-Platform:** An app can be designed across multiple platforms, mainly iOS and Android.

5. Meteor

- Meteor is an amazing open-source JavaScript web framework that is written using Node.js. It simplifies the process of developing apps by allowing rapid prototyping and producing cross-platform code.

- A full JavaScript, Meteor is made up of a collection of libraries and packages that are bound together, making it easier, flexible, faster and it requires less codes. This also results in the codes being less buggy and of a higher quality.

Benefits of Meteor:

- **Fast:** Using JavaScript for front-end and back-end development, creating apps using Meteor is simple and easy. It also allows you to ship more with less code, reducing the lines of code required for building apps.
- **One Language Development:** For JavaScript lovers, this is a great advantage. No more multiple languages for front-end and back-end development.
- **Easy to Learn:** If the developer already knows JavaScript, then they don't have to learn anything new. However, if they don't, then JavaScript is a pretty easy language to learn.
- **Real-time Applications:** Real-time is built in Meteor's core, which means developers get to produce real-time apps right out of the box.
- **Packages:** Already built smart packages to simplify coding and make building faster, for items such as User accounts, Javascript libraries, Extras like Bootstrap or Stylus, and more.

Some other popular Cross-platform Frameworks for Mobile Development are:

1. Corona SDK
2. Xamarin
3. Appcelerator Titanium

4. TheAppBuilder
5. NativeScript
6. Sencha Touch

TOOLS

What are Mobile Development Tools?

- Mobile Development Tools are software designed to assist in the creation of mobile applications. This can be accomplished in multiple ways, for example, there are ***native mobile development tools***, but also ***cross-platform mobile development tools***.
1. **Native mobile development tools** can help you create specialised apps that operate with ease and high quality, and can take advantage of all features on their designated platform.
 2. **Cross-platform mobile development tools** – on the other hand – make it possible to create a generic app for multiple platforms simultaneously, greatly cutting the costs and time needed to create an app, but this comes with a trade-off. Non-platform specific applications made in a cross-platform environment tend to have more issues and lower quality compared to native applications.

Native Mobile Development Tools

A **native development tool** is a software which allows developers to create applications for use in a single particular system family, platform or device, like Android, iOS, or Windows (note: support for Windows Mobile ends in December 2019). A native app is specially made and coded for a specific mobile platform in its native programming language, these being:

- iOS (Objective-C or Swift)
- Android (Java or Kotlin)
- Windows Phone (C#)

1. Xcode

- Xcode introduces a new way to design and build software.
- Live rendering within Interface Builder displays your hand-written UI code within the design canvas, instantly reflecting changes you type in code.
- Xcode includes everything developers need to create applications for Mac, iPhone, iPad, Apple TV, and Apple Watch. Xcode provides developers a unified workflow for user interface design, coding, testing, and debugging.

2. Android Studio

- Android Studio is an Android development Software built by Google. Its implementation editor is very useful for Android developers.
- Android studio provides shortcuts for coding and designing and its layout designer makes it very easy to use, which helps reduce time spent on coding. Android studio also provides drag and drop features to design the layout of your projects.

3. AppCode

- AppCode is an IDE for iOS/macOS development. In addition to working with Objective-C, Swift and C/C++ programming languages, it supports web technologies such as JavaScript, HTML, XML, CSS, and more.
- It provides a variety of valuable integrations including among others CocoaPods manager and built-in Reveal support.
- In addition to the benefits AppCode provides to developers (such as saving their time on automating routine tasks, locating and fixing errors, taking advantage of intelligent support from the IDE, and increasing their overall productivity), it can be an equally valuable asset for your business.

Cross-Platform Mobile Development Tools

- With **cross-platform mobile development**, programmers on one platform can develop apps for one or more other platforms or mobile operating systems simultaneously.
- This can also enable developers to essentially use the same code base for different platforms. Meaning that such generic apps can be published and used on both an Android Phone and an iPhone.

Furthermore, this category has also been split into three platforms:

- **Coding Platforms –Appcelerator,xamarin, ionic, react native**
- **Low-Coding Platforms**
- **No-Coding Platforms**

REFERENCE

1. Reto Meier, Professional Android 4 Application Development:, Wiley, 1st edition,2012
2. ZigurdMednieks, Laird Dominm G. Blake Meike, Masumi Nakamura, -Programming Androidl, O'Reilly , 2nd edition,2012
3. Alasdair allan,iphone Programmingl, O'Reilly , 1st edition,2010.

*****UNIT I COMPLETED*****

UNIT II

USER INTERFACE

GENERIC UI DEVELOPMENT – MULTIMODAL AND MULTI CHANNEL UI – GESTURE BASED UI – SCREEN ELEMENTS AND LAYOUT – VOICE XML

INTRODUCTION

GENERIC UI DEVELOPMENT

The latest generation of mobile devices is portable enough to carry at all times, connected to voice and data networks and contextually aware by using sensors and networks to preemptively complete tasks.

Mobile application creators can also use exciting new interactions with motion and gestures: zooming, swiping, tapping, turning, and shaking. These capabilities offer the chance to innovate.

—Why do we want to build generic user interfaces?!!

- First, mobile applications are typically used by a wider array of operating environments and systems than a PC. Because there is such a wide array of end clients for mobile applications, we need to be able to adapt the application quickly, if not in real time, with very little or no additional development effort.
- The mechanism through which user's access a software application is referred to as the user interface.
- First, there are the human factors. The way it is used has a great impact on the utilization of any computing system. Though defining human factors in one sentence is a difficult task, it can be defined as the set of those concerns qualifying the interaction of the user with the software system.
- For any typical application, we need to consider the following as the elements of human factors consideration:

- 1 The look and feel of the application and how the users —like the user interface,
 - 2 The ease of learning the interface well and becoming efficient at using the user interface, and
 - 3 Health issues in using the user interface
- Another factor is to think of what a typical user considers desirable.

- Once again, this might include a variety of esthetics, timing, color, etc. It is also important to keep in mind that most user interfaces are typically **used many times, not just once.**

HUMAN FACTORS AND OTHER CONSIDERATION FOR DEVELOPING UI

1. Intuitiveness
2. Consistency
3. Learnability
4. Nonintrusively Helpful
5. Accommodating Expert Users(shortcuts)
6. Trustable
7. Robustness (Recover from Error)

Other consideration

1. Short transaction cycles
2. Expectation of consumer devices (Boot up)
3. Lack of focus
4. Intermittent network connectivity
5. Multichannel user interface

BUILDING GENERIC USER INTERFACE

DESCRIBE BUILDING USER INTERFACE(PART C)

The reason for building a generic user interface for mobile systems is the wide variety of devices and user interfaces that an application might need to support. The idea here is to layer the different parts of the user interface, build a generic user interface, and then specialize it to a given device or type of user interface.

Consider applications that can benefit from a layered user interface approach:

- 1. Applications that change frequently**

Many applications change very frequently. Such constant changing of state may be caused by the business model that the application serves or a variety of other reasons. The life of a software application is only meaningful as long as it is serving its economic or scientific reason for existence.

- 2. Applications that support a wide variety of devices**

The advantage of building a generic interface to a system, and then specializing it, is that it minimizes the amount of code needed to perform the necessary tasks. And we all know that the less code we have, the better off we are. This is probably the most popular reason for building generic user interfaces.

- 3. Applications that must have many loosely coupled parts**

One of the advantages of building a generic user interface to a system is that it enables loose coupling between the user interface components themselves and among the look-and-feel

components, interaction logic, and application flow logic. This loose coupling is one of the principle features of various existing technologies such as XSL templates and CCS.

4. **Applications that offer multiple user interfaces with a range of complexity:**

A good reason to justify building systems with generic user interfaces is the requirement of supporting multiple user interfaces, each with some difference in the required feature sets

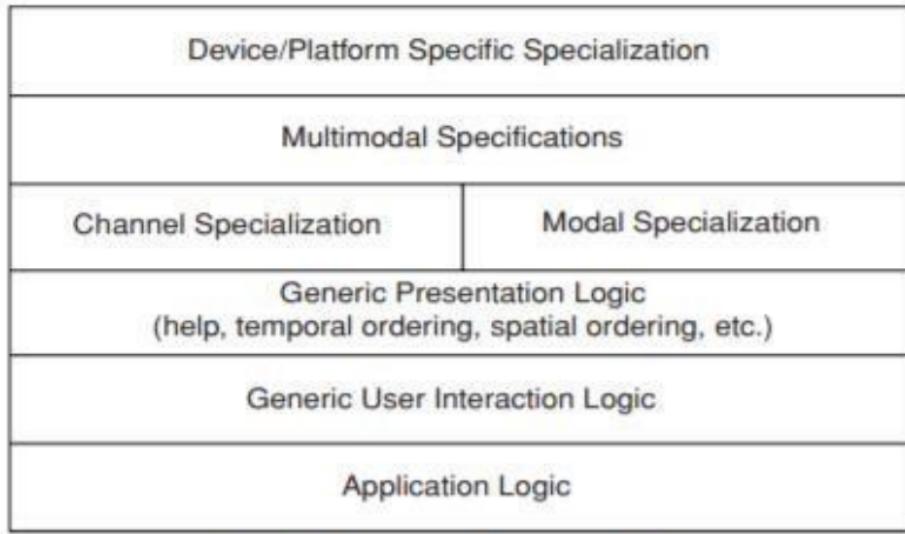


FIGURE 5.1. Layering User Interfaces.

BINDING AND SPECIALIZATION OF GENERIC USER INTERFACES

WHAT ARE THE FACTORS USED IN DESIGNING USER INTERFACE ?(PART A, B)

- The generic interface has to be specialized to the specific intended user interface. The specialization process could include user settings, device settings, discovery of available channels, QOS of the network, and many other factors. In any case, a binding process has to take place between the components that produce the generic user interfaces and the components that specialize in the generic interfaces.
- This binding can be done at run time for all user interfaces; alternatively, what binding is possible to do at compile time may be done then and the rest done at run time.

There are a number of factors that must be considered when designing a system this way:

1. **Performance:** As a rule of thumb, layering software and performance are inversely proportional. The key here is to first evaluate the performance needs of the users and then to see what the cost of each additional layer is. The extreme case is when every microsecond counts
2. **Development Process:** a typical development team includes business analysts who gather the requirements, graphic and voice user interface designers who design the look- and-feel of the system, application developers who build the components that represent the business logic to be performed by the system, and database engineers who design the persistence layer.

3. **Where the Various Components Reside:** Probably the most important aspect of the design of a system that uses generic user interfaces is how the work is distributed among the servers, clients, or peers.

THE ELEMENTS OF THE USER INTERFACE

WRITE DOWN THE ELEMENTS OF USER INTERFACE(PART B,C)

- User interfaces can be **defined from several perspectives**, none of which is inherently superior to the others.
- Some analyze user interfaces from a **look-and-feel perspective**, some look at them as a **component in a communication system between humans and computers**, and yet others look at the various functions that they (user interfaces) and their components perform.

1. Channels

- In the generic context of communication systems, a channel is the **medium through which the sender and the receiver of a message communicate**. In this way, a user interface that uses multiple media such as sound, text, and video are often referred to as multichannel interfaces as each type of medium requires its own channel.

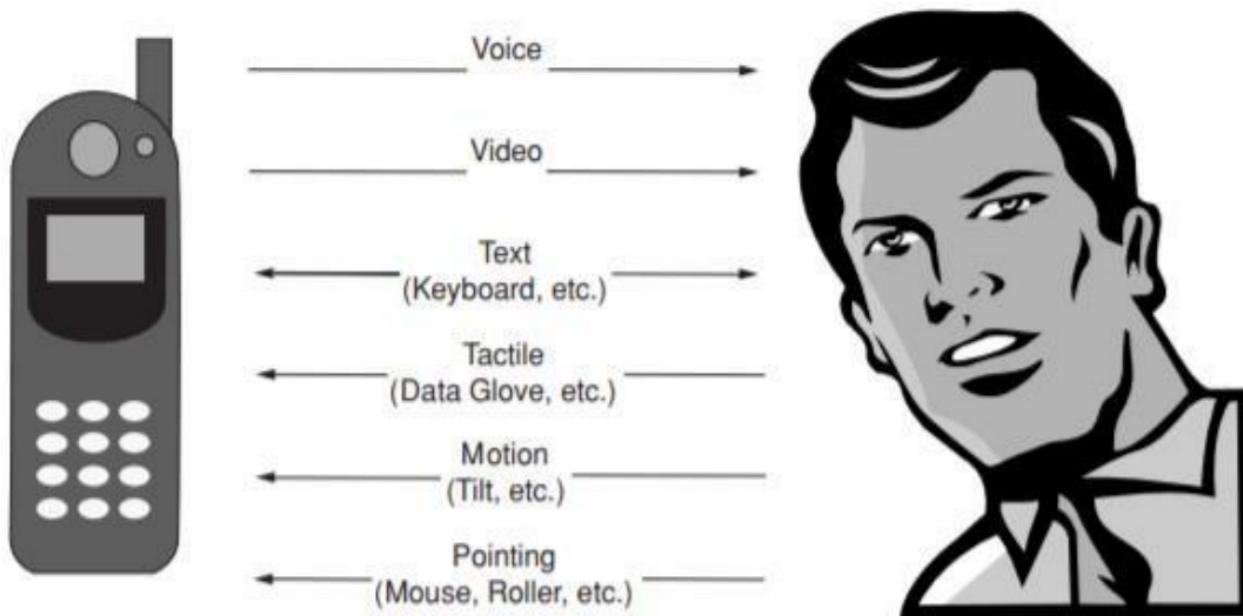


FIGURE 5.2. Channels: Tunnels of Communication between Humans and Computers.

- In the computing industry, this definition has also been extended to include presenting the same content, particularly text content, in multiple formats for multiple devices.
- Because various devices may render the same textual content in different ways, with a bit of a reach, we can say that each type of rendered text is part of the channel that sends messages from the system to the user.

2. Channel types

1. Keyboards and Monitors (Text Entry and GUI Display):

a. Keyboards and monitors are perhaps the two most popular user interface mechanisms to any computing system. Keyboards are basically used to enter text that may be behavioral in nature (commands such as save, print, etc.) or data (such as typing in your user name and password).

2. Touch-Screens (Touch Entry and GUI Display):

a. Touch-screens have been around for a long time. In many environments such as manufacturing environments or mobile applications, a keyboard, a mouse, and other extraneous devices may not be suitable. They may be too cumbersome to use or be damaged too frequently because of dust, motion, or other environmental conditions. Touchscreens allow the user to touch pressure-sensitive screens to send messages to the system.

3. Stylus (Handwriting Recognition and Touch Entry):

- a. The stylus is a penlike device used to write on a screen or press buttons on a screen. The stylus effectively functions as a keyboard and a mouse combined into one. The stylus is a very effective device for mobile environments. The stylus is light and enables multiple input channels to the same device. The stylus provides only an input channel. The output channel accompanying the stylus, once again, is typically some sort of a visual screen.
4. **Telephone** (Voice Recognition):
- a. The telephone is the most pervasive electronic communication device. Voice channels take advantage of our ability to be able to understand speech and hear sounds. M
5. **Device Motion (Entry by Positioning):**
- a. The position and orientation of the device itself can be used as an input mechanism.
6. **Dataglove** (Entry through Touch) [Dix et al. 1998]:
- a. As the name suggests, this is a glove that, when worn, can be used as a data entry device. This device is now used mostly in research environments, but it has made advances in performance and reliability during the past few years, making it more and more viable commercially
7. **Printed Paper and Other Materials (Output Text and Graphics):** In the early days of computing, paper was the main method of interfacing with computing systems.

3. Interactions

- Interactions, as mentioned, are composed of messages being passed back and forth between the user and the system. The messages being sent can take atomic or composite forms.
- An atomic interaction is made up of one message being sent from the user to the system, or from the system to the user, possibly accompanied by a response message from the respective counterpart
- Composite interactions are those interactions that may be made up of two or more interactions that are meaningful on their own.

Elements

1. Control message
2. Prompts
3. Responses

Types

1. Commands
2. Menus
3. Forms
4. Natural language

5. Mixed initiative

EFFECTIVE USE OF SCREEN REAL ESTATE

- The first step **to use the smaller interfaces of mobile devices effectively is to know the context of use.** Who are the users, what do they need and why, and how, when, and where will they access and use information?
- **Mobile design is difficult, as developers** try to elegantly display a telescoped view of almost limitless information.

Embrace Minimalism

- **Limit the features available on each screen, and use small, targeted design features.**
- Content on the screen can have a secondary use within an application, but the application designer should be able to explain why that feature is taking up screen space.

Use a Visual Hierarchy

- Help users **fight cognitive distractions with a clear information hierarchy.** Draw attention to the most important content with visual emphasis.

- A consistent hierarchy means consistent usability; mobile application creators can create a hierarchy with position, form, size, shape, color, and contrast.

Stay Focused

- Start with a focused strategy, and keep making decisions to stay focused throughout development.
- A smaller file size is a good indicator of how fast an application will load, so the benefits of fighting feature creep extend beyond in-application user experience.

UNDERSTANDING MOBILE APPLICATION USERS

- The Gestalt principles have had a considerable influence on design, describing how the human mind perceives and organizes visual data.
- The Gestalt principles refer to theories of visual perception developed by German psychologists in the 1920s.
- According to these principles, every cognitive stimulus is perceived by users in its simplest form.
- Key principles include
 - ❖ Proximity
 - ❖ Closure

- ❖ Continuity
- ❖ Figure and ground
- ❖ Similarity

Proximity

- ❖ Users tend to group objects together. Elements placed near each other are perceived in groups.
- ❖ Icons that accomplish similar tasks may be categorically organized with proximity. Place descriptive text next to graphics so that the user can understand the relationship between these graphical and textual objects.

Closure

- ❖ Closure If enough of a shape is available, the missing pieces are completed by the human mind.
- ❖ In perceiving the unenclosed spaces, users complete a pattern by filling in missing information.

Continuity



FIGURE 4-1: Proximity



FIGURE 4-2: Closure



FIGURE 4-3: Continuity

- ❖ The user's eye will follow a continuously-perceived object. When continuity occurs, users are compelled to follow one object to another.
- ❖ They perceive the horizontal stroke as distinct from the curled stroke, even though these separate elements overlap.
- ❖ Smooth visual transitions can lead users through a mobile application, such as a link with an indicator pointing toward the next object and task.

Figure and Ground

- ❖ A figure, such as a letter on a page, is surrounded by white space, or the ground.
- ❖ Complex designs can play with the line between -figure and -ground, but mobile interfaces speed user frustration with unclear distinctions.
- ❖ Primary controls and main application content should maintain a distinct separation between figure and ground.



FIGURE 4-4: Figure and ground

Similarity

- ❖ Similar elements are grouped in a semi automated manner, according to the strong visual perception of color, form, size, and other attributes.

UNDERSTANDING MOBILE INFORMATION DESIGN

Mobile devices offer an exciting space to design information, fitting personalized and real-time data into tightly-constrained screens.

INFORMATION DISPLAY

- **A microwave has a simple display.** When the timer alerts us the popcorn is done, we can check to see if the bag looks adequately puffed, and then open the microwave door and popcorn bag to start eating.
- **People identify signals, interpret the meaning of these signals, and determine the goal according to these interpretations, and then carry out an action until the goal is reached.**

DESIGN PATTERNS

WHAT ARE THE DESIGN PATTERNS USED IN UI DEVELOPMENT?(PART B,C)

- **Hardware and operating systems become irrelevant far quicker than design that reaches users.**



FIGURE 4-7: Annunciator panel

- A design pattern recycles and repurposes components, reusing the best ideas. More than time efficiency, patterns have been refined by use.

i) Navigation

Annunciator Panel

- An annunciator panel, seen at the top of Figure gives information on the state of a mobile device. Though each mobile device will provide a slightly different panel, developers can modify or suppress the annunciator panel — which lists the hardware features such as network connection and battery power — within an application.
- Because the information in this area is only notifications, application users will not usually have any direct interaction with the annunciator panel.

ii) Fixed Menu

A menu that remains fixed to the viewport as users roam content is useful in many situations:

- ❖ When users need immediate access to frequently selected functionality on multiple screens
- ❖ When a revealable menu is already used on the same screen
- ❖ When a lack of controls, conflict with key interactions, or low discovery makes a revealable menu a poor choice

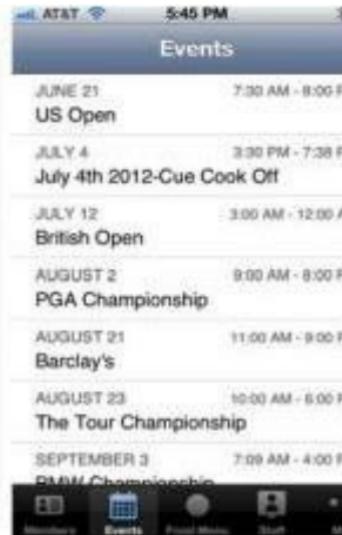
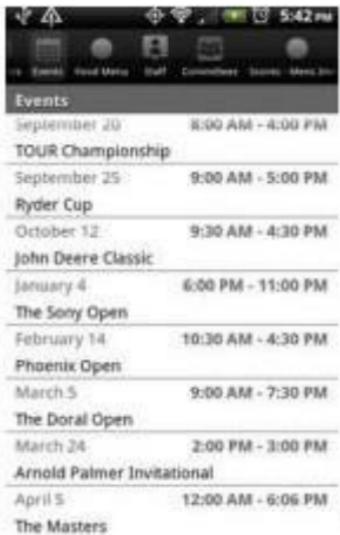


FIGURE 4-8: Fixed menu

iii) Expandable Menu

- ❖ When all function options for one page cannot fit on the viewport, an expanding menu can provide selectively available options.
- ❖ A gesture, like a swipe or double tap, may prompt the reveal as well as selecting an on-screen icon.
- ❖ Soft keys — hardware buttons connected to on-screen labels — may also prompt a menu to reveal.
- ❖ Users often benefit from multiple access options, especially for the harder-to-find gestural prompts.

iv) Scroll

- ❖ As in the case of a revealable menu giving additional functionality, there will often be more content on a screen than can be seen in the device viewport.
- ❖ It is best to limit scrolling, limiting application screens to the size of the viewport whenever possible.
- ❖ Some types of apps you develop will require scrolling to present data to the user effectively, such as an email client.
- ❖ When scrolling must occur, check that the design communicates the area that can be scrolled, the current context of a user, and the ratio of the current view to the total content available.

v) Content and Controls

- ❖ Input mechanisms, the controls users manipulate to enter or modify data, can be enhanced during the design process by keeping a few key concepts of mobile usability.
- ❖ Use standard controls, which positively shape user perceptions of an application.
- ❖ Reveal Content You can display content either as a full page or revealed in context.
- ❖ When you use a full-page layout, the user should not need to constantly bounce back and forth between pages: all necessary information to use the features on that page should be displayed on that very page. If content might help users decide how to proceed, a quickly revealed, in-context design (such as a pop-up box) may be the better design choice.

MULTIMODAL AND MULTICHANNEL UI

INTRODUCTION

- Humans interact with technology to do something better or faster. It's that simple.
- When we try to divine the future of mobile, we need to start with speed. Specifically, how the UI can drive the —critical path to information,|| assuming —information|| encompasses communication, entertainment, commerce, and anything else we may do on our devices.

Multichannel and multimodal mean for these two terms have been incorrectly used widely in the literature on mobile computing. To start, remember that multimodal and multichannel is not the same thing.

Multichannel may have two different meanings.

1) **First, it can be used to imply a user interface that establishes more than one communication channel to the user.** For example, a user interface may present an audio and a video channel to the user. It can also be used to refer to the number of different types of channels that a networked mobile application uses to reach other nodes on the network: If there is more than one channel—for example, a PSTN channel and a TCP/IP channel—the application is a multichannel application.

2) **Multimodality refers to the number of ways a user interface for an application may be presented. Hence, an application that only has a GUI but makes this GUI available to Palm devices as well as Windows-based desktops is a multimodal application.** Multimodality is a superset of the first definition of multichannel we discussed (where multichannel refers to the number of channels of communication between the computing apparatus and the user).

WHAT IS MULTIMODAL UI? (PART A,B)

DEFINITION

- Multimodal interfaces support user input and processing of two or more modalities—such as speech, pen, touch and multi-touch, gestures, gaze, and virtual keyboard.
 - These input modalities may coexist together on an interface, but be used either simultaneously or alternately. The input may involve recognition-based technologies (e.g., speech, gesture), simpler discrete input (e.g., keyboard, touch), or sensor-based information.
 - Some of these modalities may be capable of expressing semantically rich information and creating new content (e.g., speech, writing, keyboard), while others are limited to making discrete selections and controlling the system display (e.g., touching a URL to open it, pinching to shrink a visual display).
-
- Multimodal interaction provides the user with multiple modes of interacting with a system. A multimodal interface provides several distinct tools for input and output of data.
 - Multimodal UI enables faster and more free and natural communication. It can offer a flexible and efficient interface allowing users to interact through voice, text,

hand gesture and even gaze, and to receive information from the app through output modalities, such as speech synthesis, smart graphics and more.

For example, you're thinking about buying a new laptop. You might start by asking your mobile phone voice assistant —**What are the best laptops?** This would generate a list of laptops, which you can read.

- Spoken interaction
- Reading interaction
- Spoken and Reading Interactions
- Spoken
- Typing and Reading interaction.
- Touch ID

WHY WE NEED MULTIMODAL APPLICATIONS?

- To make the application more convenient to use given the condition of the mobile user and the dimensions of mobility. By default, all human communication behavior is multimodal

- One of our goals in building any type of user interface, not just mobile ones, is to get as close to the natural state of human-to-human communication for human-to-computer interactions.

What are the critical factors that affect the user experience of the mobile user?

1. **Context-Aware Computing:** This context adds meaning to when, where, and how the user interacts with the computing application and gives a different meaning to the individual interactions. Much of the context is defined by the domain: What is the problem that the application is trying to solve (commerce, navigation, game, etc.)? Users typically think of context-aware applications as -smarter (if they are well designed). What this translates to is a better usage experience by the user.

2. **Environmental Factors:** We know that our target users are mobile. We also understand that the dimensions of mobility distinguish the design and implementation of mobile applications from their stationary counterparts. Environmental conditions can further complicate the problem. For example, an application that is to be used in a rugged environment with low visibility may have additional requirements when it comes to the user interface.

MULTIMODALITY, MULTICHANNEL COMMUNICATION WITH THE NETWORK, AND NETWORK INFRASTRUCTURES

- **Most mobile devices are resource-starved.** This means that there is typically some bottleneck in network communication, CPU, permanent storage, memory, or some other capability of the device. Implementing multimodality related features, not unlike any other feature that we may wish to build into the mobile application, requires CPU, storage space, memory, and other resources.
- **Multimodality is not always desirable or possible based on the amount of resources on the device.**
- For example, audio and video channel
- With GSM and any cellular system prior to GSM, there is only one channel to the mobile device. General Packet Radio Service (GPRS) is the first type of network that enables multiple channels to exist simultaneously.
- WLANs (wireless LANs) such as WIFI and Bluetooth are becoming more and more pervasive, thereby offering high-bandwidth access to the network through IP-based technologies.

TYPES OF MULTIMODALITY

User interfaces establish channels of communication to the user. There is input and output channels depending on whether the channel can present information to the user, obtain information from the user, or do both.

Multimodality, then, is the use of two or more channels.

1. **Symmetric** → **Symmetric multimodality means that all input modes** (speech, gesture, facial expression, etc.) are also available for output, and vice versa
 2. **Asymmetric** → A touch-screen comprises a symmetric channel. Obviously, **at some point in the internal implementation of the interface channel, be it hardware or software, things are not symmetric:** The mechanism used to light up the pixels on the screen is different than the mechanism that collects the user's touch on the screen. Also, we could say that touch-screens are not symmetric because the tactile sense is also used
-
1. **Sequential multimodal input is the simplest type, where at each step of the interaction;** either one or the other input modality is active, but never more than one simultaneously.

2. Uncoordinated, simultaneous multimodal input allows concurrent activation of more than one modality. However, should the user provide input on more than one modality, this information is not integrated but will be processed in isolation, in random order.
3. Coordinated, simultaneous multimodal input fully exploits multimodality, providing for the integration of complementary input signals from different modalities into a joint event, based on time stamping.

Bernsen [Bernsen 2002] introduces taxonomy of interaction types for multimodal user interfaces based on the characteristics of interactions

1. Atomic
2. Composite
3. Linguistic/ non linguistic
4. Arbitrary/ non arbitrary

USABILITY-CENTERED USAGE OF MULTIMODALITY

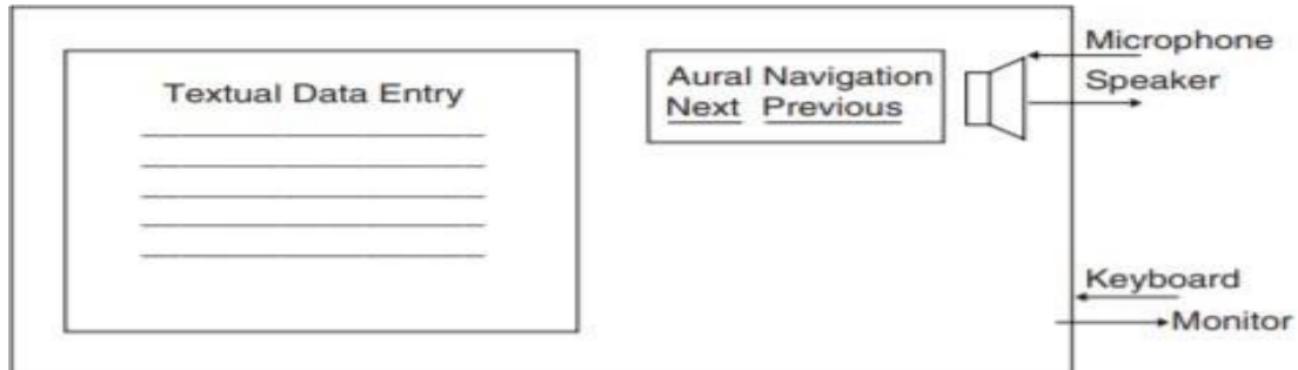
- The point of building multimodal user interfaces is to improve the usability of an application.

- Multimodal user interfaces are a relatively new technology. Understanding usability issues is something that comes about after years of maturity and deployment experience for a particular technology.
 1. Maximum width
 2. Portability
 3. Ubiquity (where ever)
 4. Saliency(Importance)
 5. Simultaneity
 6. Privacy

MODELING MULTICHANNEL AND MULTIMODAL APPLICATIONS WITH UML

Mandel, Koch, and Maier use abstract data views to basically represent the twodimensional spatial layout of a GUI. This is essentially a functional mockup of the user interface for GUIs. **Next, they introduce configuration diagrams; these diagrams can be quite useful in representing properties of multimodal user interfaces.** They have the following characteristics:

1. They show the composite structure and composite behavior of the user interface as well as showing the inputs and outputs into a user interface.
2. They also show, at a very high granularity level, the relationship between user interface components and non-user-interface components in the system.
3. They are describable, with UML metamodeling, as a real UML extension.



Configuration diagram for basic two mode user interface

Next, Mandel and colleagues introduce an improved version of ADVcharts to provide a visual schema for the specification of the dynamic aspects of the user interface.

Works by Hausmann, Sauer, Heckel, and Engles use UML and its extensions to model multimedia content (which is typically synonymous to multimodal content). These extensions are called the Object-Oriented Modeling of Multimedia Applications (OMMMA)

Abstraction Name	Description	Representation
User Input	<p>This abstraction is created to show the data items that the user gives the system. You can think of these as form fields. Though, per Mandel et al.'s specifications, these are data items that are requested from the user and there is nothing that keeps us from using this abstraction for natural-language or mixed-initiative input modeling.</p>	<div style="border: 1px solid black; padding: 5px;"> <p>InputName</p> <p>FieldName1</p> <p>FieldName2</p> </div>
Anchors	<p>Anchors are the starting points of the navigation [Mandel, Koch, and Maier 1999]. You can think of them as the initial state of a UML state diagram or a UML sequence diagram. This is largely a concept from the design and implementation of HTTP and HTML. Anchors are a navigational tool in HTML.</p>	<div style="border: 1px solid black; padding: 5px;"> <p>AnchorName</p> </div>
Collections	<p>This abstraction is basically created to indicate that some composition of the other abstractions is itself a valid abstraction; for example, a viewable table may be recognized as a collection.</p>	<div style="border: 1px solid black; padding: 5px;"> <p>*Object 1</p> <p>*Object 2</p> </div>
Sound	<p>This abstraction provides a method to</p>	<div style="border: 1px solid black; padding: 5px;">  </div>

TABLE 8.2. Basic OMMMA Stereotypes

Stereotype	Description
Presentation	This stereotype represents the encapsulation of the job of those objects that are responsible for rendering the content, through the relevant channels, to the user. These objects are responsible for determining things like the position of a certain icon on the screen, when a sound should be heard, and other temporal and spatial properties of the user interface.
Media	This stereotype represents an encapsulation of the content. For example, we could have an <code>Audio</code> class (which could be further specialized) or a <code>VisualText</code> class. We can model the commonalities and differences among various media content types using inheritance, aggregation, and more complex design patterns.
Application	This stereotype represents the abstraction of the link between the business logic components and the rest of the system. Application objects can be seen roughly as the equivalent of the model portion of MVC (discussed in earlier chapters).

MULTIMODAL CONTENT

One of the current points of contention in standards for multimodal user interfaces surrounds the creation and storage of multimodal content. SMIL gives us a standard for synchronizing different types of single-mode and single-channel content; it is a fairly well established standard.

1) X + V

- **X + V stands for XHTML and VXML. X + V is an interim technology:** Its use will be limited and only relevant to the duration of migration of content from XHTML/HTML and VXML content to truly multimodal content represented in a markup language designed with the needs of multimodality.
- The benefits of X + V are that both technologies are rather proved and that implementation of a multimodal interface can be done with relative speed.
- The problems with X + V are multifold. First, it assumes that we can tie together VXML, a language designed for representing VUIs, and XHTML, the younger but more mature sibling of HTML designed for visual user interfaces, and come up with a multimodal interface

2) M3L

- The M3L, yet another acronym for the Multimodal Markup Language, has been created for SmartKom.

- **SmartKom is a multimodal dialogue system that combines speech, gesture, and facial expressions for input and output**; it provides anthropomorphic and effective user interface through its personification of an interface agent.

3)MML

- MML there are now many different markup languages presented by different entities called Multimodal Markup Language.

TABLE @.3. MML Tags

Tag Name	Description
modalityout	This tag shows the modality of the element and its treatment For output presentation by various channels.
modalityIn	This tag encapsulates the information needed For processing of various types of input coming in from the different
producciozx	This tag is specific to speech synthesis (TTS). It encapsulates the information needed to produce the appropriate quality of speech (see Chapter 7) such as pitch, rate, etc.
Timing	This tag allows us to specify the temporal sequence of how various information such as text and audio are rendered (output).
BargeIn	This tag lets us specify if the user should be able to interrupt the presentation of multimodal information. (On a visual screen, this could be pressing the <i>stop</i> button on a browser whereas it is a simple interruption of audio played by the system through audio input in a VIII.I
tniriartve	This tag lets us specify if our dialogue is in mixed-initiative or directed mode.

4)EMMA

- Extensible Multimodal Annotation Markup Language (EMMA) is a standard being worked on by W3C.
- The conceptual design of EMMA takes after some of the other W3C user interface standards such as VXML, XHTML, and XForms.
- First, EMMA is naturally XML based. Next, EMMA defines metadata, which encapsulates information about the document, a data model, which defines what type of data can be instantiated, by the browser of the document, and instance data, which encapsulates things like the actual data, text, and pointers to files to be presented.

5)MPML

- Multimodal Presentation Markup Language (MPML), designed by a team at Ishizuka Laboratories in Japan, is a well-defined XML-based language that allows for the specification of dialogue-based interactions for a multimodal user interface

6) MMIL

- The Multimodal Interface Language (MMIL), proposed by Kumar and Romary, focuses on specifying an interface among the various

components of a multimodal architecture, specifically one that leverages multimodal integration and dialoguebased user interactions

7) InkML

- **InkML (Ink Markup Language) is an effort by the W3C multimodal interaction** working group to create a markup language for supporting text entry through a **digital pad and a styluslike device.**
- The idea of InkML is to allow us to create graphical and text-based user interfaces that can be used just like pen and paper.
- Although InkML is not specifically designed for multimodality, it is a significant markup language for developing multimodal user interfaces.

8) CUIML

- **CUIML, the Cooperative User Interfaces Markup Language, is developed by Sandor and Reicher as an extension to UIML. CUIML's syntax is based on UIML.**

Platforms

1.java

2.BREW

3. Microsoft

4. Symbian

5. WAP

TYPES OF UI

Fusion based

- The most **advanced multimodal interfaces are fusion-based ones that co- process two or more combined user input modes that can be entered simultaneously**, such as speech and pen input, speech and lip movements, or speech and manual gesturing.
- They process meaning from **two recognition-based technologies to fuse an overall interpretation based on joint meaning fragments from the two signal streams.**
- At the signal level, recognition technologies process continuous input streams (e.g., acoustic speech, digital ink trace) that contain high-bandwidth and often semantically rich information, which can be used for content creation—or to create and interact with a system's application content. Fusion-based multimodal interfaces have been **developed and commercialized in recent years.**

Alternative mode

- The simplest multimodal interfaces have input options that are used as **alternative modes, rather than being co-processed**. These rudimentary multimodal interfaces have been widely commercialized, for example on cell phones.
- They often offer more input options than a fusion-based multimodal system, such as speech, touch and multi-touch, gestures (e.g., pinching, flicking), stylus, keyboard, and sensor-based controls.

1) **Multimodal Interfaces for Content Creation** incorporate high-bandwidth or semantically rich modes, which can be used to create, modify, and interact with system application content (e.g., drawing lake on a map). They typically involve a recognition-based technology.

2) **Multimodal Interfaces for Controlling the System Display** incorporate input modes limited to controlling the system or its display, such as zooming or turning the system on, rather than creating or interacting with application content. Examples include touch and multi-touch (e.g., for selection), gestures (e.g., flicking to paginate), and sensor-based controls (e.g., tilt for angle of screen display).

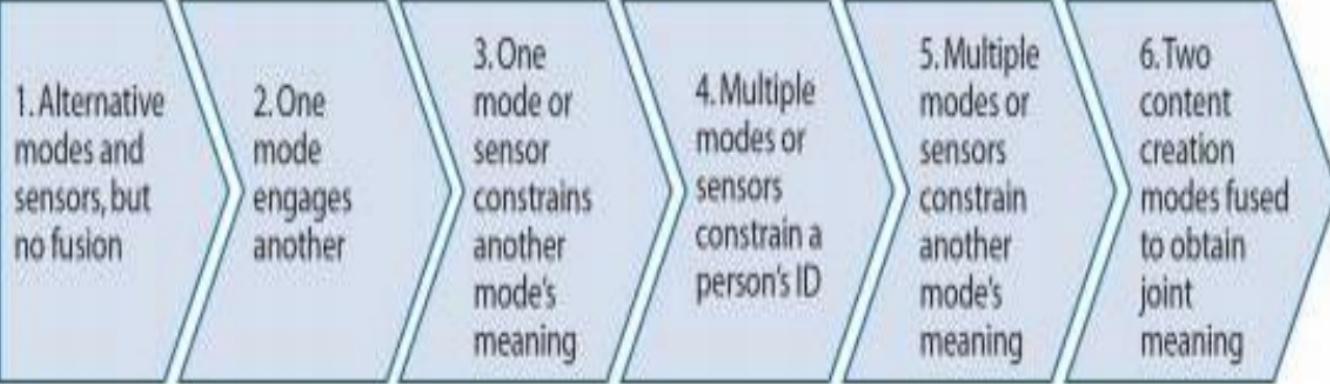
Multimodal and multi sensor

- On mobile devices, some **multimodal interfaces actually are multimodal- multisensor ones**.

- These interfaces combine multimodal user input with one or more forms of sensor control involving contextual cues.
- For example, proximity of a cell phone to the user's ear may turn it on, or the angle of holding a phone may orient its display.

Temporally cascaded

- Temporally cascaded multimodal interfaces process a combination of modalities that tend to be sequenced in a particular order, such as gaze directed at an object followed by speaking about it.
- On current smart phones has included both active and passive recognition-based input modes. It also has involved rapid incorporation of new sensors, increasing the prevalence of multimodal-multisensor interfaces.



GESTURE BASED USER INTERFACE

**BRIEFLY EXPLAIN ABOUT GETURE BASED UI AND ITS TECHNOLOGIES (PART B, C)
WRITE DOWN THE APPLICATION OF GESTURE UI(PART B,C)**

- **A gesture based user interface is an innovation that uses gestures as an input.**

- These new computers include well touch screen software which identifies gesture based human computer interaction.
- Gestural UI refers to using specific gestures, like scrolling, pinching, and tapping to operate an interface.
- It also refers to gesture recognition, including tipping, tilting, eye motion, and shaking.
Gestural user interface and gesture recognition technology has evolved from very basic motions and applications to the complex, and it is now part of everyday life for a huge number of people.

Use of a Gesture based user interface in training simulation-

- A gesture based user interface is not only useful for video game and entertainment purpose
furthermore useful for training simulation also
- Training simulation technology is a popular training choice for corporate companies and in a medical profession. With the help of Training simulation, students will get a benefit of gesture based human computer interaction and manage the system through their gestures. Trainees and students get an experience of the risky situation within the controlled environment.

3 benefits of the Gesture based user interface-

1. **Electronic industry-** It is an industry where everyone is coming with the highest focus on a user interface. Like Samsung galaxy, s4 immediately scrolls down when your eyes reach the bottom of the page.
2. **Medical industry benefit-** One of the most benefits of gesture based human computer interaction is surgeon can review patient's case file.
3. **Great for marketing strategy-** Gesture based human computer interaction and virtual reality mixture is a great strategy from marketing view point. Customer will feel real and accurate experience which stands out your product from others.

Definition

What is it?

- Gesture-based computing refers to interfaces where the human body interacts with digital resources without using common input devices, such as a keyboard, mouse, game controller, or voice-entry mechanism.
- Such systems can scan a limited space, analyze human movement within it, and interpret the motions of hands and arms, head, face, or body as input.

- Touchscreen interfaces likewise enable gestures such as swipes or taps to navigate, play games, or organize data.
- **Gesture recognition technology is used in a wide array of applications, from those that parse facial expressions to determine human emotional reactions to complex, augmented-reality simulations that evaluate whole- body movement.**

This technology is playing a very important role in many fields such as

- **Smartphone, Tablets and other devices**
- **Automobiles**
- **Transit sector**
- **Electronics sector**
- **Gaming sector**

GESTURE TYPES

- ◎ **Offline gestures:** Those gestures that are processed after the user interaction with the object. An example is the gesture to activate a menu.
- ◎ **Online gestures:** Direct manipulation gestures. They are used to scale or rotate a tangible object.

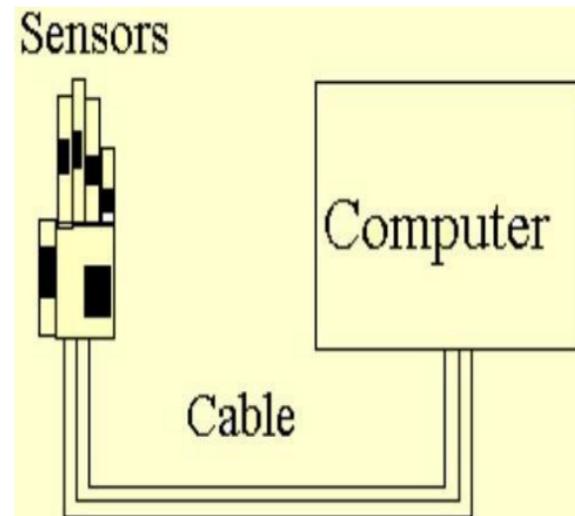
HOW DOES IT WORK?

- Motion may be used to **input navigation commands** or it may serve as data. When it is employed for navigation without a touchpad, users might zoom by stepping forward and backward. Or they could control a video stream: pointing left to replay, presenting the flat of a hand to stop, or pointing right to continue.
- The approach is **widely used in interacting with games and to control entertainment systems, machinery, or robotic devices and to interact with smartphones and tablets.**
- Systems such as **Microsoft Kinect and Leap Motion** use camera/scanner technologies to capture body movements, which are interpreted by software.
- These gesture-recognition systems can capture multiple points of interaction simultaneously, making them valuable where captured motion might be analyzed for mastery, as in dance, athletic technique, or physical therapy.
 1. Replace mouse and keyboard
 2. Pointing gestures
 3. Navigate in a virtual environment
 4. Interact with a 3D world
 5. No physical contact with computer
 6. Communicate at a distance

WHY IS IT SIGNIFICANT?

- The introduction of the mouse allowed computer inputs to move beyond the linear dimension of the keyboard to two dimensions.
- Gesture-based computing is the next step in that evolution, **enabling 3D input that involves users in the computing activity.** The result can be less like talking about a molecule or DNA chain and more like lifting one to examine it from every angle.
- Combined with simulation, gesture-based interfaces allow learners to see results of actions taken, as when they gesture to control the movement of robotic devices, converse with a system using sign language, or hone their skills in a speech-therapy session.
- Tabletop systems that employ surface gestures may give small-group discussion a boost. Unlike a keyboard or mouse, where a single individual controls input, a **gesture-based system enables input from multiple participants.**

TECHNOLOGIES



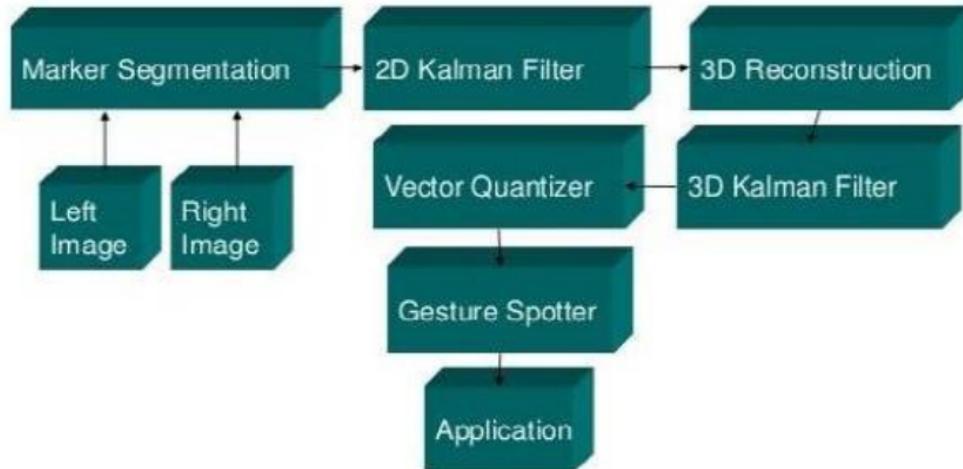
Two common technologies for hand getrure recognition are:

1. Glove based method

- Using special glove based device to extract hand posture
- Annoying

2. Vision based method

- 3D hand/arm modeling
- Apperance modeling



SCREEN ELEMENTS AND LAYOUTS

WRITE DOWN THE SCREEN ELEMENTS AND LAYOUT(PART B)

WHAT ARE THE TYPES OF LAYOUT AVAILABLE? EXPLAIN IN DETAIL.(PART B, C)

Layout is defined as the particular way elements are arranged. In the case of interface design, these elements are the informational, functional, framing, and decorative parts of the screen.

THE BASICS OF LAYOUT

VISUAL HIERARCHY

Visual hierarchy plays a part in all forms of design. The most important content should stand out the most, and the least important should stand out the least. A viewer should be able to deduce the informational structure from its layout.

A good visual hierarchy gives instant clues about the following:

- The relative importance of screen elements
- The relationships among them
- What to do next

WHAT MAKES THINGS LOOK IMPORTANT?

1. **Size** → The size of the headlines and subheads give the viewer a clue about order and importance.
2. **Position** → Simply by glancing at the size, position, and colors of the layout
3. **Density** → Density refers to the amount of space between elements of the screen.
4. **Background color** → Adding shade or a background color will draw attention to a block of text and distinguish it from other text.
5. **Rhythm** → Lists, grids, whitespace, and alternating elements such as headlines and summaries can create a strong visual rhythm that irresistibly draws the eye
6. **Emphasizing small items** → To make small items stand out, put them at the top, along the left side, or in the upper-right corner. Give them high contrast and visual weight, and set them off with whitespace.
7. **Alignment and grid** → In digital design, legibility is critical. Creating a design that is based on a grid allows the designer to focus on the content

8. **Four Important Gestalt Principles**→-Gestalt is a term that comes from a psychological theory that took hold in the 1920s. **Gestalt is a German word that means —form|| or —shape.||**

1. Promixity
2. Similarity
3. Continuity
4. Closure
5. Visual flow

USER INTERFACE ELEMENTS

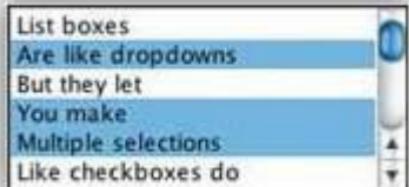
- **When designing your interface, try to be consistent and predictable in your choice of interface elements.**
 - Interface elements include but are not limited to:
1. **Input Controls:** checkboxes, radio buttons, dropdown lists, list boxes, buttons, toggles, text fields, date field

2. **Navigational Components:** breadcrumb, slider, search field, pagination, slider, tags, icons
3. **Informational Components:** tooltips, icons, progress bar, notifications, message boxes, modal windows
4. **Containers:** accordion

INPUT CONTROLS

Element	Description	Examples
Checkboxes	Checkboxes allow the user to select one or more options from a set.	<input type="checkbox"/> NonFederal (99) <input type="checkbox"/> Federal (57)

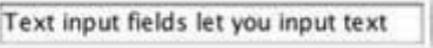
Element	Description	Examples
Radio buttons	<p>Radio buttons are used to allow users to select one item at a time.</p>	
Dropdown lists	<p>Dropdown lists allow users to select one item at a time, similarly to radio buttons, but are more compact allowing you to save space.</p>	

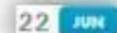
Element	Description	Examples
List boxes	<p>List boxes, like checkboxes, allow users to select a multiple items at a time, but are more compact and can support a longer list of options if needed.</p>	 <p>List boxes Are like dropdowns But they let You make Multiple selections Like checkboxes do</p>

Element	Description	Examples
Buttons	<p>A button indicates an action upon touch and is typically labeled using text, an icon, or both.</p>	
Dropdown Button	<p>The dropdown button consists of a button that when clicked displays a drop-down list of mutually exclusive items.</p>	

Element**Description****Examples**

Element	Description	Examples
Toggles	<p>A toggle button allows the user to change a setting between two states. They are most effective when the on/off states are visually distinct.</p>	
Text fields	<p>Text fields allow users to enter text. It can allow either a single line or multiple</p>	

Element	Description	Examples
	lines of text.	 Text input fields let you input text
Date and time pickers	A date picker allows users to select a date and/or time. By using the picker, the information is consistently formatted and input into the system.	

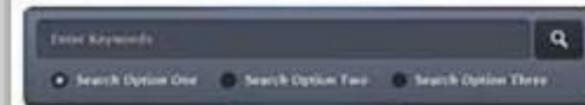
Element**Description****Examples**

NAVIGATIONAL COMPONENTS

Element	Description	Examples
Search Field	<p>A search box allows users to enter a keyword or phrase (query) and submit it to search the index with the intention of getting back the most relevant results. Typically search fields are single-line text</p>	

Element**Description****Examples**

boxes and are often accompanied by a search button.



Element	Description	Examples
Breadcrumb	<p>Breadcrumbs allow users to identify their current location within the system by providing a clickable trail of proceeding pages to navigate by.</p>	
Pagination	<p>Pagination divides content up between pages, and</p>	

Element	Description	Examples
	<p>allows users to skip between pages or go in order through the content.</p>	

Element	Description	Examples
Tags	Tags allow users to find content in the same category. Some tagging systems also allow users to apply their own tags to content by entering them into the system.	

Tags

Costs (72)

Health Conditions (54)

Improving Care (53)

Prevention (50)

Rights, Protections and
Benefits (135)

Insurance Coverage (141)

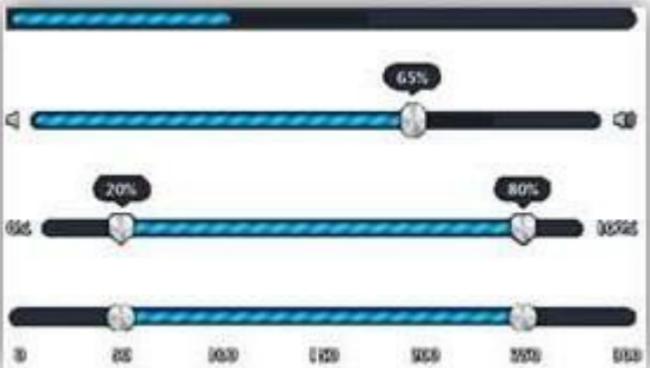
Clean

Fresh

Modern

Unique 



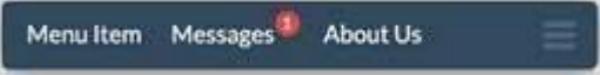
Element	Description	Examples
Sliders	<p>A slider, also known as a track bar, allows users to set or adjust a value. When the user changes the value, it does not change the format of the interface or other info on the screen.</p>	 <p>The image displays four horizontal sliders, each with a blue track and a black slider bar. The first slider has a scale from 0 to 100, with a value of 65%. The second slider has a scale from 0 to 100, with a value of 40%. The third slider has a scale from 0 to 100, with a value of 20%. The fourth slider has a scale from 0 to 100, with a value of 80%. Each slider has a small black circle with a percentage value above it.</p>

Element	Description	Examples
Icons	<p>An icon is a simplified image serving as an intuitive symbol that is used to help users to navigate the system.</p> <p>Typically, icons are hyperlinked.</p>	
Image	Image carousels allow	

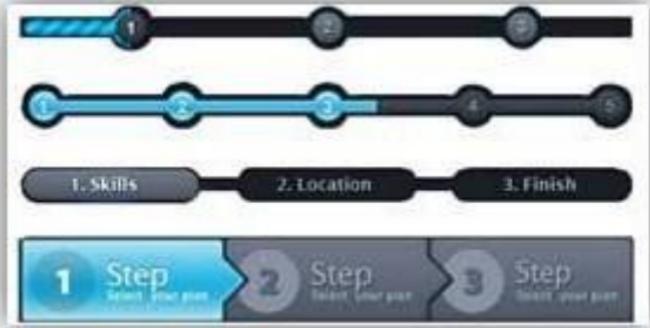
Element	Description	Examples
Carousel	<p>users to browse through a set of items and make a selection of one if they so choose. Typically, the images are hyperlinked.</p>	

INFORMATION COMPONENTS

Element	Description	Examples

Element	Description	Examples
Notifications	<p>A notification is an update message that announces something new for the user to see. Notifications are typically used to indicate items such as, the successful completion of a task, or an error or warning</p>	

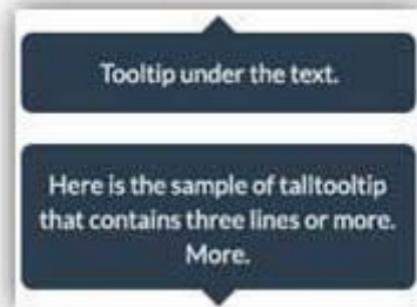
Element	Description	Examples
	message.	

Element	Description	Examples
Progress Bars	<p>A progress bar indicates where a user is as they advance through a series of steps in a process. Typically, progress bars are not clickable.</p>	 <p>The image displays three distinct progress bar designs. The top two are horizontal bars with five numbered circles (1-5) indicating progress. The first bar has circles 1, 2, and 3 filled with light blue, while 4 and 5 are dark grey. The second bar has circles 1, 2, and 3 filled with light blue, while 4 and 5 are dark grey. Below these are two dark grey horizontal bars with rounded ends. The first bar is labeled '1. Skills', the second '2. Location', and the third '3. Finish'. The bottom example is a vertical bar divided into three segments, each containing a circle with a number (1, 2, 3) and the word 'Step' followed by a descriptive phrase: 'Select your plan', 'Select your plan', and 'Select your plan' respectively.</p>

Element	Description	Examples
---------	-------------	----------

Tool Tips

A tooltip allows a user to see hints when they hover over an item indicating the name or purpose of the item.



Element	Description	Examples
Message Boxes	<p>A message box is a small window that provides information to users and requires them to take an action before they can move forward.</p>	 <p>This is a box</p> <p>Placeholder text for the message box.</p> <p>READ MORE</p>

Element	Description	Examples
Modal Window (pop-up)	<p>A modal window requires users to interact with it in some way before they can return to the system.</p>	

CONTAINERS

Element	Description	Examples
Accordion	<p>An accordion is a vertically stacked list of items that utilizes show/ hide functionality. When a label is clicked, it expands the section showing the content within. There can have one or more items showing at a time and may have default states that reveal one or more sections</p>	

Element	Description	Examples
	without the user clicking	

LAYOUTS

WHAT IS MEANT BY LAYOUT(PART A)

EXPLAIN THE TYPES OF LAYOUT IN DETAIL(PART B, C)

- Android Layout is used to define the user interface that holds the UI controls or widgets that will appear on the screen of an android application or activity screen. Generally, every application is a combination of View and ViewGroup.

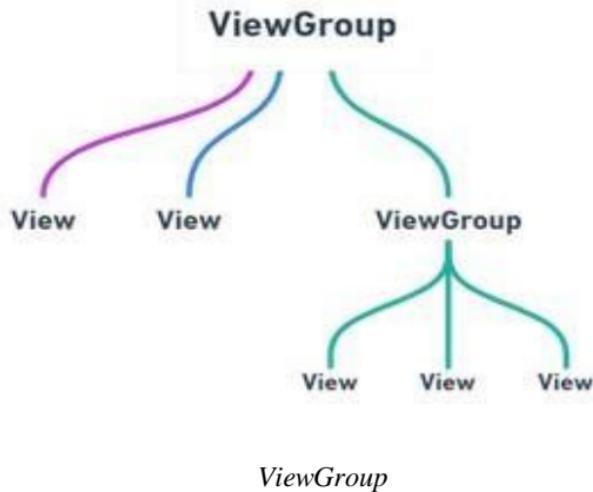
- An android application contains a large number of activities and we can say each activity is one page of the application.
- So, each activity contains multiple user interface components and those components are the instances of the View and ViewGroup. All the elements in a layout are built using a hierarchy of **View** and **ViewGroup** objects.

VIEW

A View is defined as the user interface which is used to create interactive UI components such as TextView, ImageView, EditText, RadioButton, etc., and is responsible for event handling and drawing. They are Generally Called Widgets.



A ViewGroup act as a base class for layouts and layouts parameters that hold other Views or ViewGroups and to define the layout properties. They are Generally Called layouts.



The Android framework will allow us to use UI elements or widgets in two ways:

- Use UI elements in the XML file
- Create elements in the Kotlin file dynamically

TYPES OF ANDROID LAYOUT

- **Android Linear Layout:** LinearLayout is a ViewGroup subclass, used to provide child View elements one by one either in a particular direction either horizontally or vertically based on the orientation property.
- **Android Relative Layout:** RelativeLayout is a ViewGroup subclass, used to specify the position of child View elements relative to each other like (A to the right of B) or relative to the parent (fix to the top of the parent).
- **Android Constraint Layout:** ConstraintLayout is a ViewGroup subclass, used to specify the position of layout constraints for every child View relative to other views present. A ConstraintLayout is similar to a RelativeLayout, but having more power.

- **Android Frame Layout:** FrameLayout is a ViewGroup subclass, used to specify the position of View elements it contains on the top of each other to display only a single View inside the FrameLayout.
- **Android Table Layout:** TableLayout is a ViewGroup subclass, used to display the child View elements in rows and columns.
- **Android Web View:** WebView is a browser that is used to display the web pages in our activity layout.
- **Android ListView:** ListView is a ViewGroup, used to display scrollable lists of items in a single column.
- **Android Grid View:** GridView is a ViewGroup that is used to display a scrollable list of items in a grid view of rows and columns.

VOICE XML

DEFINE THE USE OF VOICE XML. (PART A)

WHAT IS MEANT BY VOICE XML? AND HOW ITS USED? (PART A,B,C)

What Does Voice XML Mean?

Voice XML is an Extensible Markup Language (XML) standard for storing and processing digitized voice, input recognition and defining human and machine voice interaction. Voice XML uses voice as an input to a machine for desired processing, thereby facilitating voice application development. A voice-based application is managed by a voice browser.

There are two basic Voice XML file types:

- Static: Hard coded by the application developer
- Dynamic: Generated by the server in response to client requests.

Voice XML architecture is based on the following components:

- Document server: Like a server that accepts client requests and generates appropriate response post processing
- Voice XML interpreter subsystem: Processes request response output generated by the document server.
- Implementation platform: Generates actions in response to user input

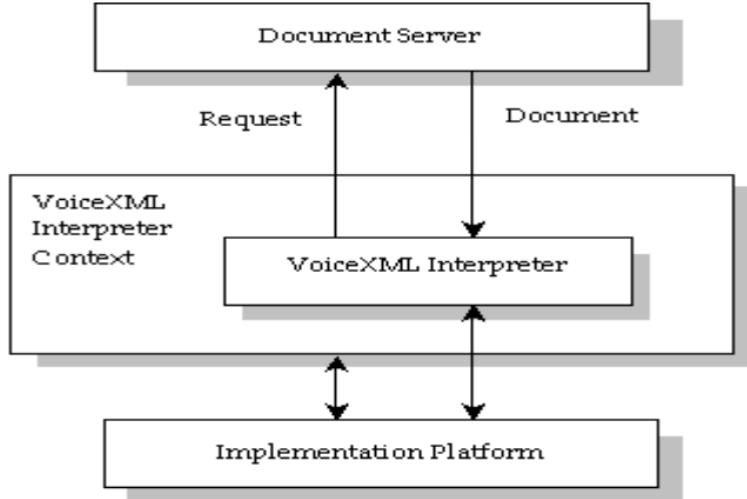
- Voice XML goals: **Integrates voice-based services with Web applications for efficiency**

Voice XML facilitates the following:

- Reduces client/server interactions by specifying multiple interactions per document
- Shields developers from low level implementation platform details
- Focuses on clear separation of business logic and interaction code
- Functions and delivers the same results, regardless of underlying implementation platform
- Creates and processes simple dialogs. Complex dialogs may be constructed and maintained with the help of Voice XML language tools.

ARCHITECTURAL MODEL

The architectural model assumed by this document has the following components:



- A *document server* (e.g. a Web server) processes *requests* from a client application, *the VoiceXML Interpreter*, through the *VoiceXML interpreter context*.
- The server produces *VoiceXML documents* in reply, which are processed by the VoiceXML interpreter. The VoiceXML interpreter context may monitor user inputs in parallel with the VoiceXML interpreter.

- For example, one VoiceXML interpreter context may always listen for a special escape phrase that takes the user to a high-level personal assistant, and another may listen for escape phrases that alter user preferences like volume or text-to-speech characteristics.
- The *implementation platform* is controlled by the VoiceXML interpreter context and by the VoiceXML interpreter.
- For instance, in an interactive voice response application, the VoiceXML interpreter context may be responsible for detecting an incoming call, acquiring the initial VoiceXML document, and answering the call, while the VoiceXML interpreter conducts the dialog after answer.
- The implementation platform generates events in response to user actions (e.g. spoken or character input received, disconnect) and system events (e.g. timer expiration).
- Some of these events are acted upon by the VoiceXML interpreter itself, as specified by the VoiceXML document, while others are acted upon by the VoiceXML interpreter context.

DESIGN GOALS

The developers of VoiceXML had several goals, many centered on how VoiceXML relates to Internet architecture and development. Here are some of the top areas where VoiceXML benefits from the Internet:

- **VoiceXML is an XML-based language.** This allows it to take advantage of the powerful Web development tools on the market. It also allows developers to use existing skill sets to build voice-based applications.
- **VoiceXML applications are easy to deploy.** Unlike many of the proprietary Interactive Voice Response (IVR) systems, VoiceXML servers can be placed anywhere on the Internet, taking advantage of common Internet server-side technologies.
- **The server logic and presentation logic can be cleanly separated.** This allows VoiceXML applications to take advantage of existing business logic and enterprise integration. Using a common back end allows the development of different forms of presentation logic based on the requesting device.
- **VoiceXML applications are platform-independent.** Developers do not have to worry about making VoiceXML applications work on multiple browsers over multiple networks. When developing VoiceXML applications, the only concern is making sure it works

with the VoiceXML browser being used. This leads to quicker development and less maintenance when compared to wireless Internet and desktop Web applications.

SCOPE OF VOICEXML

The language describes the human-machine interaction provided by voice response systems, which includes:

- Output of synthesized speech (text-to-speech).
- Output of audio files.
- Recognition of spoken input.
- Recognition of DTMF input.
- Recording of spoken input.
- Control of dialog flow.
- Telephony features such as call transfer and disconnect.

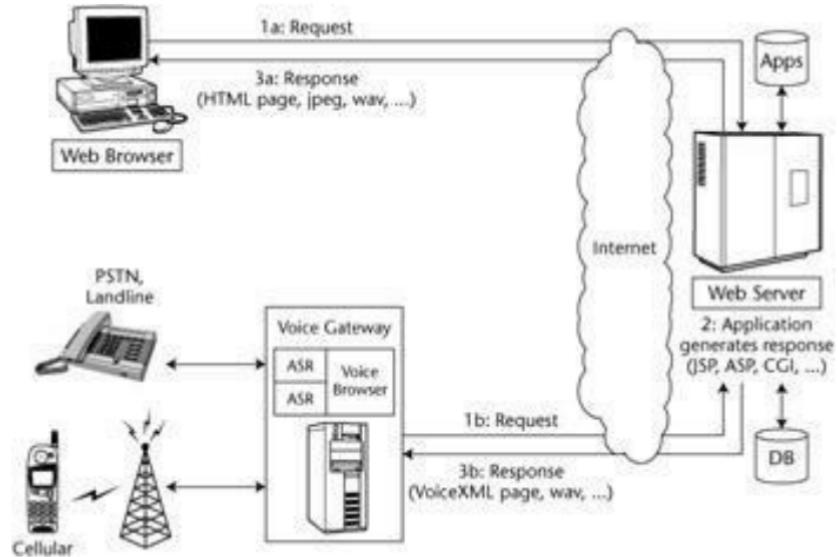
The language provides means for collecting character and/or spoken input, assigning the input results to document-defined request variables, and making decisions that affect the interpretation of documents

written in the language. A document may be linked to other documents through Universal Resource Identifiers (URIs).

VOICEXML ARCHITECTURE

VoiceXML uses an architecture similar to that of Internet applications. The main difference is the requirement for a VoiceXML gateway. Rather than having a Web browser on the mobile device, VoiceXML applications use a voice browser on the voice gateway. The voice browser interprets VoiceXML and then sends the output to the client on a telephone, eliminating the need for any software on the client device. Being based on Internet technology makes voice application development much more approachable than previous voice systems.

The VoiceXML application does have some additional complexity when compared to the Internet application. Instead of using the standard request/response mechanism used in Internet applications, the VoiceXML application goes through additional steps on the voice gateway.



VoiceXML architecture.

Just as Internet users enter a URL to access an application, VoiceXML users dial a telephone number. Once connected, the public switched telephone network (PSTN) or cellular network communicates with the voice gateway. The gateway then forwards the request over HTTP to a Web server that can service the request. On the server, standard server-side technologies such as JSP, ASP, or CGI can be used to generate the VoiceXML content, which is then returned to the voice gateway. On the gateway, a voice

browser interprets the VoiceXML code using a voice browser. The content is then spoken to the user over the telephone using prerecorded audio files or digitized speech. If user input is required at any point during the application cycle, it can be entered via either speech or tone input using Dual-Tone Multifrequency (DTMF). This entire process will occur many times during the use of a typical application.

As just stated, the main difference between Internet applications and VoiceXML applications is the use of a voice gateway. It is at this location where the voice browser resides, incorporating many important voice technologies, including Automatic Speech Recognition (ASR), telephony dialog control, DTMF, text-to-speech (TTS) and prerecorded audio playback. According to the VoiceXML 2.0 specification, a VoiceXML platform must support the following functions in order to be complete:

- **Document acquisition.** The voice gateway is responsible for acquiring VoiceXML documents for use within the voice browser. This can be accomplished within the context of another VoiceXML document or by external events, such as receiving a phone call. When issuing an HTTP request to a Web server, the gateway has to identify itself using the User-Agent variable in the HTTP header, providing both the browser name and the version number: "<name>/<version>".

- **Audio output.** Two forms of audio output must be supported: text-to-speech (TTS) and prerecorded audio files. Text-to-speech has to be generated on the fly, based on the VoiceXML content. The resulting digitized speech often sounds robotic, making it difficult to comprehend. This is where prerecorded audio files come into play. Application developers can prerecord the application output to make the voice sound more natural. The audio files can reside on a Web server and be referred to by a universal resource identifier (URI).
- **Audio input.** The voice gateway has to be able to recognize both character and spoken input. The most common form of character input is DTMF. This input type is best suited for entering passwords, such as PIN numbers, or responses to choice menus. Unfortunately, DTMF input is quite limited. It does not work well for entering data that is not numeric, leading to the requirement for speech recognition.
- **Transfer.** The voice gateway has to be capable of making a connection to a third party. This most commonly happens through a communications network, such as the telephone network.

When it comes to speech recognition, the Automatic Speech Recognition (ASR) engine has to be capable of recognizing input dynamically. A user will often speak commands into the telephone, which have to be recognized and acted upon. The set of suitable inputs is called a grammar. This set of data can either be directly incorporated to the VoiceXML document or referenced to an external location by

a URI. It is important to provide "tight" grammars so the speech recognition engine can provide accurate speech recognition in noisy environments, such as over a cell phone.

In addition to speech recognition, the gateway also has to be able to record audio input. This capability is useful for applications that require open dictation, such as notes associated with a completed work order.

REFERENCE

1. Reto Meier, Professional Android 4 Application Development;, Wiley, 1st edition,2012
2. ZigurdMednieks, Laird Dominm G. Blake Meike, Masumi Nakamura, -Programming Android, O'Reilly , 2nd edition,2012
3. Alasdair allan,iphone Programming, O'Reilly , 1st edition,2010.

*******UNIT II COMPLETED*******

UNIT III

MEMORY MANAGEMENT-DESIGN PATTERNS FOR LIMITED MEMORY – WORKFLOW FOR APPLICATION DEVELOPMENT–JAVA API–DYNAMIC LINKING–PLUGINS AND RULE OF THUMB FOR USING DLL–CONCURRENCY AND RESOURCE MANAGEMENT

OVERVIEW OF MEMORY MANAGEMENT

EXPLAIN ABOUT MEMORY MANAGEMENT IN MOBILE OS (PART B, C)

DESCRIBE MOBILE MEMORY MANAGEMENT (PART B, C)

DEFINITION

Memory management is the functionality of an operating system which handles or manages primary memory and moves processes back and forth between main memory and disk during execution. Memory management keeps track of each and every memory location, regardless of either it is allocated to some process or it is free.

When mobile operating system are considered, memory management plays the key role. As the mobile devices are constrained about hardware part, special care has to be taken for the memory management.

Android operating system is based on the linux kernel which mainly has the paging system. The memory is categorizes as internal memory, swap memory, external memory etc.,

- The Android Runtime (ART) and dalvik virtual machine use paging and memory-mapping (mmapping) to manage memory.
- This means that any memory an app modifies—**whether by allocating new objects or touching mmapped pages—remains resident in RAM and cannot be paged out.**
- The only way to release memory from an app is **to release object references that the app holds**, making the memory available to the garbage collector.
- That is with one exception: any files mmapped in without modification, such as code, can be paged out of RAM if the system wants to use that memory elsewhere.

LINUX KERNEL VS ANDROID OS

- Android OS is nearly similar to the Linux kernel. Android OS has enhanced its features by adding more custom libraries to the already existing ones in order to support better system functionalities. For example last seen first killed design, kill the least recently used process first.
- Memory management is the hardest part of mobile development. Mobile devices, from cheaper ones to the most expensive ones, have limited amount of dynamic memory compared to our personal computers.
- The basic facilities run by the kernel are process management, memory management, device management and system calls. Android also supports all these features.

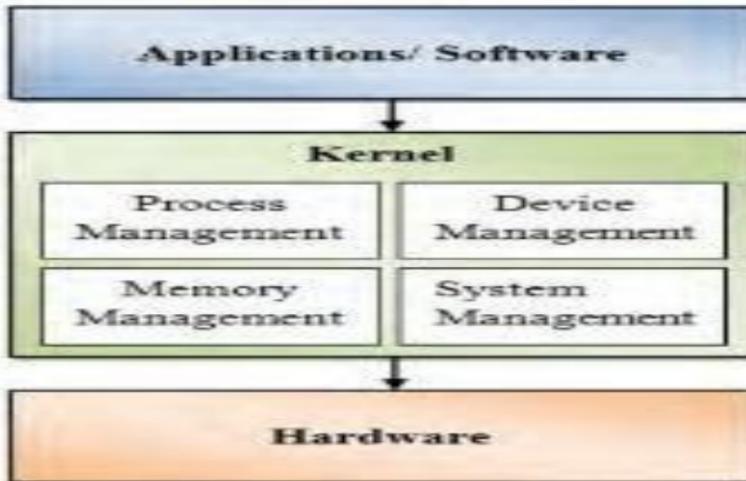


Fig 1: The Kernel Relation with Hardware and Applications

Poor memory management shows itself in various ways:

1. Allocating more memory space than application actually needs,
2. Not releasing the memory area retained by application,
3. Releasing a memory area more than once (usually as a result of multi-thread operations),
4. While using automatic memory solutions (ARC or GC), losing the tracks of your objects.

GARBAGE COLLECTION

- A managed memory environment, like the ART or Dalvik virtual machine, keeps track of each memory allocation.
- Once it determines that a piece of memory is no longer being used by the program, it frees it back to the heap, without any intervention from the programmer. **The mechanism for reclaiming unused memory within a managed memory environment is known as garbage collection.**

Garbage collection has two goals:

1. Find data objects in a program that cannot be accessed in the future;
2. Reclaim the resources used by those objects.

Android's memory heap is a generational one, meaning that there are different buckets of allocations that it tracks, based on the expected life and size of an object being allocated. For example, recently allocated objects belong in the *Young generation*. When an object stays active long enough, it can be promoted to an older generation, followed by a permanent generation.

Each heap generation has its **own dedicated upper limit on the amount of memory** that objects there can occupy. Any time a generation starts to fill up, the system executes a

garbage collection event in an attempt to free up memory. **The duration of the garbage collection depends on which generation of objects it's collecting and how many active objects are in each generation.**

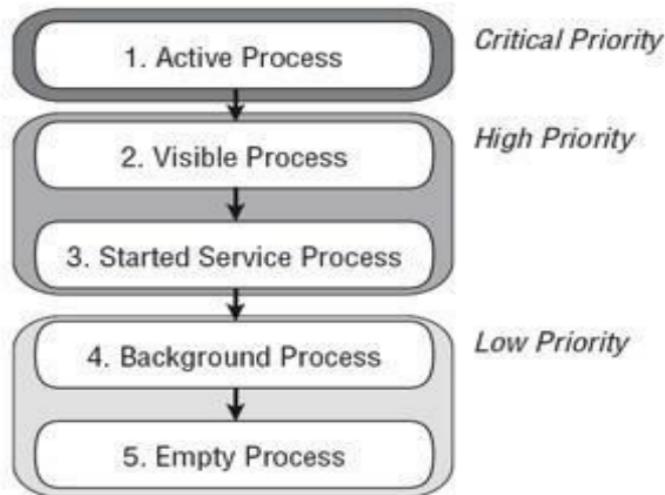
- **The system has a running set of criteria for determining when to perform garbage collection. When the criteria are satisfied, the system stops executing the process and begins garbage collection.**
- If garbage collection occurs in the middle of an intensive processing loop like an animation or during music playback, it can increase processing time. This increase can potentially push code execution in your app past the recommended 16ms threshold for efficient and smooth frame rendering.

UNDERSTANDING APPLICATION PRIORITY AND PROCESS STATES

- The order in **which processes are killed to reclaim resources is determined by the priority of the hosted applications.** An application's priority is equal to its highest-priority component.
- Where two applications have the same priority, the process that has been at a lower priority longest will be killed first. **Process priority is also affected by interprocess**

dependencies; if an application has a dependency on a Service or Content Provider supplied by a second application, the secondary application will have at least as high a priority as the application it supports.

- All Android applications will remain running and in memory until the system needs its resources for other applications.



Active Processes

- Active (foreground) processes are those **hosting applications with components currently interacting with the user.**
- These are the processes Android is trying to keep responsive by **reclaiming resources.** There are generally very few of these processes, and they will be killed only as a last resort.
- **Activities in an —active! state; that is, they are in the foreground** and responding to user events. You will explore Activity states in greater detail later in this chapter.
- Activities, Services, or Broadcast Receivers that are currently executing an onReceive event handler.
- Services that are executing an onStart, onCreate, or onDestroy event handler.

Visible Processes

- **Visible, but inactive processes are those hosting —visible! Activities.** As the name suggests, visible Activities are visible, but they **aren't in the foreground or responding to user events.**

Started Service Processes

- Processes hosting Services that have been started. **Services support ongoing processing that should continue without a visible interface.**
- Because Services don't interact directly with the user, they **receive a slightly lower priority than visible Activities.**

- They are still considered to be foreground processes and won't be killed unless resources are needed for active or visible processes.

Background Processes

- Processes hosting Activities that aren't visible and that don't have any Services that have been started are considered background processes.
- There will generally be a large number of background processes that Android will kill using a last-seen-first-killed pattern to obtain resources for foreground processes.

Empty Processes

- To improve overall system performance, Android often retains applications in memory after they have reached the end of their lifetimes.
- Android maintains this cache to improve the start-up time of applications when they're re-launched. These processes are routinely killed as required.

DDMS

- Android Studio includes a debugging tool called the **Dalvik Debug Monitor Service (DDMS)**. DDMS provides services like screen capture on the device, threading, heap information on the device, logcat, processes, incoming calls, SMS checking, location, data spoofing, and many other things related to testing your Android application.

- DDMS connects the IDE to the applications running on the device. On Android, every application runs in its own process, each of which hosts its own virtual machine (VM). And each process listens for a debugger on a different port.
- When it starts, DDMS connects to **ADB (Android Debug Bridge**, which is a command-line utility included with Google's Android SDK.).
- An Android Debugger is used for debugging the Android app and starts a device monitoring service between the two. This will notify DDMS when a device is connected or disconnected.
- When a device is connected, a VM monitoring service is created between ADB and DDMS, which will notify DDMS when a VM on the device is started or terminated.

DESIGN PATTERNS FOR LIMITED MEMORY

EXPLAIN THE DESIGN PATTERNS FOR LIMITED MEMORY DEVICES (PART B, C)

When composing designs for devices with a limited amount of memory, the most important principle is not to waste memory, as pointed out by Noble and Weir (2001). This means that the design should be based on the most adequate data structure, which offers the right operations.

LINEAR DATA STRUCTURES

- In contrast to data structures where a separate memory area is reserved for each item, linear data structures are those where different elements are located next to each other in the memory.
- Examples of non-linear data structures include common implementations of lists and tree-like data structures, whereas linear data structures can be lists and tables, for instance.
- The difference in the allocation in the memory also plays a part in the quality properties of data structures.

Linear data structures are generally better for memory management than non-linear ones for several reasons, as listed in the following:

- Less fragmentation. Linear data structures occupy memory place from one location, whereas non-linear ones can be located in different places. Obviously, the former results in less possibility for fragmentation.
- Less searching overhead. Reserving a linear block of memory for several items only takes one search for a suitable memory element in the run-time environment, whereas non-linear structures require one request for memory per allocated element. Combined with a design where one object allocates a number of child objects, this may also lead to a serious performance problem.

- **Design-time management.** Linear blocks **are easier to manage at design time**, as fewer reservations are made. This usually leads to cleaner designs.
- **Monitoring.** Addressing can be performed in a monitored fashion, because it is possible to check that the used index refers to a legal object.
- **Cache improvement.** When using linear data structures, it is more likely that the next data element is already in cache, as cache works internally with blocks of memory. A related issue is that most caches expect that data structures are used in increasing order of used memory locations. Therefore, it is beneficial to reflect this in designs where applicable.
- **Index uses less memory.** An absolute reference to an object usually consumes 32 bits, whereas by allocating objects to a vector of 256 objects, assuming that this is the upper limit of objects, an index of only 8 bits can be used. Furthermore, it is possible to check that there will be no invalid indexing.

BASIC DESIGN DECISIONS

1. Allocate all memory at the **beginning of a program**. This ensures that the application always has all the memory it needs, and memory allocation can only fail at the beginning of the program.
2. **Allocate memory for several items, even if you only need one.** Then, one can build a policy where a number of objects is reserved with one allocation request. These objects can then be used later when needed
3. Use standard allocation sizes

4. Reuse objects
5. Release early, allocate late
6. **Use permanent storage or ROM when applicable.** In many situations, it is not even desirable to keep all the data structures in the program memory due to physical restrictions.
7. **Avoid recursion.** Invoking methods obviously causes stack frames to be generated. While the size of an individual stack frame can be small – for instance, in Kilo Virtual Machine (KVM), which is a mobile Java virtual machine commonly used in early Java enabled mobile phones, the size of a single stack frame is at least 28 bytes (7×4 bytes) – functions calling themselves recursively can end up using a lot of stack, if the depth of the recursion is not considered beforehand.

Data Packing

Data packing is probably the most obvious way to reduce memory consumption. There are several sides to data packing.

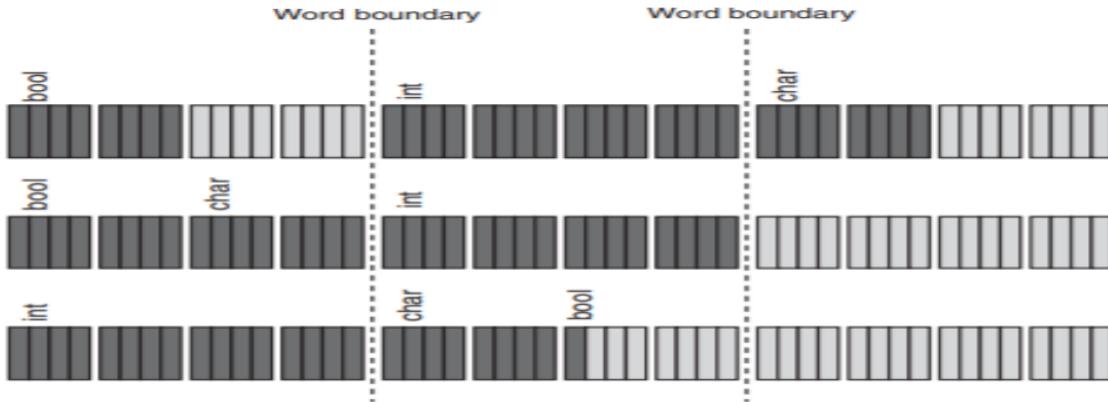


Figure 2.3 Saving memory by considering data structure alignment

Use compression with care. In addition to considering the data layout in memory, there are several compression techniques for decreasing the size of a file.

- Table compression, also referred to as nibble coding or Huffman coding, is about encoding each element of data in a variable number of bits so that the more common elements require fewer bits.

- **Difference coding is based on representing sequences of data according to the differences between them.** This typically results in improved memory reduction than table compression, but also sometimes leads to more complexity, as not only absolute values but also differences are to be managed.
- **Adaptive compression is based on algorithms that analyze the data to be compressed and then adapt their behavior accordingly.** Again, further complexity is introduced, as it is the compression algorithm that is evolving, not only data.

WORKFLOW FOR APPLICATION DEVELOPMENT

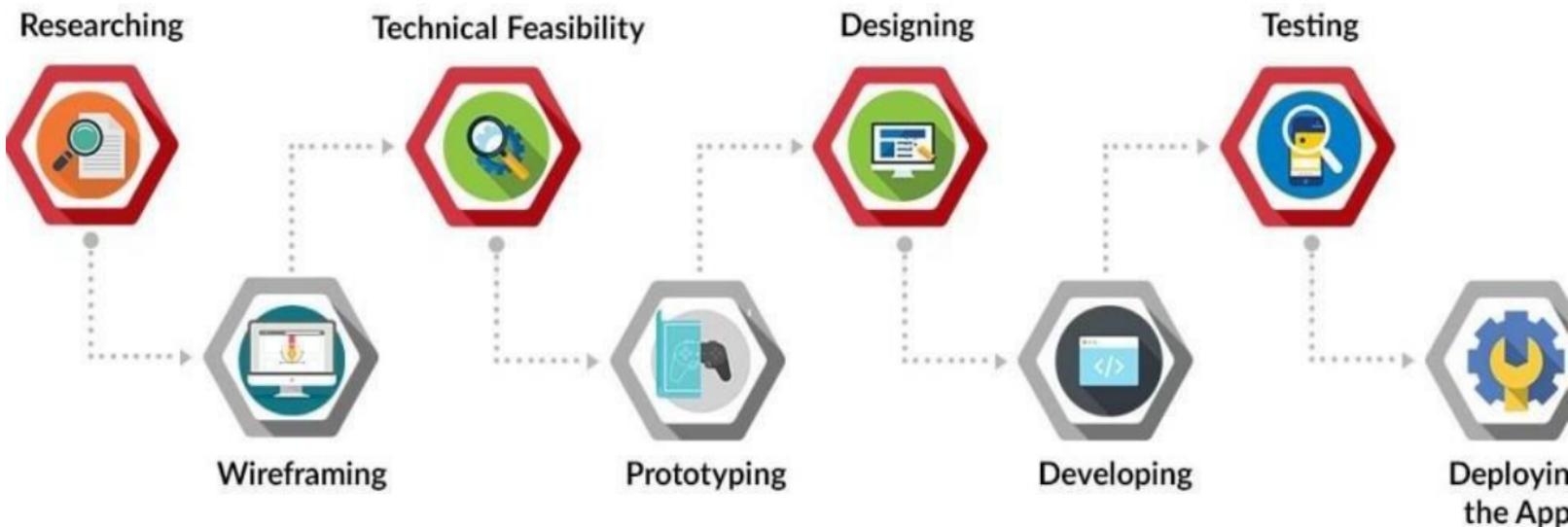
DESCRIBE THE WORKFLOW FOR APPLICATION DEVELOPMENT(PART B, C)

DEFINE WORKFLOW (PART A)

We have provided the complete phases of **a mobile app development process** that would guide you throughout your development journey.

1. **Researching**
2. **Wire framing**
3. **Evaluating Technical Feasibility**
4. **Prototyping**
5. **Designing**

6. Developing
7. Testing
8. Deploying the app



1. Researching

- You might already have plenty of ideas for your mobile app; it is still good to dig deeper into demographics, behavior patterns, and demand of your targeted audiences. The other important thing that covers up under this phase of the mobile app development process is not to overlook your competitors. By researching thoroughly, get yourself answers to these following questions:
 - **Who is your targeted audience?**
 - **What will be a suitable platform to launch your app?**
 - **What are your competitors offering to customers?**
- These are just a few questions from the long list that you have to keep in mind. While researching, think from your customers' perspective to find out what additional features should be there in your app to make it stand out in the crowd. Sparing enough time for researching and brainstorming will build a strong foundation for your mobile app development.

2. Wire framing

- **Wire framing gives a clear understanding of your app's features and functionalities, and hence, is a crucial phase.** Draw **detailed sketches of the product** you want to build to reveal the usability problems beforehand. Wire framing help narrow down the ideas and organize all the app design components correctly.
- Try to identify how your planned features will blend into a fully-functional mobile application. Also, make a storyboard or roadmap to demonstrate how a user will use and explore your app. Your prime focus should be on delivering an excellent customer experience by simplifying the roadmap.

3. Evaluating Technical Feasibility

- In this phase of mobile app development process, **you have to check whether the backend system would be capable of supporting the app's functionality or not.** To figure out the technical feasibility of your app's idea, access the public data by sourcing public application programming interfaces (APIs).
- We have to understand that an app with different formats (wearables, smartphones, tablets, etc.) and platforms (Android, iOS, or any other) will not have the same needs. By the end of this stage, you would have various ideas for your app's functionality.

4. Prototyping

- **Prototyping helps to determine whether you are moving in the right direction or not.** It is totally understandable that you cannot fully deliver the experience to let your users' knowledge about the working and functioning of your app without developing it completely.
- Create a prototype that conveys the app's concept to your targeted audience to verify how it works. We can allow your stakeholders to have the first look at your app and touch the prototype to deliver their feedback about it.

5. Designing

- **UI (User Interface) and UX (User Experience) are the two vital components of your mobile app design.** The former is responsible for the look and appeal of your application, whereas the latter facilitates the interaction between the design elements.
- The time required for designing cannot be specified as it may take only a couple of hours to a few days. Another factor that impacts your app designing time is the experience of the developers from your mobile app development services provider.

- It is a multistep process that needs to be done carefully to ensure that the outcome provides a clear vision of your business idea.

6. Developing

- Generally, this phase of the android and iOS app development process starts at the very initial stage.
- Right after you finalize an app idea, the developers **need to develop a prototype** to authenticate the features and functionalities.
- **The development phase is further divided into various sections**, where the team or a developer writes pieces of code, which then tested by another team. After marking the first section as bug-free, the development team moves further.
- In the case of complex projects with frequently changing user requirements, it is good to opt for an agile methodology. This type of methodology leads to progressive development and brings flexibility in the planning process.

7. Testing

- Testing early and frequently gives developers the **advantage of fixing a bug right** when it occurs.
- It also controls the **final cost of the development as** it will require both money and efforts to fix a bug, which occurred at the first stage, after you reach on fifth or more.
- **While testing your app, consider compatibility, security, usability, performance, UI checks, and other factors in mind.** Check whether your application serves its purpose or not.

8. Deploying the app

- **In this stage, your app is ready to launch.** To do so, select a day and release your mobile application on the desired platforms.
- Deploying the app is not the final step technically as you receive feedback from your audience and thus, have to make the changes accordingly.
- The other two mobile app development processes that remain associated with a mobile app are support and maintenance. **Maintenance is required to bring new functionalities and features to your app over time.** On the other hand, support should be there to

answer the queries of the users using your mobile application. It would be no wrong to say that mobile app development is a long-term commitment rather than just a short-term project.

JAVA API

BRIEFLY EXPLAIN ABOUT JAVA API. (PART B, C)

WHAT ARE THE ADVANTAGES OF JAVA API? (PART B,C)

What is Java?

- **Java is an object-oriented programming language that runs on almost all electronic devices. Java is platform-independent because of Java virtual machines (JVMs).**
- **It follows the principle of "write once, run everywhere."** When a JVM is installed on the host operating system, it automatically adapts to the environment and executes the program's functionalities.
- To install Java on a computer, the developer must download the JDK and set up the Java Runtime Environment (JRE).



As previously noted, a Java download consists of two files:

- JDK
- JRE

The JDK file is key to developing APIs in Java and consists of:

- The compiler
- The JVM
- The Java API

WHAT IS JAVA API AND THE NEED FOR JAVA APIs?

Java application programming interfaces (APIs) are predefined software tools that easily enable interactivity between multiple applications.

Compiler

A Java compiler is a predefined program that converts the high-level, user-written code language to low-level, computer-understandable, byte-code language during the compile time.

JVM

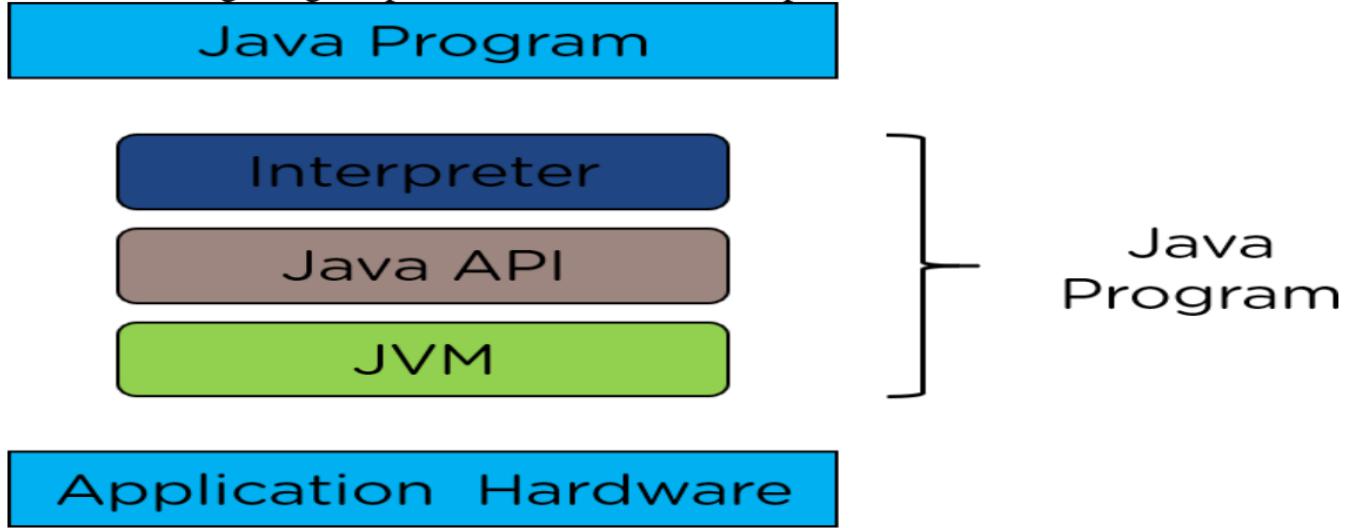
A JVM processes the byte-code from the compiler and provides an output in a user-readable format.

Java APIs

Java APIs are integrated pieces of software that come with JDKs. APIs in Java provides the interface between two different applications and establish communication. WHAT ARE JAVA APIs?

- APIs are important software components bundled with the JDK.
- APIs in Java include classes, interfaces, and user Interfaces.

- They enable developers to integrate various applications and websites and offer real-time information.
- The following image depicts the fundamental components of the Java API.



WHO USES JAVA APIs?

Three types of developers use Java APIs based on their job or project:

1. Internal developers

2. Partner developers
3. Open developers

Internal Developers

Internal developers use internal APIs for a specific organization. Internal APIs are accessible only by developers within one organization.

Applications that use internal APIs include:

- B2B
- B2C
- A2A
- B2E

Examples include Gmail, Google Cloud VM, and Instagram.

Partner Developers

Organizations that establish communications develop and use partner APIs. These types of APIs are available to partner developers via API keys.

Applications that use partner APIs include:

- B2B
- B2C

Examples include Finextra and Microsoft (MS Open API Initiative),

Open Developers

Some leading companies provide access to their APIs to developers in the **open-source format**. These businesses provide access to APIs via a key so that the company can ensure that the API is not used illegally.

The application type that uses internal APIs is:

- B2C

Examples include Twitter and Telnyx.

THE NEED FOR JAVA APIS

Java developers use APIs to:

Streamline Operating Procedures

Social media applications like Twitter, Facebook, LinkedIn, and Instagram

provide users with multiple options on one screen. Java APIs make this functionality possible.

Improve Business Techniques

Introducing APIs to the public leads many companies to release private data to generate new ideas, fix existing bugs, and receive new ways to improve operations.

Create Powerful Applications

Online banking has changed the industry forever, and APIs offer customers the ability to manage their finances digitally with complete simplicity.

TYPES OF JAVA APIs

There are four types of APIs in Java:

- Public
- Private
- Partner
- Composite



Public

Private

Partner

Composite

Public

Public (or open) APIs are Java APIs that come with the JDK. They do not have strict restrictions about how developers use them.

Private

Private (or internal) APIs are developed by a specific organization and are accessible to only employees who work for that organization.

Partner

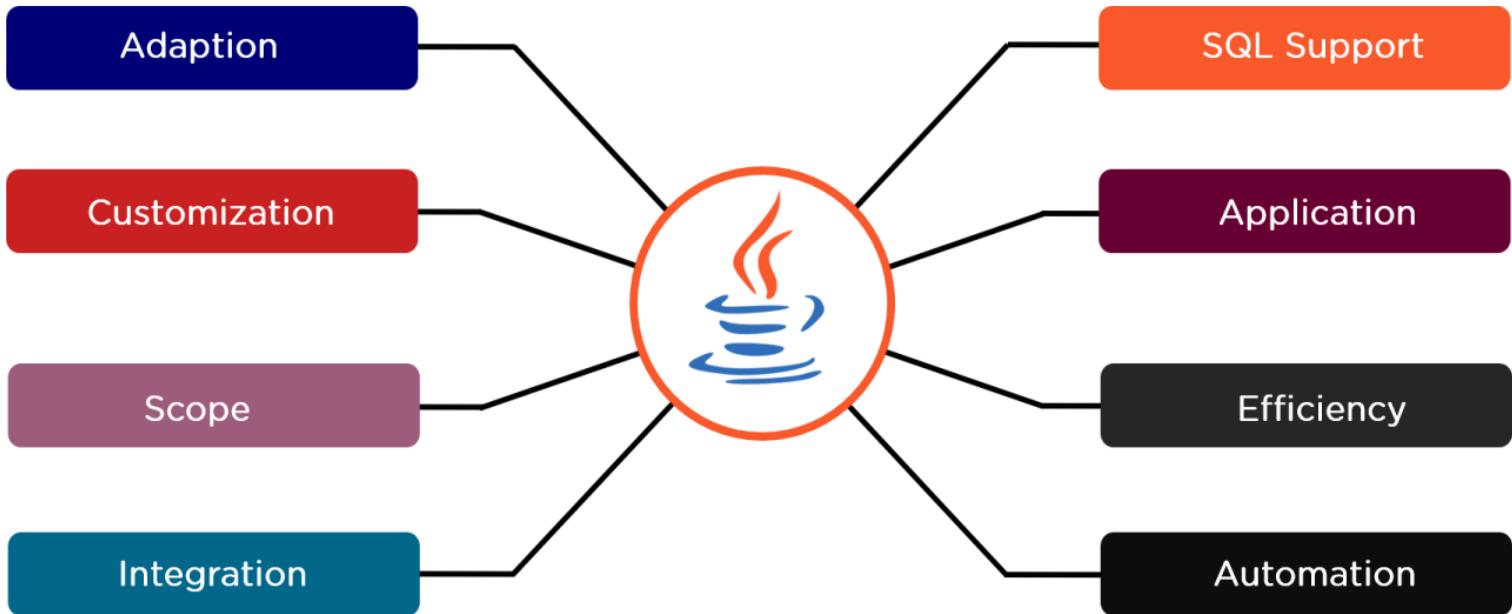
Partner APIs are considered to be third-party APIs and are developed by organizations for strategic business operations.

Composite

Composite APIs are micro services, and developers build them by combining several service APIs.

THE ADVANTAGES OF APIs

Some of the main advantages of using Java APIs include:



Extensive SQL Support

APIs in Java enable a wide range of SQL support services in user applications through a component-based interface.

Application

APIs in Java provide effortless access to all of an application's major software components and easily deliver services.

Efficiency

Java APIs are highly efficient because they enable rapid application deployment. Also, the data that the application generates is always available online.

Automation

APIs allow computers to automatically upload, download, update and delete data automatically without human interaction.

Integration

Java APIs can integrate into any application and website and provide a fluid user experience with dynamic data delivery.

Scope

Java APIs easily make websites, applications, and information available to a wide range of users and audiences.

Customization

Java APIs enable developers and businesses to build applications that personalize the user interface and data.

Adaptability

Java APIs are highly flexible and adaptable because they can easily accept feature updates and changes to frameworks and operating environments.

DYNAMIC LINKING

DEFINE DYNAMIC LINKING (PART A)

WRITE SHORT NOTES ABOUT DYNAMIC LINKING (PART B, C)

Dynamic linking, often implemented with dynamically linked libraries (DLL), is a common way to partition applications and subsystems into smaller portions, which can be compiled, tested, reused, managed, deployed, and installed separately.

- Several applications can use the library in such a fashion that only one copy of the library is needed, thus saving memory.
- Application-specific tailoring can be handled in a convenient fashion, provided that supporting facilities exist.
- Smaller compilations and deliveries are enabled
- Composition of systems becomes more flexible, because only a subset of all possible software can be included in a device when creating a device for a certain segment.
- It is easier to focus testing to some interface and features that can be accessed using that interface.
- Library structure eases scoping of system components and enables the creation of an explicit unit for management.

- Work allocation can be done in terms of dynamic libraries, if the implementation is carried out using a technique that does not support convenient mechanisms for modularity.

STATIC VERSUS DYNAMIC DLLS

- While dynamically linked libraries are all dynamic in their nature, there are two different implementation schemes.
- **One is static linking, which most commonly means that the library is instantiated at the starting time of a program**, and the loaded library resides in the memory as long as the program that loaded the library into its memory space is being executed.
- In contrast to static DLLs, **dynamic DLLs, which are often also referred to as plug-in, especially if they introduce some special extension, can be loaded and unloaded whenever needed**, and the facilities can thus be altered during an execution.
- The benefit of the approach is that one can introduce new features using such libraries. For instance, in the mobile setting one can consider that sending a message is an operation that is similar to different message types (e.g. SMS, MMS, email), but different implementations are needed for communicating with the network in the correct fashion.

CHALLENGES WITH USING DLLS

1. A common risk associated with dynamic libraries is the **fragmentation of the total system** into small entities that refer to each other seemingly uncontrollably
2. Another problem is that if a dynamic library is used by only one application, **memory consumption increases due to the infrastructure** needed for management of the library.

IMPLEMENTATION TECHNIQUES

- Fundamentally, **dynamically linked libraries can be considered as components that offer a well-defined interface for other pieces of software to access it.**
- Additional information may be provided to ease their use. This is not a necessity, however, but they can be **self-contained as well**, in which case the parts of the program that use libraries must find the corresponding information from libraries.
- Usually, this is implemented with standard facilities and an application programmer has few opportunities to optimize the system.

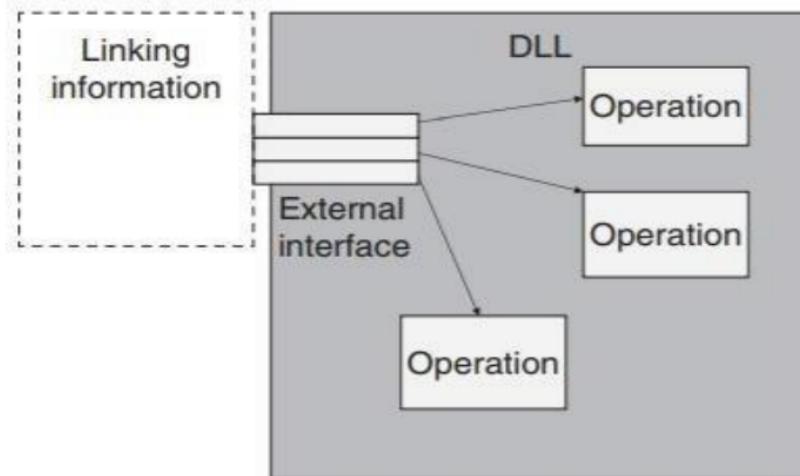


Figure 4.1 Dynamically linked library

Dynamically linked libraries can be implemented in two different fashions.

1. Offset based linking
2. Signature based linking

OFFSET BASED LINKING

- Linking based on offsets is probably **the most common way to load dynamic libraries.**
- The core of the approach is to add a table of function pointers to the library file, which identifies where the different methods or procedures exported from the dynamically linked library are located, thus resembling the virtual function table used in inheritance.

SIGNATURE BASED LINKING

- In contrast to offset-based linking of dynamically linked libraries, **also language-level constructs, such as class names and method signatures, can be used as the basis for linking.**
- Then, the linking is based on loading the whole library to the memory and then performing the linking against the actual signatures of the functions, which must then be present in one form or another.

PLUGINS AND RULE OF THUMB FOR USING DLL

WRITE THE USE OF PLUGINS AND ITS RULES IN DLL (PART B,C)

IMPLEMENTING PLUG INS

- Plugins, which dynamically loaded dynamically linked libraries are often referred to as, especially if they play a role of an extension or specialization, are a special type of DLL that enable differentiation of operations for different purposes at runtime.

- Usually implement a common interface used by an application, but their operations can still differ at the level of implementation.
- One could implement different plugins for different types of messages that can be sent from a mobile device.

PLUGIN PRINCIPLES

- Plugins take advantage of the **binary compatibility of the interface provided by a dynamically linked library.**
- The important concepts of a plugin are the interfaces they implement, and the implementations they provide for interfaces. The interface part is used by applications using the plugin for finding the right plugins, and the implementation defines the actual operations.
- Commonly some special information regarding the interface is provided, based on which the right plugin library can be selected.
- **When a plugin is selected for use, its implementation part is instantiated in the memory similarly to normal dynamically linked libraries.**
- **It is possible to load and unload several plugins for the same interface** during the execution of an application, depending on required operations.

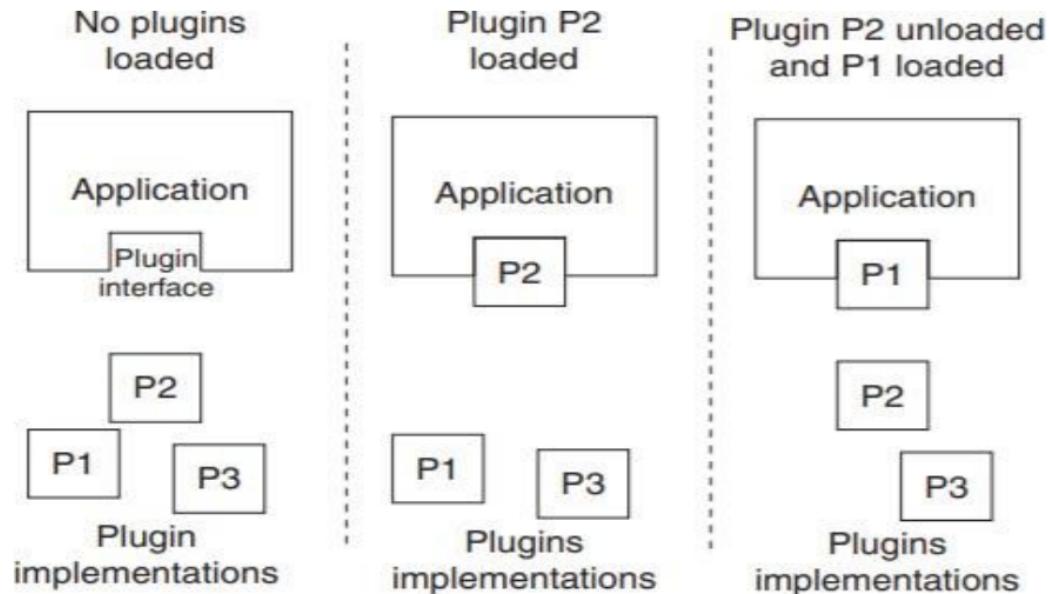


Figure 4.3 Using plugins

One applicable solution for the implementation of plugins is the abstract factory design pattern introduced by Gamma et al. (1995).

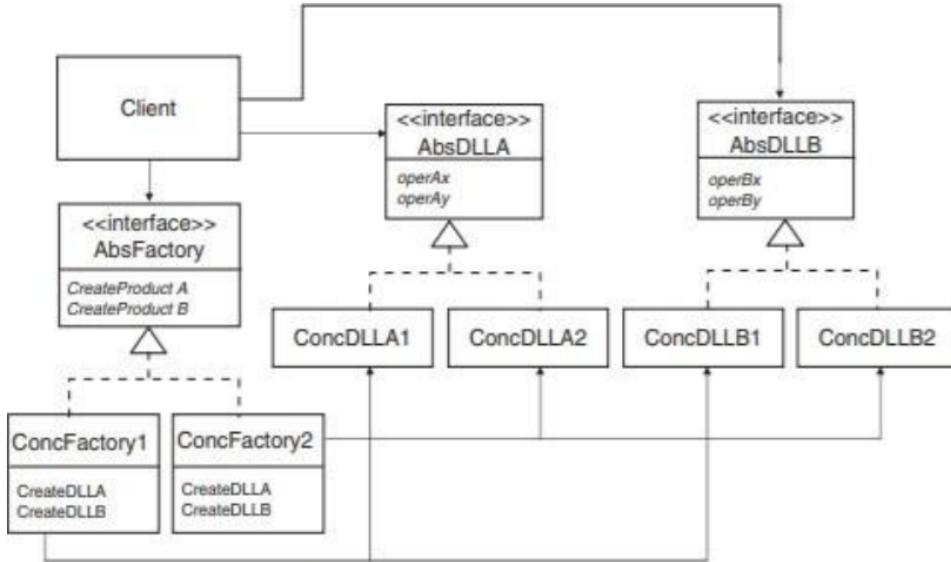


Figure 4.4 Abstract factory

- **Which prefixes Abs and Conc refer to abstract and concrete elements of the design.** In some cases, the programmer is responsible for all the operations, in which

case all the plugins are just plain dynamically linked libraries from which the programmer selects one.

- In other cases, sophisticated support for plugins is implemented, where the infrastructure handles plugin selection based on some heuristics, for instance.
- **The idea of plugins can be applied in a recursive fashion.** This means that a plugin used for specializing some functions of the system can use other plugins in its implementation to allow improved flexibility.

IMPLEMENTATION-LEVEL CONCERNS

To begin with, a framework is commonly used for loading and unloading plugins. This implies a mechanism for extending (or degenerating) an application on the fly when some new services are needed.

Secondly, in order to enable loading, facilities must be offered for searching all the available implementations of a certain interface. This selection process is commonly referred to as resolution. In many implementations, a default implementation is provided if no other libraries are present, especially when a system plugin that can be overridden is used.

Finally, a policy for registering components and removing them is needed in order to enable dynamic introduction of features. This can be based on the use of registry files, or simply on copying certain files to certain locations, for instance.

MANAGING MEMORY CONSUMPTION RELATED TO DYNAMICALLY LINKED LIBRARIES

Memory consumption forms a major concern in the design of software for mobile devices. At the same time, a dynamically linked library is often the smallest unit of software that can be realistically managed when developing software for mobile devices.

MEMORY LIMIT

Setting explicit limits regarding memory usage for all parts of the system is one way to manifest the importance of controlling memory usage. Therefore, make all dynamically linked libraries (and other development-time entities) as well as their developers responsible for the memory they allocate.

INTERFACE DESIGN PRINCIPLES

1. Select the right operation granularity

- In many cases, it is possible to reveal very primitive operations, out of which the clients of a dynamically linked library can then compose more complex operations. In contrast, one can also provide relatively few operations, each of which is responsible for a more complex set of executions.
- A common rule of thumb is to select the granularity of the visible interface operations so that they are logical operations that a client can ask the library to perform, and not to allow setting and getting of individual values (overly simplistic operations) or simply commanding the library to doIt() (overly abstract operations), for instance.

2. Allow client to control transmissions.

- This allows implementations where clients optimize their memory and general resource situation in their behaviors, whereas if the service provider is in control, all the clients get the same treatment, leaving no room for special situations on the client side.

3. Minimize the amount of data to be transmitted.

- For individual functions that call each other this means that the size of the stack remains smaller. Furthermore, there will be less need for copying the objects that are passed from one procedure to another.

4. Select the best way to transmit the data.

There are three fundamentally different ways to transmit data, referred to as lending, borrowing, and stealing. They are described in the following in detail.

1. Lending. When a client needs a service, it provides some resources (e.g. memory) for the service provider to use. The responsibility for the resource remains in the client's hands.

2. Borrowing. When a client needs a service, it expects the service provider to borrow some of its own resources for the client's purposes. The client assumes the responsibility for the deallocation, but uses the service provider's operation for this.

3. Stealing. The service provider allocates some resources whose control is transferred to the client. The owner of the resource is changed, and the client assumes full ownership, including the responsibility for the deallocation.

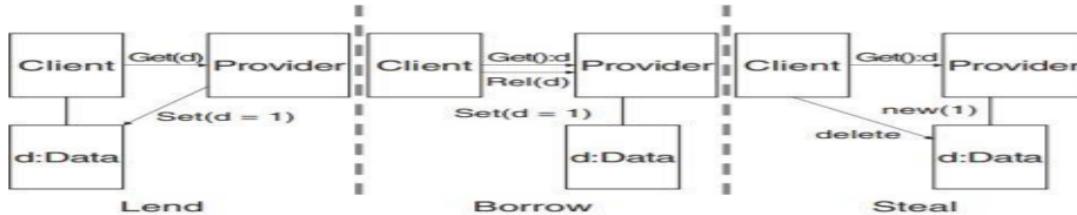


Figure 4.5 Ways to transmit data from a service provider to a client

RULES OF THUMB FOR USING DYNAMICALLY LOADED LIBRARIES

In principle, any unit of software can be implemented as a separate library. In practice, however, creating a complex collection of libraries that depend on each other in an ad-hoc fashion is an error that is to be avoided at all cost.

In the following, we list some candidate justifications for creating a separate dynamically linked library out of a program component.

- **Reusable or shareable components** should be implemented using dynamically loaded libraries, as otherwise all the applications that use the components must contain them separately. This in turn consumes memory. In addition, sharing can take place in a form where the same model, implemented as a dynamically loaded library, is reused in several different devices that require a specialized user interface for each device.

- **Variation or management point** can be preferable to implement in terms of dynamic libraries. This makes variation or management more controlled, as it can be directly associated with a software component. Moreover, the library can be easily changed, if updates or modifications are needed. For instance, interfaces can be treated in this fashion even if the underlying infrastructure (e.g. C++) would not offer such an option. Management can also be associated with scoping in general, allowing that a certain module can be evolved separately.

- **Software development processes** such as automated testing may require that all the input is in a form that can be directly processed. Then, performing the tests may require that binary components are delivered.

- **Organizational unit** can be authorized to compose a single library that is responsible for a certain set of functions. Requesting a library then results in a separate deliverable that can be directly integrated into the final system.

CONCURRENCY AND RESOURCE MANAGEMENT

BRIEFLY EXPLAIN ABOUT CONCURRENCY IN MOBILE DEVICES (PART A, B, C)

EXPLAIN IN DETAIL ABOUT RESOURCE MANAGEMENT IN MOBILE DEVICES (PART B, C)

INTRODUCTION

- The software run inside a mobile device can in general be taken as an event handler, which simply responds to the events received from the outside world.
- The implementation mechanisms for concurrent programming are more or less standardized.
- Threads and processes give boundaries for managing executions and resources.
- In addition, some mechanisms are available to ensure that several threads do not modify the same variables at the same time.

INFRASTRUCTURE FOR CONCURRENT PROGRAMMING

When programming a system where some input is generated by the environment and requires immediate reaction whereas other input leads to extensive executions, parallel processing is usually needed.

Three different cases can be considered:

1. **Executions are unknown to each other. However, they can still affect each other by competing for the same resource, like processor execution time.**
2. **Executions are aware of each other indirectly. For instance, they may have some common shared resource via which they cooperate.**
3. **Executions communicate with each other directly.**

THREADING

- **Threads that wait for stimuli (response or activity) and react to them are a commonly used mechanism for creating highly responsive applications.**
- This allows incoming stimuli to initiate operations, which in turn can generate new stimuli for other threads, or perhaps more generally lead to the execution of some procedures by the threads themselves.
- **The cost of using threads is that each thread essentially requires memory for one execution stack, and causes small overhead during scheduling.**
- Threads can be executed in a pre-emptive or non-pre-emptive fashion.
- The former means that the thread that is being executed can be interrupted, and another thread can be selected for execution. The latter implies that once a thread is being executed, its execution will only be interrupted for another thread when the executing thread is ready.

INTER-THREAD COMMUNICATION

- While threads are a mechanism for creating executions, in order to accomplish operations at a higher level of abstraction, several threads are often needed that cooperate for completing a task.

- For example, establishing and maintaining a phone call requires the cooperation of a user interface, communication protocols, radio interface, a microphone, a speaker, and a unit that coordinates the collaboration.
- **This cooperation requires inter-thread communication.** There are several mechanisms for making threads communicate.

1. **Shared Memory**

- ❖ **Probably the simplest form of communication is the case where threads use a shared variable for their communication.**
- ❖ In most cases, the access to the variable must be implemented such that threads can **temporarily block** each other, so that only one thread at a time performs some operations on the variable.
- ❖ In general, such operations commonly address memory that is shared by a number of threads, **and blocking is needed to ensure that only one thread at a time enters the critical region.**
- ❖ For more complex cases, semaphores are a common technique for ensuring that only one thread at a time enters the critical region

2. **Message Passing**

- ❖ Message passing is another commonly used mechanism for implementing cooperation between threads.
- ❖ In this type of an approach, the idea is that threads can send messages to each other, and that kernel will deliver the messages.
- ❖ The architecture of such a system resembles message-dispatching architecture, where the kernel acts as the bus, and individual processes are the stations attached to it.

COMMON PROBLEMS

Concurrent programs can fail in three different ways.

1. One category of errors in mutual exclusion can lead to locking, where all threads wait for each other to release resources that they can reserve.
2. Secondly, starvation occurs when threads reserve resources such that some particular thread can never reserve all the resources it needs at the same time.
3. Thirdly, errors with regard to atomic operations, i.e., operations that are to be executed in full without any intervention from other threads, can lead to a failure.

MIDP JAVA AND CONCURRENCY

While Java again in principle hides the implementation of the threading model from the user, its details can become visible to the programmer in some cases.

1. Threading in Virtual Machine

- ❖ In the implementation of the Java virtual machine, one important topic is how to implement threads. There are two alternative implementations, one where threads of the underlying operating system are used for implementing Java threads, and the other where the used virtual machine is run in one operating system thread, and it is responsible for scheduling Java threads for execution.
- ❖ The latter types of threads are often referred to as **green threads; one cannot see them from green grass.**
- ❖ The scheme can be considered as a sophisticated form of **event-based programming**, because in essence, the virtual machine simply acts as an event handler and schedules the different threads into execution in accordance to incoming events.

2. Using Threads in Mobile Java

- ❖ Using threads in Java is simple. There is a **type thread that can be instantiated as a Runnable**, which then creates a new thread.
- ❖ The new thread starts its execution from method run after the creator calls the start method of the thread.
- ❖ Mutual exclusion is implemented either in the fashion the methods are called or using the synchronized keyword, which enables monitor-like implementations.

As an example, consider the following case. There is a shared variable that is shared by two classes. The variable is hosted by IncrementerThread, but the other thread (MyThread) is allowed to access the value directly. The hosting thread (instance of IncrementerThread) will only increment the shared integer value.

The definition of the thread is the following:

```
public class IncrementerThread extends Thread
{
    public int i;
    public IncrementerThread()
    {
        i = 0;
        Thread t = new Thread(this);
        t.start();
    }
    public void run()
    {
        for(;;) i++;
    }
}
```

Problems with Java Threading

1. Perhaps the main problem of the scheme presented above is that changing between threads is a very costly operation.
2. The threads are rather independent entities, which is inline with the principle that objects are entities of their own.

RESOURCE MANAGEMENT

DESCRIBE THE USE OF RESOURCE MANAGEMENT (PART B,C)

DISCUSS RESOURCE MANAGEMENT (PART B, C)

RESOURCE-RELATED CONCERNS IN MOBILE DEVICES

A mobile device is a specialized piece of hardware. It has several different types of resources that may require special monitoring and management activities that require scalable yet uniform designs.

OVERVIEW

- In many ways, each resource included in a mobile device can be considered as a potential variation and management point.
- The parts that are associated with the file system and disks in general should form a subsystem.
- We are essentially creating a manager for all aspects of the file system inside a specialized entity. This gives a clear strategy for connecting software to the underlying hardware.

- **First, a device driver addresses the hardware, and on top of the driver, a separate resource manager takes care of higher-level concerns.**
- Subsystems can communicate with each other for instance sending messages to each other.

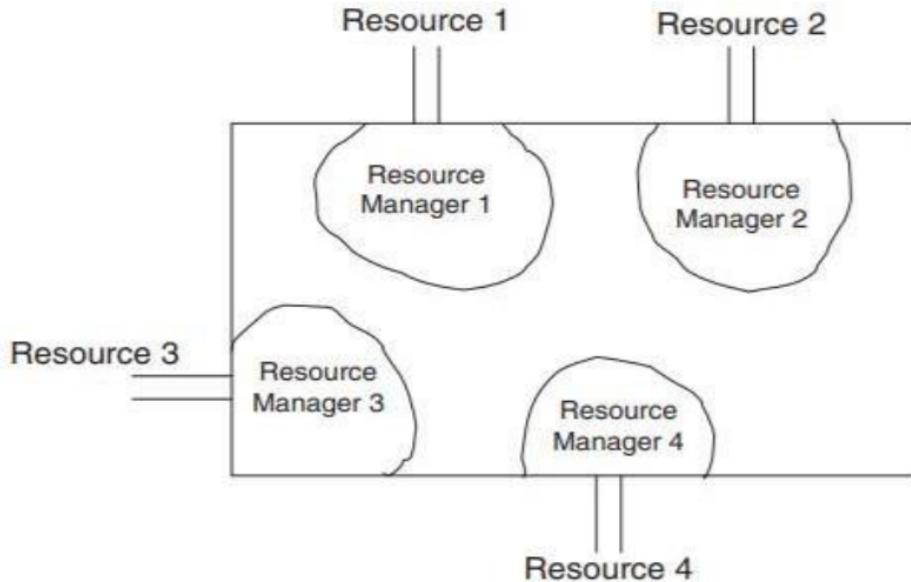


Figure 6.1 Resource managers and hardware interface

- Process boundaries can be used for separating the different resources at the level of an implementation. Unfortunately this essentially makes the option of handling errors that occur in the manager of a certain resource more difficult.
- Another problem in **isolating resource managers is memory protection**: in many cases **resource managers can use the same data but memory protection may require the use of copies**. A practical guideline for designing such isolation is that it should be possible to reconstruct all events for **debugging purposes**.

There are two fundamentally different solutions for embedding resources in the system.

1. The first solution is to **put all resources under one control**. This can be implemented using a monolithic kernel or a virtual machine through which the access to the resources of the device is provided.
2. The alternative is to use an approach **where all resource managers run in different processes and the kernel only has minimal scheduling and interrupt handling responsibility**.

GROUPING RESOURCE MANAGERS

A monolithic design, where several, if not all, resource-related operations are embedded in the OS kernel, requires a design where the kernel includes a large amount of code and auxiliary data.

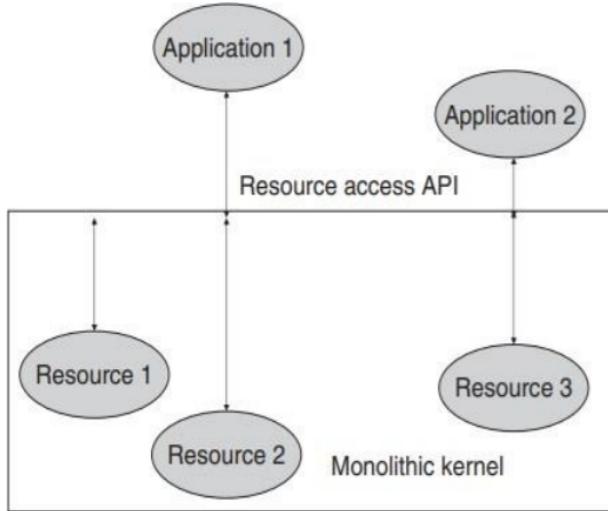


Figure 6.3 Monolithic resource-hosting kernel

- Where ellipses denote resources and application processes, and the monolithic kernel is shown as a rectangle.

- The **interface to the kernel can be understood as an API** to all the resources that are accessed via the kernel, although in a practical implementation an interrupt-like routine is used.
 - A practical example of such a system is Linux, where the kernel is in principle monolithic, but dedicated modules are used for several rather independent tasks, like processor and cache control, memory management, networking stacks, and device and I/O interfaces, to name some examples.
 - Such a system is commonly implemented in terms of (procedural) interfaces between resources.
- ❖ Positive and negative aspects of this approach are the following. **Addressing different parts of the kernel using procedural interfaces can be implemented in a performance-effective fashion as no context switching is required but all the resources can be accessed directly.**
- ❖ The **operating system can serve the requests of programs faster**, in particular when an operation that requires coordination in several resources is needed. On the downside, without careful management, it is possible to create a tangled code base in the kernel, where the different parts of the system are very tightly coupled.

SEPARATING RESOURCE MANAGERS

- **Parallel partitioning of resource-management-related facilities of a mobile device leads to a design where individual resources are managed by separate software modules.**
- These modules can then communicate with each other using messages, leading to an architecture **commonly referred to as message passing architecture**

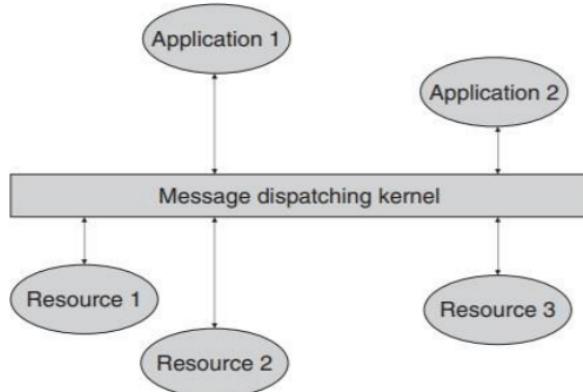


Figure 6.4 Message dispatching kernel

One common implementation for this type of architecture is that all the modules run in a process of their own. Inside a module, **a thread is used to listen to messages**. Whenever a message is received, the thread executes a message-handling procedure, which fundamentally is about event processing. It needs several context switching to avoid this we introduce shared memory.

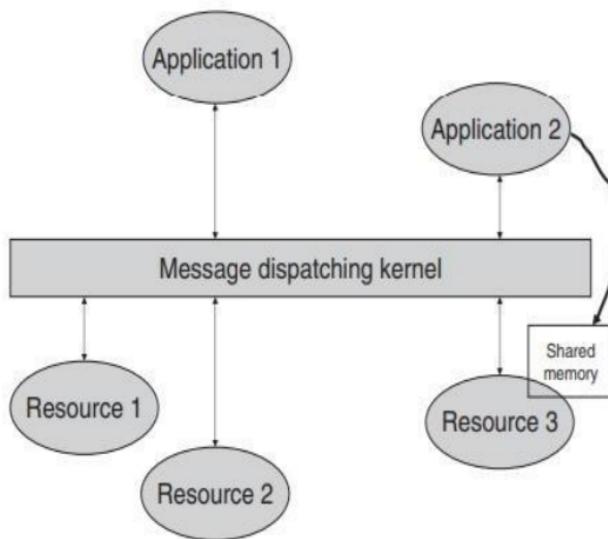


Figure 6.5 Message dispatching and shared memory

RESOURCE-HOSTING VIRTUAL MACHINE

- **One more approach is to introduce a separate software entity, a virtual machine to host resources of the underlying platform.**
- Virtual machines can be of several levels of abstraction, ranging from low-level virtual machines that can be considered a feature of the underlying hardware to complete interpreters that can offer sophisticated services, and their features may vary accordingly.
- The benefits of using a virtual machine in this context are the expected ones.
- Firstly, **porting can be eased**, and in fact it is possible to define a standard execution environment, with well-defined, standardized interfaces, as defined by mobile Java. Moreover, techniques such as dynamic compilation can also be used.

COMMON CONCERNs

There are several common concerns when considering resource use of a mobile device. Many of them are derivatives of scarce hardware resources, but some of them can be traced to the requirements of organizations developing mobile systems. For instance, one can consider the following concerns:

- **Extension and adaptation** is needed for being **able to reuse the same code base in different devices and contexts** whose hardware and software characteristics and available

resources may differ. For instance, some devices can use hardware-supported graphics acceleration, whereas others use the main processor for this task.

- **Performance** requires **special attention, because in many cases mobile device hardware is slower than workstation on the one hand**, and, due to the generic nature of mobile devices as application platforms, optimization for a single purpose is harder than in a purely embedded setting on the other hand.
- **Energy management** is an issue that arises when mobile devices are used more like computers and less like single-purpose devices; being **active consumes more memory**.
- **Internal resource management** is needed for ensuring that the **right resources are available at the right time**. Furthermore, it may be a necessity to introduce features for handling hardware-specific properties. The issue is emphasized by the fact that in many cases, resource management is always running in the background.

REFERENCE

4. Reto Meier, Professional Android 4 Application Development:, Wiley, 1st edition,2012
5. ZigurdMednieks, Laird Dominm G. Blake Meike, Masumi Nakamura, -Programming Android, O'Reilly , 2nd edition,2012
6. Alasdair allan,iphone Programming, O'Reilly , 1st edition,2010.
7. Programming Mobile Devices An Introduction For Practitioners, Tommi Mikkonen, John Wiley & Sons.s

*****UNIT III COMPLETED*****

VASAVI VIDYA TRUST GROUP OF INSTITUTIONS, SALEM- 103

CLASS: I MCA

SUBJECT CODE: MC5209

SUBJECT: MOBILE APPLICATION DEVELOPMENT

UNIT: IV

UNIT IV

**MOBILE OS: ANDROID, IOS - ANDROID APPLICATION ARCHITECTURE - ANDROID
BASIC COMPOENETS - INTENTS AND SERVICES - STORING AND RETRIEVING DATA
- PACKAGING AND DEPLOYMENT - SECURITY AND HACKING.**

What is OS?

The Operating System is a base infrastructure software component of a computerized system. It is a interface between the hardware and software. It controls all basic operations of the computer (or other electronic devices such as PDA, smart phone, etc.). The Operating System allows the user to install and execute third-party applications (commonly called apps for short), usually adding new functionality to the device.

The most popular OS's for mobile devices (smart phones and tablets) are Apple's iOS and Google's Android and they are the only ones.

MOBILE OS

DEFINE MOBILE OS. (PART A)

WHAT IS MEANT BY MOBILE OS? (PART A, B)

- The operating systems that control the mobile device is called mobile OS.
- A mobile operating system is an operating system that helps to run other application software on mobile devices.
- The operating systems found on smart phones include Symbian OS, iPhone OS, RIM's BlackBerry, Windows Mobile, Palm WebOS, Android, and Maemo. Android, WebOS, and Maemo are all derived from Linux.
- The iPhone OS originated from BSD and NeXTSTEP, which are related to UNIX.
- It combines the beauty of computer and hand use devices. It typically contains a cellular built-in modem and SIM tray for telephony and internet connections.



Layers of operating system

POPULAR PLATFORMS OF THE MOBILE OS

1. Android OS: The Android operating system is the most popular operating system today. It is a mobile OS based on the **Linux Kernel** and **open-source software**. The android operating system was developed by **Google**. The first Android device was launched in **2008**.

2. Bada (Samsung Electronics): Bada is a Samsung mobile operating system that was launched in 2010. The Samsung wave was the first mobile to use the bada operating system. The bada operating system offers many mobile features, such as 3-D graphics, application installation, and multipoint-touch.

3. BlackBerry OS: The BlackBerry operating system is a mobile operating system developed by **Research In Motion** (RIM). This operating system was designed specifically for BlackBerry handheld devices. This operating system is beneficial for the corporate users because it provides synchronization with Microsoft Exchange, Novell GroupWise email, Lotus Domino, and other business software when used with the BlackBerry Enterprise Server.

4. iPhone OS / iOS: The iOS was developed by the Apple inc for the use on its device. The iOS operating system is the most popular operating system today. It is a very secure operating system. The iOS operating system is not available for any other mobiles.

5. Symbian OS: Symbian operating system is a mobile operating system that provides a high-level of integration with communication. The Symbian operating system is based on the java language. It combines middleware of wireless communications and personal information management (PIM) functionality. The Symbian operating system was developed by **Symbian Ltd** in **1998** for the use of mobile phones. **Nokia** was the first company to release Symbian OS on its mobile phone at that time.

6. Windows Mobile OS: The window mobile OS is a mobile operating system that was developed by **Microsoft**. It was designed for the pocket PCs and smart mobiles.

7. Harmony OS: The harmony operating system is the latest mobile operating system that was developed by Huawei for the use of its devices. It is designed primarily for IoT devices.

8. Palm OS: The palm operating system is a mobile operating system that was developed by **Palm Ltd** for use on personal digital assistants (PADs). It was introduced in **1996**. Palm OS is also known as the **Garnet OS**.

9. WebOS (Palm/HP): The WebOS is a mobile operating system that was developed by **Palm**. It based on the **Linux Kernel**. The HP uses this operating system in its mobile and touchpads.

ANDROID OS

WHAT IS MEANT BY ANDROID OS? (PART A)

WRITE ABOUT ANDROID OS FEATURE, ADVANTAGES, HISTORY (PART B, C)

- Android is an operating system. That is, **it's software that connects hardware to software and provides general services.**
- Android is an open source operating system produced by the Open Handset Alliance (OHA). Java language is mainly used to write the android code even though other languages can be used.
- The goal of android project is to create a successful real-world product that improves the mobile experience for end users.

What is Open Handset Alliance (OHA)?

- It's a consortium of 84 companies such as google, samsung, AKM, synaptics, KDDI, Garmin, Teleca, Ebay, Intel etc.
- **It was established on 5th November, 2007, led by Google.** It is committed to advance open standards, provide services and deploy handsets using the Android Platform.

Features of Android

- After learning what is android, let's see the features of android. The important features of android are given below:
 - 1) It is open-source.
 - 2) Anyone can customize the Android Platform.
 - 3) There are a lot of mobile applications that can be chosen by the consumer.
 - 4) It provides many interesting features like weather details, opening screen, live RSS (Really Simple Syndication) feeds etc.

Categories of Android applications

There are many android applications in the market. The top categories are:

- **Entertainment**
- **Tools**
- **Communication**
- **Productivity**

- **Personalization**
- **Music and Audio**
- **Social**
- **Media and Video**
- **Travel and Local etc.**

HISTORY OF ANDROID

The history and versions of android are interesting to know. The code names of android ranges from A to J currently, such as **Aestro, Blender, Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich, Jelly Bean, KitKat and Lollipop**. Let's understand the android history in a sequence.

- 1) Initially, **Andy Rubin** founded Android Incorporation in Palo Alto, California, United States in October, 2003.
- 2) In 17th August 2005, Google acquired android Incorporation. Since then, it is in the subsidiary of Google Incorporation.
- 3) The key employees of Android Incorporation are **Andy Rubin, Rich Miner, Chris White** and **Nick Sears**.
- 4) Originally intended for camera but shifted to smart phones later because of low market for camera only.

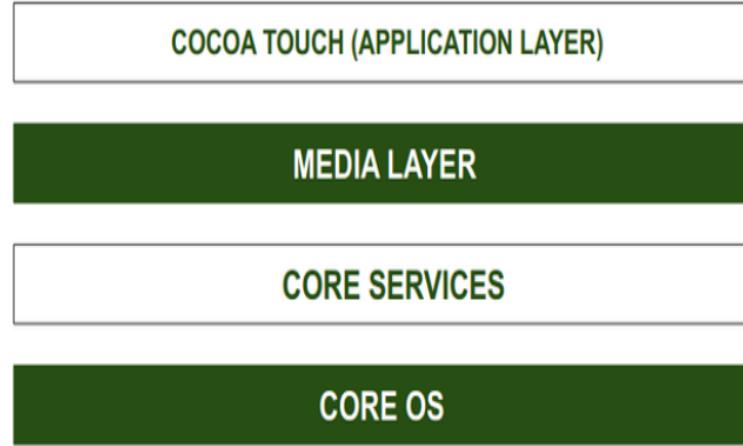
- 5) Android is the nick name of Andy Rubin given by coworkers.
- 6) In 2007, Google announces the development of android OS.
- 7) In 2008, HTC launched the first android mobile.



IOS

WRITE ABOUT IOS. (PART B, C)

- **OS (formerly iPhone OS) is a mobile operating system created and developed by Apple Inc.**
- **IOS is a Mobile Operating System that was developed by Apple Inc. for iPhones, iPads, and other Apple mobile devices.**
- iOS is the second most popular and most used Mobile Operating System after Android.
- The structure of the **iOS operating System is Layered based.** Its communication doesn't occur directly.
- The layer's between the Application Layer and the Hardware layer will help for Communication. The lower level gives basic services on which all applications rely and the higher-level layers provide graphics and interface-related services.
- **Most of the system interfaces come with a special package called a framework.**
- **A framework is a directory that holds dynamic shared libraries** like .a files, header files, images, and helper apps that support the library. Each layer has a set of frameworks that are helpful for developers.



Architecture of IOS

CORE OS LAYER

All the IOS technologies are built under the lowest level layer i.e. Core OS layer. These technologies include:

1. Core Bluetooth Framework

2. External Accessories Framework
3. Accelerate Framework
4. Security Services Framework
5. Local Authorization Framework etc.

It supports **64 bit** which enables the application to run faster

CORE SERVICES LAYER

- Some important frameworks are present in the CORE SERVICES Layer which helps the iOS operating system to cure itself and provide better functionality.
 - It is the 2nd lowest layer in the Architecture as shown above.
1. **Address Book Framework** - The Address Book Framework provides access to the contact details of the user.
 2. **Cloud Kit Framework** - This framework provides a medium for moving data between your app and iCloud.
 3. **Core Data Framework** - This is the technology that is used for managing the data model of a Model View Controller app.
 4. **Core Foundation Framework** - This framework provides data management and service features for iOS applications.

5. **Core Location Framework** - This framework helps to provide the location and heading information to the application.
6. **Core Motion Framework** - All the motion-based data on the device is accessed with the help of the Core Motion Framework.
7. **Foundation Framework** - Objective C covering too many of the features found in the Core Foundation framework.
8. **HealthKit Framework** - This framework handles the health-related information of the user.
9. **HomeKit Framework** - This framework is used for talking with and controlling connected devices with the user's home.
10. **Social Framework** - It is simply an interface that will access users' social media accounts.
11. **StoreKit Framework** - This framework supports for buying of contents and services from inside iOS apps.

MEDIA LAYER

Using the media layer, we will enable all graphics video, and audio technology of the system. This is the second layer in the architecture. The different frameworks of MEDIA layers are:

1. **UIKit Graphics** - This framework provides support for designing images and animating the view content.
2. **Core Graphics Framework** - This framework support 2D vector and image-based rendering ad it is a native drawing engine for iOS.
3. **Core Animation** - This framework helps in optimizing the animation experience of the apps in iOS.
 4. **Media Player Framework** - This framework provides support for playing the playlist and enables the user to use their iTunes library.
5. **AV Kit** - This framework provides various easy-to-use interfaces for video presentation, recording, and playback of audio and video.
6. **Open AL** - This framework is an Industry Standard Technology for providing Audio.
7. **Core Images** - This framework provides advanced support for motionless images.
8. **GL Kit** - This framework manages advanced 2D and 3D rendering by hardware-accelerated interfaces.

COCOA TOUCH

COCOA Touch is also known as the application layer which acts as an interface for the user to work with the iOS Operating system. It supports touch and motion events and many more features.

1. EvenKit Framework

This framework shows a standard system interface using view controllers for viewing and changing events.

2. GameKit Framework

This framework provides support for users to share their game-related data online using a Game Center.

3. MapKit Framework

This framework gives a scrollable map that one can include in your user interface of the app.

4. PushKit Framework

This framework provides registration support.

FEATURES OF IOS OPERATING SYSTEM:

1. Highly Securer than other operating systems.
2. iOS provides multitasking features like while working in one application we can switch to another application easily.
3. iOS's user interface includes multiple gestures like swipe, tap, pinch, Reverse pinch.
4. iBooks, iStore, iTunes, Game Center, and Email are user-friendly.
5. It provides Safari as a default Web Browser.

6. It has a powerful API and a Camera.
7. It has deep hardware and software integration

APPLICATIONS OF IOS OPERATING SYSTEM

1. iOS Operating System is **the Commercial Operating system of Apple** Inc. and is popular for its security.
2. iOS operating system comes with pre-installed apps which were developed by Apple like Mail, Map, TV, Music, Wallet, Health, and Many More.
3. Swift Programming language is used for Developing Apps that would run on IOS Operating System.
4. In iOS Operating System we can perform Multitask like Chatting along with Surfing on the Internet.

ADVANTAGES OF IOS OPERATING SYSTEM

1. More secure than other operating systems.
2. Excellent UI and fluid responsive
3. Suits best for Business and Professionals
4. Generate Less Heat as compared to Android.

DISADVANTAGES OF IOS OPERATING SYSTEM

1. More Costly.
2. Less User Friendly as Compared to Android Operating System.
3. Not Flexible as it supports only IOS devices.
4. Battery Performance is poor.

ANDROID APPLICATION ARCHITECTURE

EXPLAIN THE ARCHITECTURE OF ANDROID SYSTEM WITH NEAT DIAGRAM. (PART B, C)

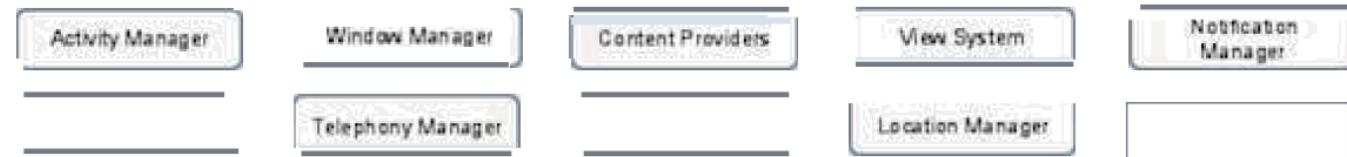
EXPLAIN THE LAYERS OF ANDROID ARCHITECTURE. (PART B, C)

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.

Applications



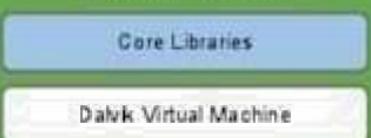
Application Framework



Libraries



Android Runtime



Linux Kernel



LINUX KERNEL

- At the bottom of the layers is Linux - Linux 3.6 with approximately 115 patches.
- This **provides a level of abstraction between the device hardware and it contains all the essential hardware drivers like camera, keypad, display etc.**
- 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 LIBRARIES

This category encompasses those Java-based libraries that are specific to Android development.

- **android.app** – Provides access to the application model and is the cornerstone of all Android applications.

- **android.content** – Facilitates content access, publishing and messaging between applications and application components.
- **android.database** – Used to access data published by content providers and includes SQLite database management classes.
- **android.opengl** – A Java interface to the OpenGL ES 3D graphics rendering API.
- **android.os** – Provides applications with access to standard operating system services including messages, system services and inter-process communication.
- **android.text** – Used to render and manipulate text on a device display.
- **android.view** – The fundamental building blocks of application user interfaces.
- **android.widget** – A rich collection of pre-built user interface components such as buttons, labels, list views, layout managers, radio buttons etc.
- **android.webkit** – A set of classes intended to allow web-browsing capabilities to be built into applications.

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.

The Android framework includes the following key services –

- **Activity Manager** – Controls all aspects of the application lifecycle and activity stack.
- **Content Providers** – Allows applications to publish and share data with other applications.
- **Resource Manager** – Provides access to non-code embedded resources such as strings, color settings and user interface layouts.
- **Notifications Manager** – Allows applications to display alerts and notifications to the user.
- **View System** – An **extensible set of views used to create application user interfaces.**

ANDROID BASIC COMPONENTS

WHAT ARE THE BASIC COMPONENTS OF ANDROID? (PART A)

EXPLAIN IN SHORT NOTES ABOUT ANDROID COMPONENTS. (PART B, C)

In android, **application components** are the basic building blocks of an application and these components will act as an entry point to allow system or user to access our app.

An android **component** is simply a piece of code that has a well defined life cycle e.g. **Activity, Receiver, and Service etc.**

The **core building blocks** or **fundamental components** of android are **activities, views, intents, services, content providers, fragments and AndroidManifest.xml.**

The following are the basic core application components that can be used in Android application.

- Activities
- Services
- Content Providers



- Broadcast Receivers
- Intents

ACTIVITIES

- An activity represents a single screen with a user interface, in-short Activity performs actions on the screen.
- Activities are said to be the presentation layer of our applications. The UI of our application is built around one or more extensions of the Activity class.
- For example, an email application might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. If an application has more than one activity, then one of them should be marked as the activity that is presented when the application is launched.
- An activity is implemented as a subclass of **Activity** class as follows –

```
public class MainActivity extends Activity
{  
}
```

SERVICES

- A service is a component that runs in the background to perform long-running operations.

- Services are like **invisible workers of our app.**
- These components run at the backend, updating your data sources and Activities, triggering Notification, and also broadcast Intents. They also perform some tasks when applications are not active.
- For example, a service might play music in the background while the user is in a different application, or it might fetch data over the network without blocking user interaction with an activity.
- A service is implemented as a subclass of **Service** class as follows –

```
public class MyService extends Service
{
}
```

We have two types of services available in android, those are

- ❖ Local Services
- ❖ Remote Services

Local service is accessed from within the application whereas remote service is accessed remotely from other applications running on the same device.

CONTENT PROVIDERS

- **It is used to manage and persist the application data also typically interacts with the SQL database.**
- **They are also responsible for sharing the data beyond the application boundaries.**
- The Content Providers of a particular application can be configured to allow access from other applications, and the Content Providers exposed by other applications can also be configured.
- A content provider component supplies data from one application to others on request. Such requests are handled by the methods of the *ContentResolver* class. The data may be stored in the file system, the database or somewhere else entirely.
- A content provider is implemented as a subclass of **ContentProvider** class and must implement a standard set of APIs that enable other applications to perform transactions.

```
public class MyContentProvider extends ContentProvider {  
    public void onCreate() {}  
}
```

BROADCAST RECEIVERS

- **Broadcast Receivers simply respond to broadcast messages from other applications or from the system.**
- **In android, Broadcast Receiver is a component that will allow a system to deliver events to the app like sending a low battery message to the app. The apps can also initiate broadcasts to let other apps know that required data available in a device to use it.**
- Generally, we use Intents to deliver broadcast events to other apps and Broadcast Receivers use status bar notifications to let the user know that broadcast event occurs.
- For example, applications can also initiate broadcasts to let other applications know that some data has been downloaded to the device and is available for them to use, so this is broadcast receiver who will intercept this communication and will initiate appropriate action.
- A broadcast receiver is implemented as a subclass of **BroadcastReceiver** class and each message is broadcaster as an **Intent** object.

```
public class MyReceiver extends BroadcastReceiver {  
    public void onReceive(context, intent) {}
```

ANDROID INTENTS

- It is a powerful inter-application message-passing framework. They are extensively used throughout Android.
- Intents can be used to start and stop Activities and Services, to broadcast messages system-wide or to an explicit Activity, Service or Broadcast Receiver or to request action be performed on a particular piece of data.

In android, Intent is a messaging object which is used to request an action from another component.

In android, intents are mainly used to perform the following things.

- **Starting an Activity**
- **Starting a Service**
- **Delivering a Broadcast**

There are two types of intents available in android, those are

1. Implicit Intents

2. Explicit Intents

ADDITIONAL COMPONENTS

In android, we have additional components which are used to build the relationship between the above components (Activities, Intents, Content Providers, Services and Broadcast Receivers) to implement our application logic, those are

COMPONENT	DESCRIPTION
Fragments	These are used to represent the portion of user interface in an activity
Layouts	These are used to define the user interface (UI) for an activity or app

COMPONENT	DESCRIPTION
Views	These are used to build a user interface for an app using UI elements like buttons, lists, etc.
Resources	To build an android app we required external elements like images, audio files, etc. other than coding
Manifest File	It's a configuration file (AndroidManifest.xml) for the application and it will contain the information about <u>Activities</u> , <u>Intents</u> , <u>Content Providers</u> , <u>Services</u> , <u>Broadcast Receivers</u> , permissions, etc.

INTENTS AND SERVICES

DEFINE INTENT (PART A)

EXPLAIN IN DETAIL ABOUT INTENT AND SERVICES IN ANDROID. (PART B,C)

WRITE SHORT NOTES ABOUT INTENTS AND ITS TYPES. (PART B)

WRITE SHORT NOTES ABOUT SERVICES IN ANDROID (PART B, C)

In Android, it is quite usual for users to witness a jump from one application to another as a part of the whole process, for example, searching for a location on the browser and witnessing a direct jump into Google Maps or receiving payment links in Messages Application (SMS) and on clicking jumping to PayPal or GPay (Google Pay).

This process of taking users from one application to another is achieved by passing the **Intent** to the system. Intents, in general, are used for navigating among various activities within the same application, but note, is not limited to one single application, i.e., they can be utilized from moving from one application to another as well.

DEFINITION

- Intent is a messaging object that you can use to request an action from an app component.
- Intent is basically an intention to do an action. It's a way to communicate between Android components to request an action from a component, by different components.
- **Intents** could be Implicit, for instance, calling intended actions and explicit as well, such as opening another activity after some operations like onClick or anything else.

- Below are some applications of Intents:
- 1. Sending the User to Another App**
 - 2. Getting a Result from an Activity**
 - 3. Allowing Other Apps to Start Your Activity**

Some Important Method of Intent and its Description

Context.startActivity()	This is to launch a new activity or get an existing activity to be action.
Context.startService()	This is to start a new service or deliver instructions for an existing service.
Context.sendBroadcast()	This is to deliver the message to broadcast receivers.

DEEP LINKING

- **Deep Link is an URL that redirects the device to the API of that Missing Application and then service is run on the system to check if a version of that application exists on the device.**
- For time being, let's assume that the application is not available on the device and no previous versions ever existed. This service then makes a call to the *Play Store* from the device and the application appears, just a matter of download.

USES OF INTENT IN ANDROID

1. To start an Activity

An Activity represents a single screen in an app. You can start a new instance of an Activity by passing an Intent to `startActivity()`. The Intent describes the activity to start and carries any necessary data along.

2. To start a Service

A Service is a component that performs operations in the background and does not have a user interface. You can start a service to perform a one-time operation(such as downloading a file) by passing an Intent to `startService()`. The Intent describes which service to start and carries any necessary data.

3. To deliver a Broadcast

A broadcast is a message that any app can receive. The system delivers various broadcasts for system events, such as when the system boots up or the device starts charging. You can deliver a broadcast to other apps by passing an Intent to sendBroadcast() or sendOrderedBroadcast().

TYPES OF ANDROID INTENTS

There are two types of intents in android:

- 1. Implicit and**
- 2. Explicit.**

Implicit Intent

Implicit Intent doesn't specify the component. In such a case, intent provides information on available components provided by the system that is to be invoked. For example, you may write the following code to view the webpage.

- In Implicit Intents we do not need to specify the name of the component. We just specify the Action which has to be performed and further this action is handled by the component of another application.
- The basic example of implicit Intent is to open any web page

Let's take an example to understand Implicit Intents more clearly. We have to open a website using intent in your application. See the code snippet given below

```
Intent intentObj = new Intent(Intent.ACTION_VIEW);
intentObj.setData(Uri.parse("https://www.abhiandroid.com"));
startActivity(intentObj);
```

Example:

In the below images, no component is specified, instead an action is performed i.e. a webpage is going to be opened. As you type the name of your desired webpage and click on '_CLICK' button. Your webpage is opened.

Implicit Intent Example

tp / ee o ee o g

GeeksforGeeks
.....,.....,.....,.....

Google Custom Search



Courses

Suggest an Article

[https:// www.geeksforgeeks.org](https://www.geeksforgeeks.org)

CLICK

CLICK

Must Do Coding Questions for
Companies like Amazon, Microsoft,

As the placement season is back so
are we to help you see the interview
We have selected some most commonly
asked and must do. Read More 3.4

Explicit Intent

Explicit Intent specifies the component. In such a case, intent provides the external class to be invoked.

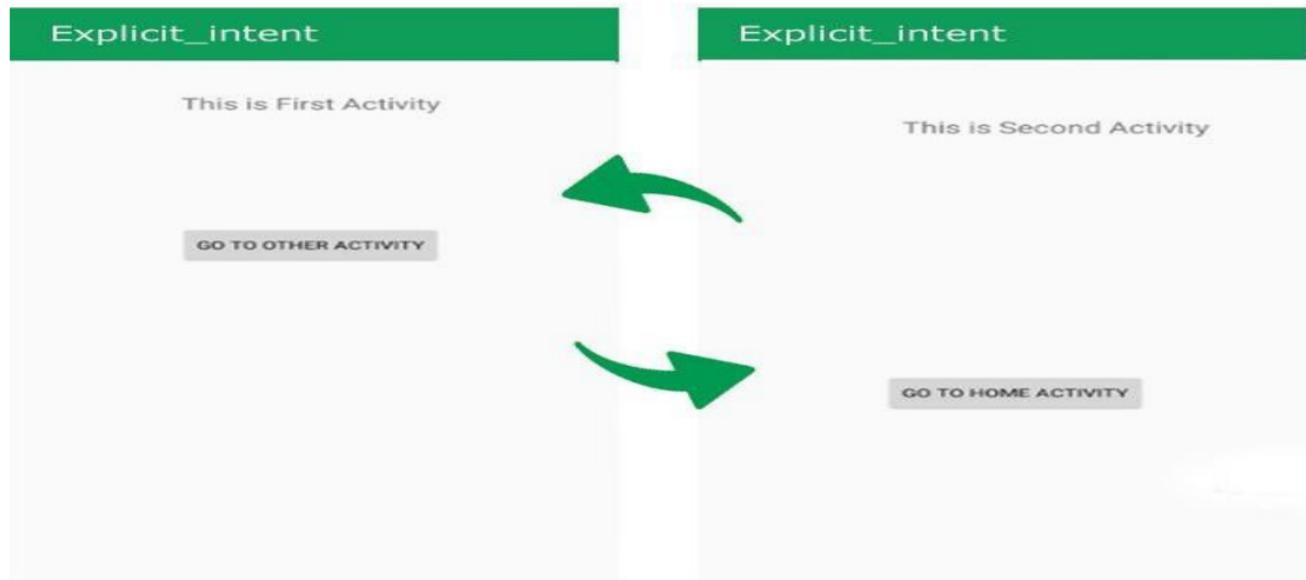
- Explicit Intents are used to connect the application internally.
- In Explicit we use the name of component which will be affected by Intent. For Example: If we know class name then we can navigate the app from One Activity to another activity using Intent. In the similar way we can start a service to download a file in background process.

Explicit Intent **work internally within an application to perform navigation and data transfer.** The below given code snippet will help you understand the concept of Explicit Intents

```
Intent intent = new Intent(getApplicationContext(), SecondActivity.class);
startActivity(intent);
```

Example:

In the below example, there are two activities (FirstActivity, SecondActivity). When you click on the “GO TO OTHER ACTIVITY” Button in the FirstActivity, then you move to the SecondActivity. When you click on the “GO TO HOME ACTIVITY” button in the SecondActivity, then you move to the first activity. This is getting done through Explicit Intent.



SERVICES

DESCRIBE THE SERVICES PROVIDED BY THE ANDROID (PART B , C)

- Android offers the Service class to create application components specifically to handle operations and functionality that should run invisibly, without a user interface.
- A service is a component that runs in the background to perform long- running operations without needing to interact with the user and it works even if application is destroyed.
- A service is a component which runs in the background without direct interaction with the user. As the service has no user interface, it is not bound to the lifecycle of an activity.
- By using Services, you can ensure that your applications continue to run and respond to events, even when they're not in active use. Services run without a dedicated GUI, but, like Activities and Broadcast Receivers, they still execute in the main thread of the application's process.
- To help keep your applications responsive, Unlike Activities, which present a rich graphical interface to users, Services run in the background updating your Content Providers, firing Intents, and triggering notifications.
- They are the perfect way to perform regular processing or handle events even after your application's Activities are invisible, inactive, or have been closed. With no visual interface,

Services are started, stopped, and controlled from other application components including other Services, Activities, and Broadcast Receivers.

- If your application regularly, or continuously, performs actions that don't depend directly on user input, Services may be the answer.

Started Services receive higher priority than inactive or invisible Activities, making them less likely to be terminated by the run time's resource management. The only time Android will stop a Service prematurely is when it's the only way for a foreground Activity to gain required resources; if that happens, your Service will be restarted automatically when resources become available.

Applications that update regularly but only rarely or intermittently need user interaction are good candidates for implementation as Services. MP3 players and sports-score monitors are examples of applications that should continue to run and update without an interactive visual component (Activity) visible.

- There is no **onPause()** or **onResume()** in Services as in Activity.
- Services doesn't have UI.

SERVICES CAN BE OF TWO TYPES – STARTED AND BOUND

STARTED SERVICE

Started services are those that are launched by other application components like an Activity or a Broadcast Receiver. They can run indefinitely in the background until stopped or destroyed by the system to free up resources.

BOUND SERVICES

A Bound Service is the server in a client- server interface. A Bound Service allows components (such as Activities) to bind to the service, send requests, receive responses and even perform inter-process communication (IPC).

While working with Android services, there comes a situation where we would want the service to communicate with an activity. To accomplish this task one has to bind a service to an activity, this type of service is called an android bound service. After a service is bound to an activity one can return the results back to the calling activity.

Onbind()

The system calls this method when another component wants to bind with the service by calling bindService(). If you implement this method, you must provide an interface that clients use to communicate with the service, by returning an *IBinder* object. You must always implement this method, but if you don't want to allow binding, then you should return *null*.

Onunbind()

The system calls this method when all clients have disconnected from a particular interface published by the service. **Started services cannot return results/values or interact with its starting component. Bound services on the other hand can send data to the launching component (client).** So for example a bound service might be playing an audio file and sending data regarding audio start/pause/stop and the time elapsed to the launching Activity component so that the UI can be updated accordingly.

UNDERSTANDING STARTED AND BOUND SERVICE BYBACKGROUND MUSIC EXAMPLE

Suppose, I want to play music in the background, so call start Service () method. But I want to get information of the current song being played, I will bind the service that provides information about the current song.

RUNNING A SERVICE IN THE FOREGROUND

A Foreground Service is one where the user is actively aware of it putting it on high priority hence won't be killed when the system is low on memory. It must provide a notification for the status bar which cannot be dismissed unless the Service is stopped or removed from the foreground. A good example of such a notification is a music player that shows the current song and other action buttons like play/pause, next, previous, etc. in a notification in the status bar. VOIP calls or file download apps could also start a foreground Service and show similar notifications.

STOPPING A SERVICE

You stop a service via the stopService() method. No matter how frequently you called the startService(intent) method, one call to the stopService() method stops the service. A service can terminate itself by calling the stopSelf() method. This is typically done if the service finishes its work.

TOASTS

A toast provides simple feedback about an operation in a small popup. It only fills the amount of space required for the message and the current activity remains visible and

interactive. For example, navigating away from an email before you send it triggers a "Draft saved" toast to let you know that you can continue editing later. **Toasts automatically disappear after a timeout.**

Android offers several techniques for applications to communicate with users without an Activity. *Toasts* are a transient, non-modal dialog-box mechanism used to display information to users without stealing focus from the active application. Where **Toasts are silent and transient, Notifications represent a more robust mechanism for alerting users.** In many cases, when the user isn't actively using the mobile phone it sits silent and unwatched in a pocket or on a desk until it rings, vibrates, or flashes. Should a user miss these alerts, status bar icons are used to indicate that an event has occurred. All these attention-grabbing antics are available to your Android application through Notifications.

STORING AND RETRIEVING DATA

WHAT IS THE MECHANISM USED FOR STORING AND RETRIEVING DATA IN ANDROID? (PART B, C)

Android provides several ways to store and share data, including access to the filesystem, a local relational database through SQLite, and a preferences system that allows you to store simple key/value pairs within applications.

Android also allows applications to share data through a clever URI-based approach called a ContentProvider. This technique combines several other Android concepts, such as the URI-based style of intents and the Cursor result set seen in SQLite, to make data accessible across different applications.

- **Shared Preferences**
- **Internal Storage**
- **External Storage**
- **SQLite Databases**
- **Network Connection**

DIFFERENT STORAGE OPTIONS IN ANDROID



Shared Preferences

- Used to store the data in Key Value pairs.

SQLite

- Used to store private data for the app, here the structured data is stored.

File Storage

- Used to store raw files on memory card or SD-card.

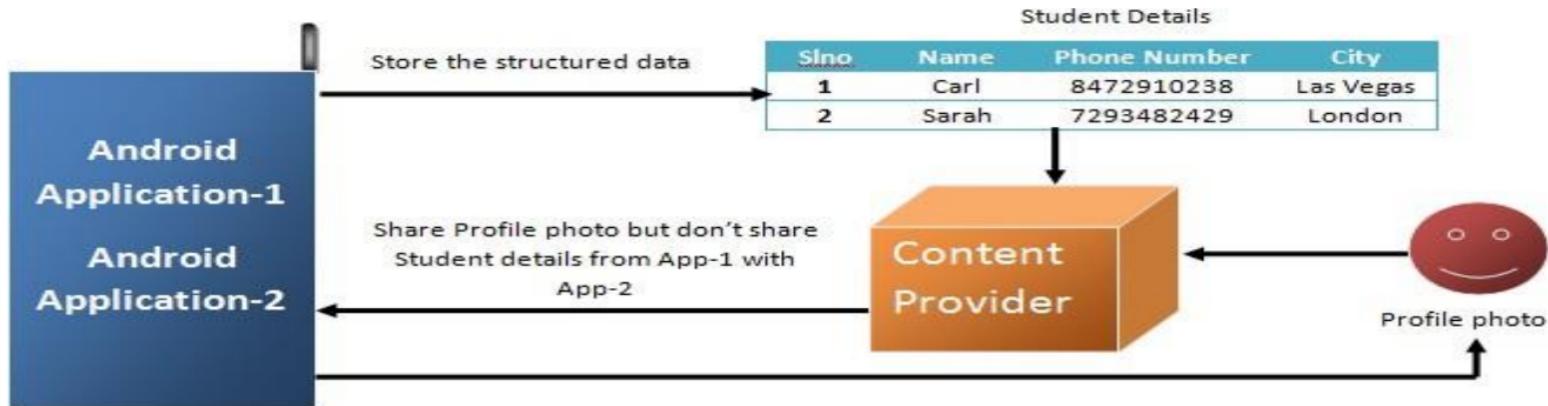
Content Providers

- Used to store semi-structured data with user configurable data access.

Cloud Storage

- Used to store third party data storage. That involves Parse API, Google Skydive etc

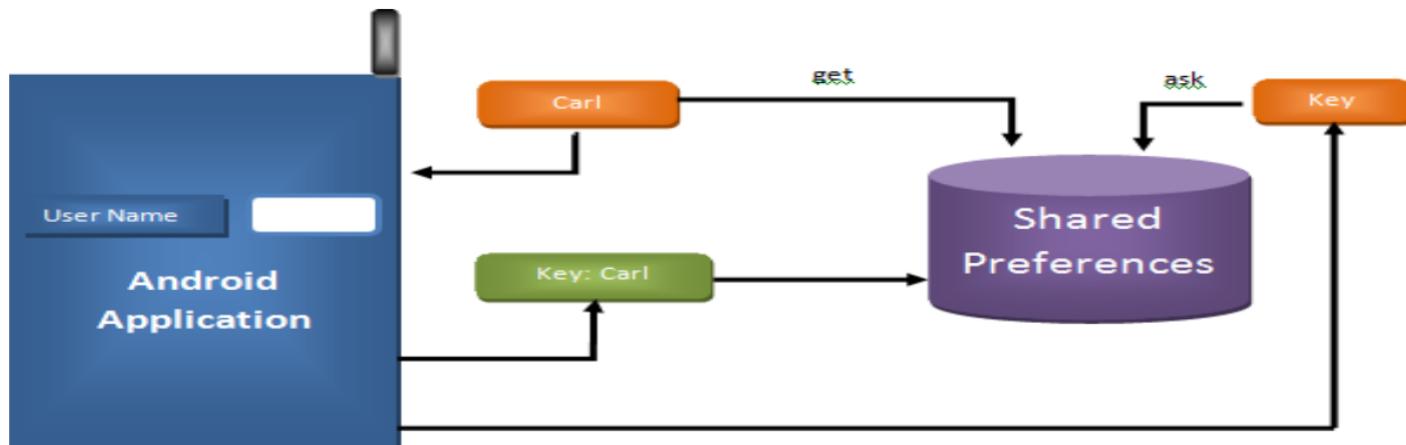
CONTENT PROVIDERS



- Consider the structured data added to the device from application1 is not accessible to another application2 present in the same device but the profile photo added to the device by application1 is available to the application2 running in the same device
- Consider android device as a city, the applications in it are the houses in the city, people in the houses(application) are the data.

- Now content provider is like an broker in the city(android device). This broker provide access for the people in the city for finding different houses referring as the content provider in the android device provide access for the data in the device for different applications.

SHARED PREFERENCES



- Consider I have an App say a Face book App which I use to log in to my account.

- Now the very first time I enter my username and password to get access to my account. Say I log out of the application an hour later again I use the same Face book App to login again to my application.
- I have to enter username and password again to login to my account and I set a theme to my application and other settings on how my app looks in my current phone
- This is un-necessary because consider I am using my phone to login to the application. So I will always use my phone to login again and again, thus entering my credentials again and again is more work shows it's not a user friendly app
- **Shared Preferences is very handy in such scenarios where I can use its feature to share my data in a xml file Which physically exists in the Android app installed in my phone which is not destroyed even if the app is closed.** Here we can save user preferences data of the current application.

- SharedPreferences is used by apps to save data in name-values pairs, like a Bundle

Value

- SharedPreferences only allows you to save primitive data types (that is, booleans, floats, longs, ints, and strings)

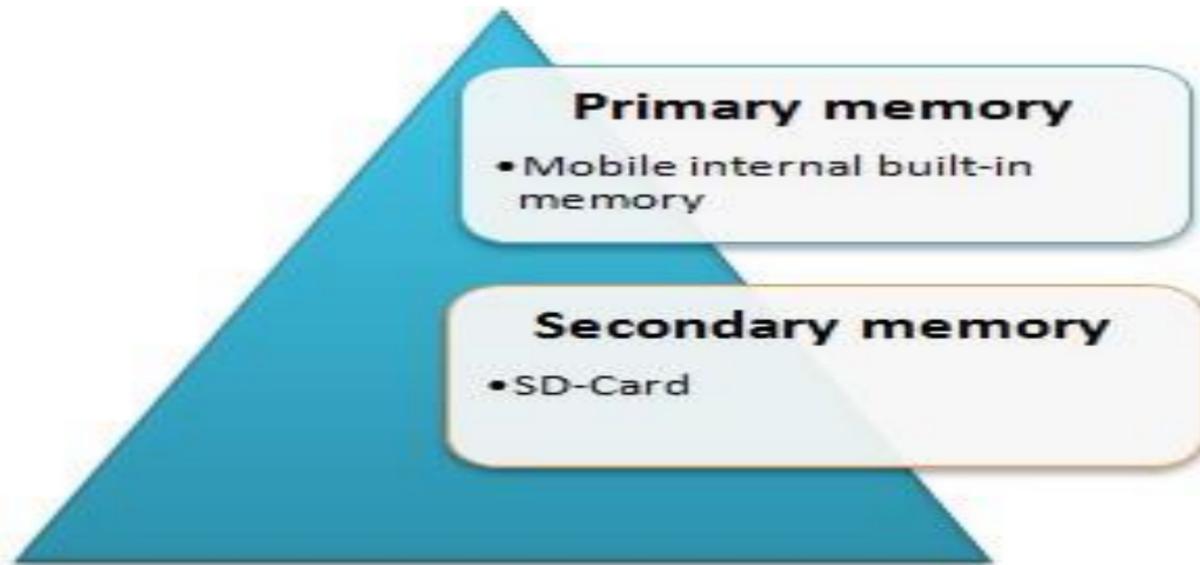
key value

last have ?unr.

location | Earth

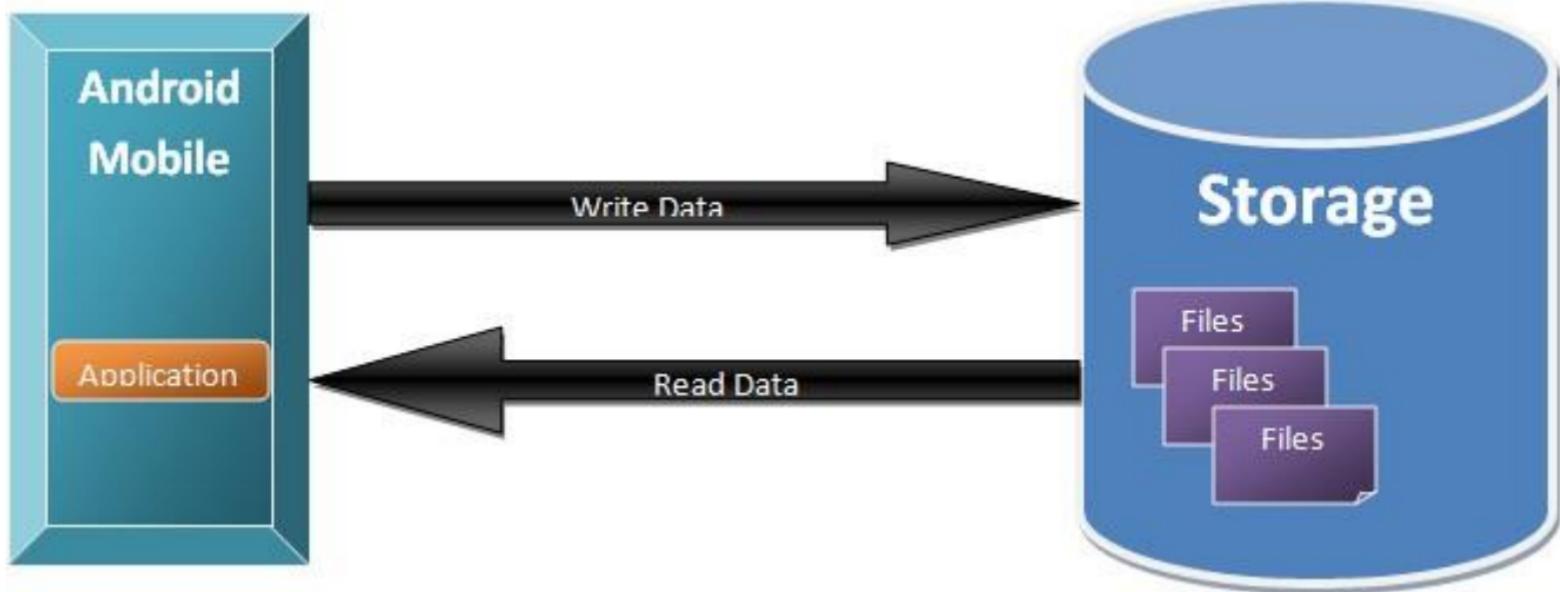
- Application preferences are simple name/value pairs like `-greeting=hello name` or `-sound = ofl`
- To work with preferences, Android offers an extremely simple approach
- **Preferences can only be shared with other components in the same package**
- **Preferences cannot be shared across packages**
- **Private preferences will not be shared at all**
- Storage location is not defined and inaccessible for other applications

FILE STORAGE



- In Android we can use the device storage space to store the data in it for the applications. The type of data involves things such as a text file, image file, video file, audio file etc.

- We can see that there are two places we can do this. One way is to write the raw files into primary/secondary storage. Another way is to write the cache files into the primary/secondary storage.
- **There is also difference between storing raw data and the cache data, the raw data once stored in memory by user has to be explicitly deleted by the user explicitly otherwise it would exist till then.** Cache data stored in memory is not a permanent data because the system automatically deletes it if it feels there is shortage of memory.



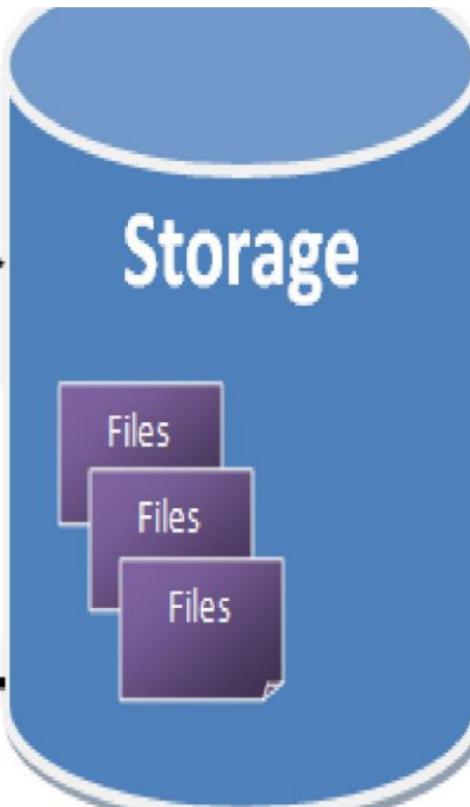
INTERNAL STORAGE

- Consider a user in an application has stored data in internal storage, then only that user of that application has access to that data on the mobile and that data is automatically deleted when the user uninstalls the application. Speaking of which **internal memory is private.**
- **The apps internal storage directory is stored using the name package name in a special place in the android filesystem.**
- Other apps or users of current app have no access to the file set by a particular user and a particular app unless it is explicitly made available to the user for readable/writable access.



Data is stored in the form of **Linear** data and **binary** data.

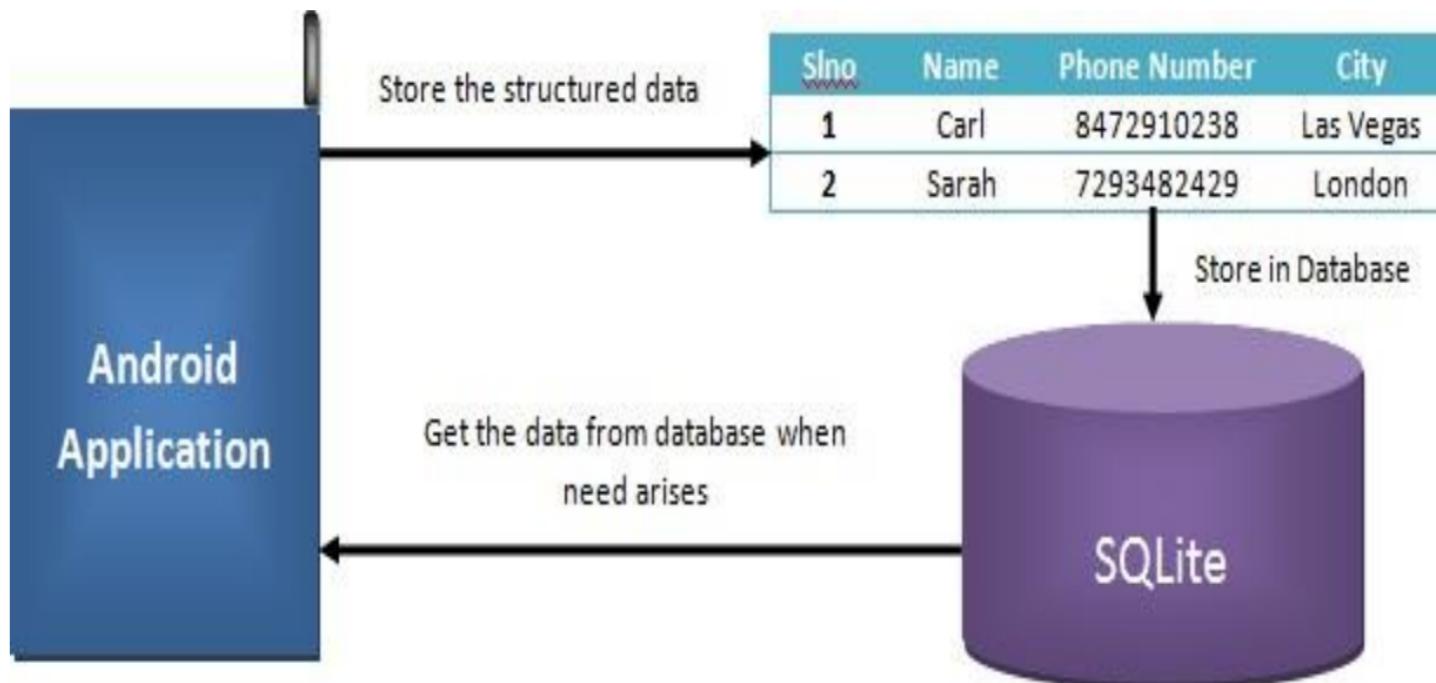
Data from the storage (binary data) is read from the application and converted into characters and displayed to the user so that the user understands it.



SharedPreferences vs Internal/External storage

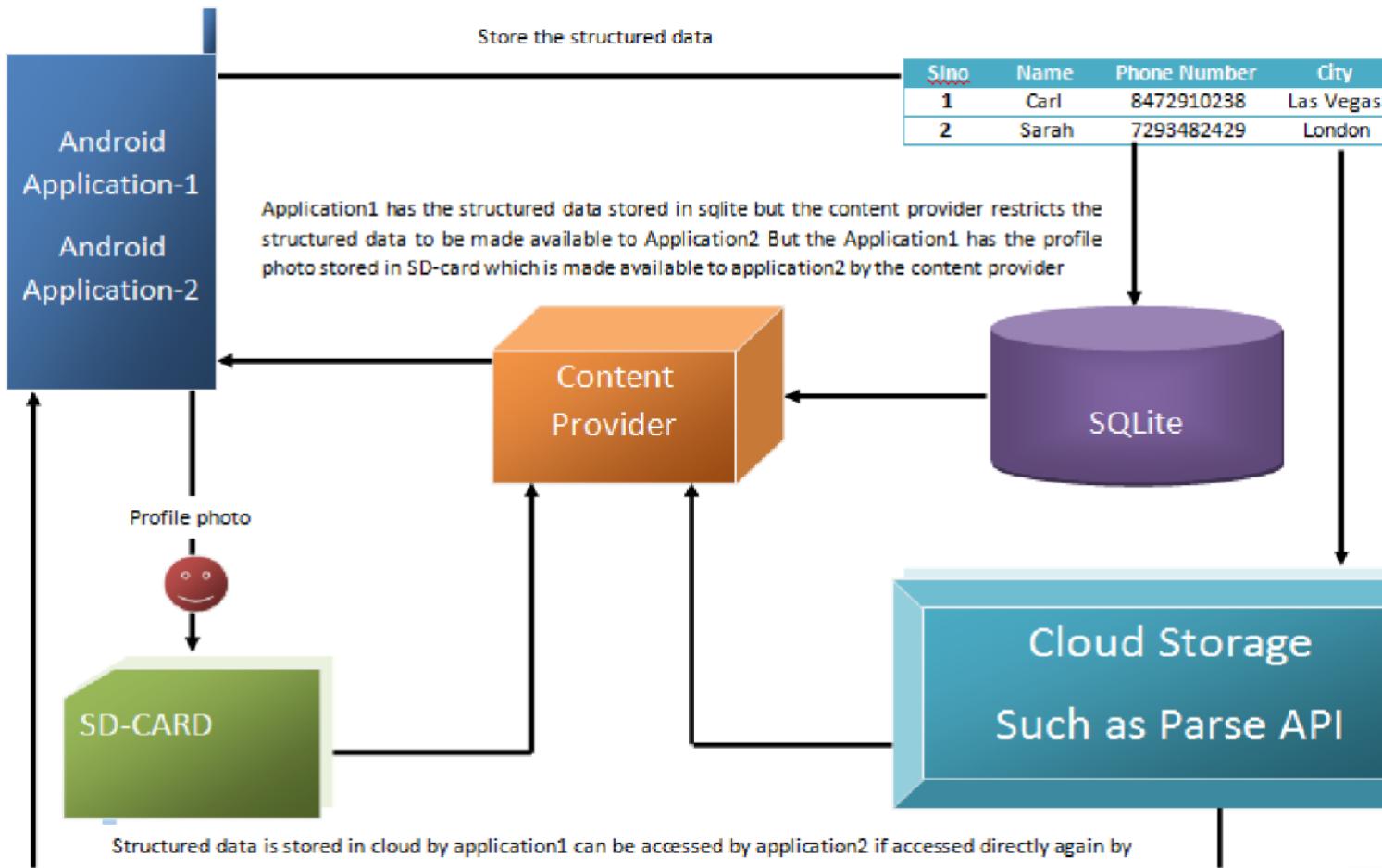
- **SharedPreferences**
- Storing data internally in XML files.
- More efficient due to less read/write overhead.
- Only primitive types and at most, arrays of strings can be stored
- **Internal/External storage**
- Storing data and files to the phone's external **Secure Digital (SD)** card.
- **Less Efficient**
- Store complex data, media, images

SQLite



- **Sqlite is used to store more structured data locally in a mobile where the android app is running.**
Structured data involves as of which shown in the figure like a student's information in the form of rows and columns.
- **Sqlite offers similar functionality like Mysql and oracle but with limited functional features.**
Some of the things involve performing query operations on tables. There are features though like creating views but also some features are not available like stored procedure.
- **Sqlite is very helpful in storing complex and large data** which can be downloaded once and can be used again and again until the application is running. When the application is closed the sqlite database is also destroyed.

PUTTING ALL THE PIECES TOGETHER



SharedPreferences is mainly for application-specific settings that you can access via your Settings menu - like application settings. It's a good idea to keep everything simple here - mostly boolean flags, short strings, or integers. SharedPreferences data persist on device reboot, and are removed along with app uninstallation. Data is saved as a key-value pair.

Internal Storage is mostly used for larger non-persistent data storage. You utilize internal storage if you want to *process* an image, a short video clip, a large text file, etc. But you don't store the processed data in the internal storage - its function is more like a CPU's RAM. The amount of available internal storage for your application depends on the device, but it's always a good idea to keep anything under 1MB. Data is referenced via its file path.

External Storage does not only refer to the SDCard storage, but for higher-end phones, this can mean internal mountable storage (like in the Galaxy Nexus or S2). This is where you store the large video files, the high-resolution images, and the 20-megabyte text file you want to parse in your application. This is also a place to store data that you want shared across devices if you swap sd cards. Data is also referenced via its file path.

Internal vs. External storage

- **Internal Storage**

- Less memory available [many MBs , few GBs]
- Access speed is more or less the same
- Only your app can access internal storage data and its deleted when the app is uninstalled

- **External storage**

- More memory available [many GBs]
- Anyone can read the external files and they are not destroyed unless they are located inside the directory returned by `getExternalFilesDir()`

SQLite Databases is where you'd store pretty much anything you want in a regular database - with the advantage of organizing things into tables, rows, and columns. It works best with things that you want displayed in the UI as lists - the best example would be the great invention known as the CursorAdapter. Data stored here also persist on device reboot, and are removed with app uninstallation. You can also share data across applications with sqlite db if you hook it up to a ContentProvider. Data is accessed using a Cursor, where you can call methods as if you're executing sql statements.

Network Connection is not really a data storage technique, but can be a way of persisting data for a specific user provided the device is connected to the internet, using some sort of authentication. You have to balance out between downloading data every time the app needs it, or having a one-time data sync, which would ultimately lead to another of the storage options mentioned above.

PACKAGING AND DEPLOYMENT

EXPLAIN IN DETAIL ABOUT PACKAGING AND DEPLOYMENT (PART A, B, C)

PACKAGING AN ANDROID APPLICATION: THE .APK FILE

Android provides an application called apk builder for generating installable Android application files, which have the extension .apk. An .apk file is in ZIP file format, just like many other Java-oriented application formats, and contains the application manifest, compiled application classes, and application resources.

Android provides the utility aapt (stands for Android Asset Packaging Tool. This tool is part of the SDK (and build system) and allows you to view, create, and update Zip-compatible archives (zip, jar, apk). It can also compile resources into binary assets.) for packaging the files that make up an .apk file, but developers typically prefer to allow their development environment to use this utility to build their applications for them. Most users simply rely on their IDE to build their .apk.

Once a developer has created an .apk file, he can choose to make it available for installation onto a device in one of several ways:

- Using the .adb (Android Debug Bridge) is a versatile command-line tool that lets you communicate with a device (an emulator or a connected Android device). interface directory, or more commonly by using an IDE
- (integrated development environment)
- Using an SD card
- Making the file available on a web server
- Uploading the file to the Android Market, and then selecting Install

PLACING AN APPLICATION FOR DISTRIBUTION IN THE ANDROID MARKET

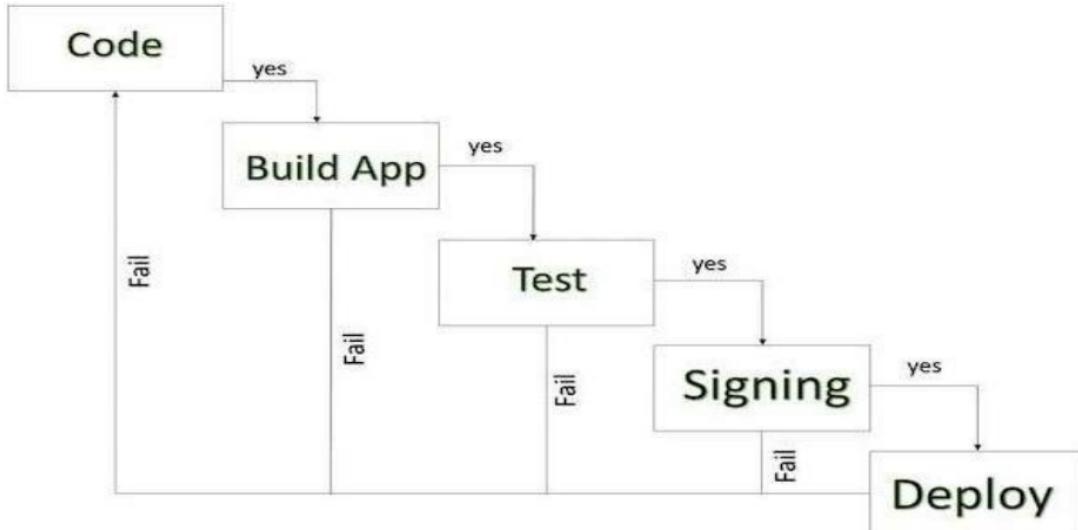
- Putting an application on the Android Market is remarkably easy. The only prerequisite is that you have a Google account such as a Gmail account.
- A \$25 credit card transaction and some information about yourself are all you need to start uploading applications to the Android Market.

API – APPLICATION PROGRAM INTERFACE

- *Application program interface (API)* is a set of routines, protocols, and tools for building software applications. An API specifies how software components should interact.
- Additionally, APIs are used when programming graphical user interface (GUI) components. A good API makes it easier to develop a program by providing all the building blocks.

DEPLOYMENT

Android application publishing is a process that makes your Android applications available to users. In fact, publishing is the last phase of the Android application development process.



Android development life cycle

Here is a simplified check list which will help you in launching your Android application –

Step	Activity
1	Regression Testing Before you publish your application, you need to make sure that its meeting the basic quality expectations for all Android apps, on all of the devices that you are targeting. So perform all the required testing on different devices including phone and tablets.
2	Application Rating When you will publish your application at Google Play, you will have to specify a content rating for your app, which informs Google Play users of its maturity level. Currently available ratings are (a) Everyone (b) Low maturity (c) Medium maturity (d) High maturity.

3

Targeted Regions Google Play lets you control what countries and territories where your application will be sold. Accordingly you must take care of setting up time zone, localization or any other specific requirement as per the targeted region.

4

Application Size Currently, the maximum size for an APK published on Google Play is 50 MB. If your app exceeds that size, or if you want to offer a secondary download, you can use APK Expansion Files, which Google Play will host for free on its server infrastructure and automatically handle the download to devices.

5

SDK and Screen Compatibility It is important to make sure that your app is designed to run properly on the Android platform versions and device screen sizes that you want to target.

6

Application Pricing Deciding whether your app will be free or paid is important because, on Google Play, free app's must remain free. If you want to sell your application then you will have to specify its price in different currencies.

7

Promotional Content It is a good marketing practice to supply a variety of high-quality graphic assets to showcase your app or brand. After you publish, these appear on your product details page, in store listings and search results, and elsewhere.

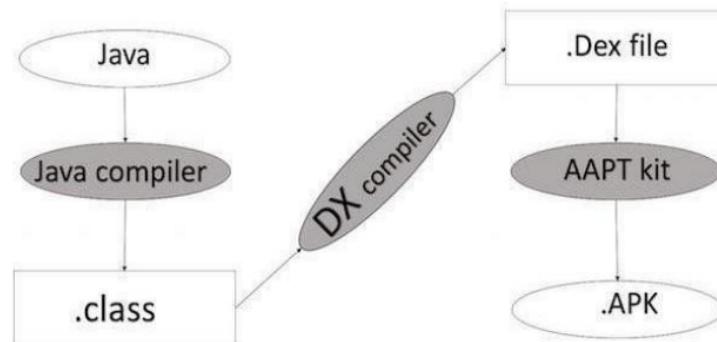
8

Build and Upload release-ready APK The release-ready APK is what you will upload to the Developer Console and distribute to users. You can check complete detail on how to create a release-ready version of your app: Preparing for Release.

9

Finalize Application Detail Google Play gives you a variety of ways to promote your app and engage with users on your product details page, from colourful graphics, screen shots, and videos to localized descriptions, release details, and links to your other apps. So you can decorate your application page and provide as much as clear crisp detail you can provide.

EXPORT ANDROID APPLICATION PROCESS

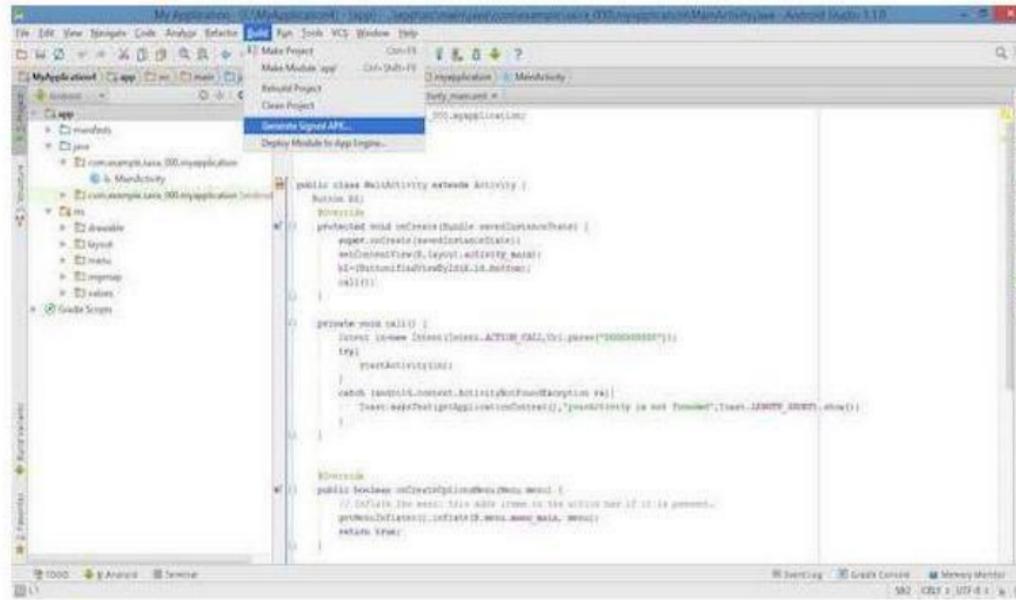


Apk development process

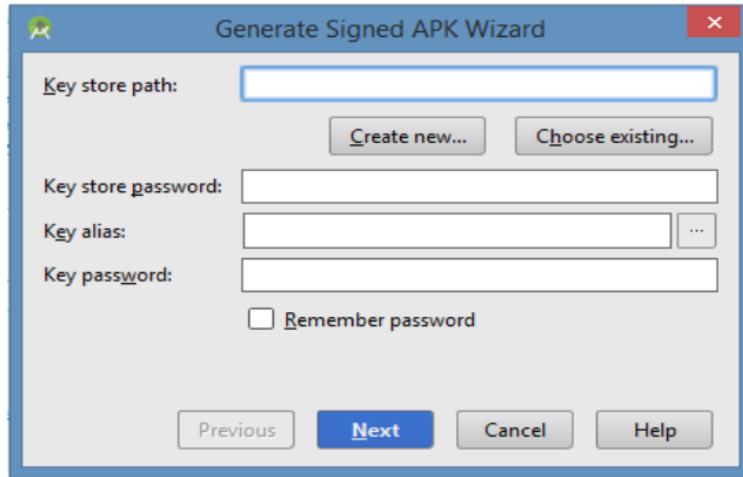
Before exporting the apps, you must some of tools

- **Dx tools**(Dalvik executable tools): It going to convert **.class file** to **.dex file**. it has useful for memory optimization and reduce the boot-up speed time
- **AAPT**(Android assistance packaging tool):it has useful to convert **.Dex file** to **.Apk**
- **APK**(Android packaging kit): The final stage of deployment process is called as **.apk**.

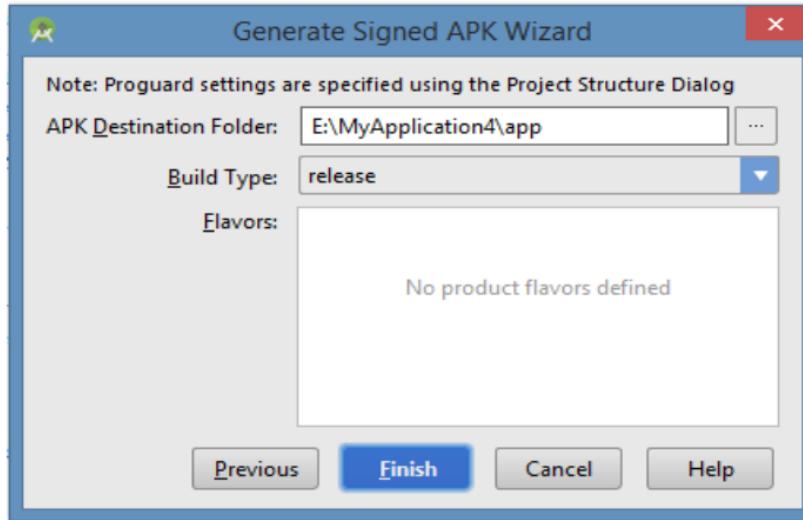
To export an application, just open that application project in Android studio and select **Build → Generate Signed APK** from your Android studio and follow the simple steps to export your application –



Next select, **Generate Signed APK** option as shown in the above screen shot and then click it so that you get following screen where you will choose **Create new keystore** to store your application.



Enter your key store path, key store password, key alias and key password to protect your application and click on **Next** button once again. It will display following screen to let you create an application –



To create a deployment configuration:

1. For development, choose **Run** and then **Debug Configurations**.

For production, choose **Run** and then **Run Configurations**

2. Navigate to MAF Application, and right-click and choose **New** to create a new configuration
3. Do the following:
 - o Accept the default name for the configuration or enter a new one.
 - o Choose the Assembly project.
 - o Choose the target from the list of those available and the device or emulator from the list of those available. If necessary, refresh the list.
 - o If necessary, select the keystore and key alias for the device you are deploying to.
4. If you want to deploy immediately, click **Run**. To save the configuration click **Apply**. Alternatively click **Close** and in the Save Changes dialog, click **Yes**.

DIFFERENCES BETWEEN RUN CONFIGURATIONS AND DEBUG CONFIGURATIONS

The Run Configurations and Debug Configurations are both invoked from the Run menu, and they differ in that Debug Configurations has an additional tab, the Debug tab, where you set debug options.

Debug—Select this option for development builds. Designating a debug build results in the inclusion of debugging symbols.

- **During development, you should choose debug in order to use the default keystore.**

At runtime, MAF indicates that an application has been deployed in the debug mode by overlaying a debugging symbol that is represented by an exclamation point within a red triangle.

- **Run—Select to compile the build ready for release, with libraries, and so on. release bits and libraries, and so on.**

SECURITY AND HACKING

WHAT IS THE SECURITY RISK RELATED TO MOBILE? (PART B, C)

Mobile security is the protection of smartphones, tablets, laptops and other portable computing devices, and the networks they connect to, from threats and vulnerabilities associated with wireless computing.

GENERAL SECURITY ISSUES

- Confidentiality

- **Integrity**
- **Availability**
- **Legitimate**
- **Accountability.**

WIRELESS SECURITY ISSUES

Wireless security issues are considered as the primary security issues of mobile computing. These are related to wireless networks. These issues occur when the hackers intercept the radio signals. Most wireless networks are dependent on other private networks, which are managed by others, so after these issues, the users have less control of security procedures. These security issues are:

DENIAL OF SERVICE (DOS) ATTACKS

- The denial of services or DOS attacks is one of the most common attacks of all kinds of networks and especially in a wireless network.
- It prevents users from using network services because the attacker sends a large amount of unnecessary data or connection requests to the communication server.
- It causes a slow network, and therefore the users cannot get benefitted from using its service.

TRAFFIC ANALYSIS

- Traffic analysis is used to identify and monitor communication between users.
- In this process, the service provider listens the traffic flowing in the wireless channel to access the private information of users affected by the attacker.

EAVESDROPPING

- It specifies that the attacker can log on to the wireless network and access sensitive data if the wireless network was not secure enough. This can also be done if the information is not encrypted.

SESSION INTERCEPTION AND MESSAGES MODIFICATION

- It specifies that the attacker can intercept the session and modify the transmitted data in this session. This scenario is called "man in the middle." It inserts the attacker's host between the sender and receiver host.

SPOOFING

- In this security issue, the attacker impersonates him as an authorized account of another user and tries to access the sensitive data and unauthorized services.

CAPTURED AND RETRANSMITTED MESSAGES

- In this security issue, the attacker can get some of the network services by getting unauthorized access. After capturing the message, he/she can reply to it with some modifications to the same destination or another.

DEVICE SECURITY ISSUES

Following is a list of some mobile computing security issues we face using mobile devices:

PUSH ATTACKS

In the push attack, the attacker creates a malicious code at the user's mobile device by hacking it and then he/she may spread it to affect other elements of the network.

PULL ATTACKS

The pull attack is a type of attack where the attacker controls the device and handles it in his/her way. He can decide which emails they want to receive. In this attack, the user can decide about the obtained data by the device itself.

FORCED DE-AUTHENTICATION

In this security issue, the attackers convince the mobile end-point or the mobile user to drop its connection and re-connection to get a new signal. Within this process, they insert their device between the mobile device and the network and steal the information or do the fraud.

MULTI-PROTOCOL COMMUNICATION

The multi-protocol communication provides the ability of many mobile devices to operate using

multiple protocols. For example, A cellular provider's network protocol. Most of the protocols have some security loopholes, which help the attacker to exploit this weakness and access to the device.

MOBILITY

This security issue may occur because of the mobility of the users and the mobile devices. You may face these security threats due to a user's location, so you must replicate the user profiles at different locations to allow roaming via different places without any concern regarding access to personal and sensitive data in any place and at any time. This repetition of sensitive data on different sites can increase seethe chances of security threats.

DISCONNECTIONS

These types of security issues occur when mobile devices go to different places. It occurs in the form of frequent disconnections caused by external parties resulting in the handoff.

HACKING

WHAT IS PHONE HACKING?

Phone hacking involves any method where someone forces access into your phone or its communications. This can range from advanced security breaches to simply listening in on unsecured internet connections. It can also involve physical theft of your phone and forcibly

hacking into it via methods like brute force. Phone hacking can happen to all kinds of phones, including Androids and iPhones

HOW TO KNOW IF SOMEONE IS HACKING YOUR PHONE

One or more of these could be a red flag that someone has breached your phone:

Your phone loses charge quickly. Malware and fraudulent apps sometimes use malicious code that tends to drain a lot of power.

Your phone runs abnormally slowly. A breached phone might be giving all its processing power over to the hacker's shady applications. This can cause your phone to slow to a crawl. Unexpected freezing, crashes, and unexpected restarts can sometimes be symptoms.

You notice strange activity on your other online accounts. When a hacker gets into your phone, they will try to steal access to your valuable accounts. Check your social media and email for password reset prompts, unusual login locations or new account signup verifications.

You notice unfamiliar calls or texts in your logs. Hackers may be tapping your phone with an SMS trojan. Alternatively, they could be impersonating you to steal personal info from your loved ones. Keep an eye out, since either method leaves breadcrumbs like outgoing messages.

ANDROID HACKING TOOLS / ANDROID HACKING APPS

In addition to manual coding, there are many applications built around hacking Android systems. These range from apps targeted at end users who want to extend their Android device's battery life or customize other parts of its operating system to deep system hacks used by more sophisticated hackers and attackers.

Here are a few of the most popular:

- **Apktool** – This tool is used for reverse engineering third party, closed, binary Android applications.
- **Dex2jar** – This widely available tool works with Android .dex and Java .class files, enabling the conversion of one binary format to another.
- **JD-GUI** – This is a graphic utility tool that stands alone and displays Java sources from .class files.

THE THREE BIGGEST THREATS TO ANDROID DEVICES

Threat One: Data in Transit

Mobile devices, including those running Android as an operating system, are susceptible to man-

in-the-middle attacks and various exploits that hack into unsecured communications over public Wi-Fi networks and other wireless communication systems.

Threat Two: Untrustworthy App Stores

Untrustworthy app stores can cause headaches due to lack of security protocols. Ensure that your app store of choice for Android applications takes adequate security precautions and has a strong security review program in place.

Threat Three: SMS Trojans

Malicious apps can sometimes include SMS trojans, which come in the form of compromised applications. This type of app accesses a mobile device's calling or text message capabilities, allowing them to do things like send text messages with malicious links to everyone in a user's address book. These links can then be used by attackers to distribute computer worms and other malicious messages to fee-based services, incurring fees on behalf of the user and profiting scammers.

THREE WAYS TO PROTECT YOUR ANDROID DEVICES

Use TLS Encryption

OWASP shows that insufficient encryption is a big problem for many types of applications. By using Transport Layer Security (TLS), you can encrypt internet traffic of all types for securely generating and exchanging session keys. This protects data against most man-in-the-middle and network spying attacks.

Test Third-Party App Security

The best way to avoid malicious apps is to only use apps from the official Google Play store. Google Play uses significantly better security checks than third-party sites, some of which may contain hundreds of thousands of malicious apps. If you absolutely need to download an app from a third-party store, check its permissions before installing, and be on the lookout for apps which that for your identity or the ability to send messages to your contacts when they don't need to.

Use Caution When Using SMS Payments

Set your Android phone to limit the ability of apps to automatically spend your money. Apps that ask for payment via SMS are a red flag and should be avoided if at all possible.

REFERENCE

- Reto Meier, Professional Android 4 Application Development:, Wiley, 1st edition,2012
- Zigmund Mednieks, Laird Dominm G. Blake Meike, Masumi Nakamura, -Programming Android, O'Reilly , 2nd edition,2012
- Alasdair Allan, iPhone Programming, O'Reilly , 1st edition,2010.
- Programming Mobile Devices An Introduction For Practitioners, Tommi Mikkonen, John Wiley & Sons.s

*******UNIT IV COMPLETED*******

VASAVI VIDYA TRUST GROUP OF INSTITUTIONS, SALEM- 103

CLASS: I MCA

SUBJECT CODE: MC5209

SUBJECT: MOBILE APPLICATION DEVELOPMENT

UNIT: V

UNIT V

APPLICATION DEVELOPMENT

**COMMUNICATION VIA THE WEB – NOTIFICATIONS AND ALARMS – GRAPHICS AND
MULTIMEDIA: LAYER ANIMATION, EVENT HANDLING AND**

GRAPHICS SERVICES – TELEPHONY – LOCATION BASED SERVICES

COMMUNICATION VIA THE WEB

WRITE ABOUT COMMUNICATION VIA THE WEB (PART B, C)

- Mobile technology is technology that goes where the user goes. It consists of portable two-way communications devices, computing devices and the networking technology that connects them.
- Currently, mobile technology is typified by internet-enabled devices like smartphones, tablets and watches. These are the latest in a progression that includes two-way pagers, notebook computers, mobile telephones (flip phones), GPS-navigation devices and more.
- The communications networks that connect these devices are loosely termed wireless technologies. They enable mobile devices to share voice, data and applications (mobile apps).

TYPES OF MOBILE NETWORKS

Cellular networks

- Radio networks using distributed cell towers that enable mobile devices (cell phones) to switch frequencies automatically and communicate without interruption across large

- geographic areas.
- The same basic switching capability enables cellular networks to accommodate many users across a limited number of radio frequencies.

4G networking

- The current cellular service standard for most wireless communication. It uses packet switching technology, which organizes data into parts or packets for transmission and reassembles the information at the destination.
- 4G —G1 for generation — is reported to be 10x faster than 3G — and 5G, faster still, is coming. 5G uses a set of aggregated frequency bands to unlock bandwidth and is approximately 20x faster than 4G.

WiFi

- Radio waves that connect devices to the internet through localized routers called hotspots. Short for wireless fidelity, WiFi networks are like cell towers for internet access, but they don't automatically pass service without establishing a WiFi connection.
- Most mobile devices allow for automatic switching between Wi-Fi and cellular networks depending upon availability and user preference.

Bluetooth

- A telecommunications industry specification for connecting devices over short distances

- using short-wavelength radio waves.
- Bluetooth enables users to quickly connect or pair devices such as headsets, speakers, phones and other devices.

MOBILE TECHNOLOGY CASE STUDIES

- ❖ Increase productivity
- ❖ Capitalize on new business models
- ❖ Create the ideal shopping scenario
- ❖ Enhance customer experiences

KEY CAPABILITIES OF EFFECTIVE MOBILE TECHNOLOGY

Scalability: Creating point solutions that don't scale across an enterprise can be costly in terms of development, management and maintenance. Apps need to be conceived holistically with

consideration for lines of business, processes and technical environments.

Integration: IDC has pointed out (PDF, 611KB) that applications offered on mobile phones and tablets have a separation between the mobile app and back-end business logic and data services. Being able to connect logic and data services to the app is critical, whether the logic and data are on premises, on the cloud or in hybrid configurations.

Reuse: Over 105 billion mobile apps were downloaded in 2018. Many are, or can be modified or combined, for business applications. Using existing apps accelerates time-to-value and improves cost efficiency by taking advantage of domain and industry expertise built into the app.

Cloud-based development: The cloud offers an efficient platform to develop, test and manage applications. Developers can use application programming interfaces (API) to connect apps to back-end data and focus on front-end functions. They can add authentication to bolster security, and access artificial intelligence (AI) and cognitive services.

Mobility management: As mobile technology is deployed, organizations look to enterprise mobility management (EMM) solutions to configure devices and apps; track device usage and

inventories; control and protect data; and support and troubleshoot issues.

BYOD: Bring your own device (BYOD) is an IT policy that allows employees to use personal devices to access data and systems. Effectively adopted, BYOD can improve productivity, increase employee satisfaction and save money.

Security: The mobile security battle is daunting in terms of volume and complexity. Artificial Intelligence (AI) is emerging as a key weapon to discern security anomalies in vast amounts of data. It can help surface and remediate malware incidents or recommend actions to meet regulatory requirements from a central dashboard.

Edge computing: One of the key advantages of 5G is that it can bring applications closer to their data sources or edge servers. Proximity to data at its source can deliver network benefits such as improved response times and better bandwidth availability. From a business perspective, edge computing offers the opportunity to perform more comprehensive data analysis and gain deeper insights faster.

NOTIFICATIONS AND ALARMS

WRITE BRIEF NOTE ON NOTIFICATION AND ALARMS (PART B, C)

DEFINITION

A notification is a message you can display to the user outside of your application's normal UI. When you tell the system to issue a notification, it first appears as an icon in the notification area. To see the details of the notification, the user opens the notification drawer. Both the notification area and the notification drawer are system-controlled areas that the user can view at any time.

Notifications are designed to tell users when apps that are inactive or running in the background have new information, including messages, upcoming events, or other relevant and timely data.

Notifications are displayed in three ways:

**Alerts or banners App
badges
Sounds or vibrations**

Android Notification provides short, timely information about the action happened in the application, even it is not running. The notification displays the icon, title and some amount of the content text.

SET ANDROID NOTIFICATION PROPERTIES

The properties of Android notification are set using **NotificationCompat.Builder** object. Some of the notification properties are mentioned below:

- **setSmallIcon()**: It sets the icon of notification.
- **setContentTitle()**: It is used to set the title of notification.
- **setContentText()**: It is used to set the text message.
- **setAutoCancel()**: It sets the cancelable property of notification.
- **setPriority()**: It sets the priority of notification.

Notifications are a way for your applications to alert users, without using an Activity.

Notifications are handled by the Notification Manager, and currently include the ability to:

- Create a new status bar icon.
- Display additional information (and launch an Intent) in the extended status bar window.
- Flash the lights/LEDs.
- Vibrate the phone.
- Sound audible alerts (ringtones, media store sounds).

Notifications are the preferred way for invisible application components (Broadcast Receivers, Services, and inactive Activities) to alert users that events have occurred that require attention.

- As a User Interface metaphor, Notifications are particularly well suited to mobile devices. It's likely that your users will have their phones with them at all times but quite unlikely that they will be paying attention to them, or your application, at any given time.
- Generally, users will have several applications open in the background, and they won't be paying attention to any of them. In this environment, it's important that your applications be able to alert users when specific events occur that require their attention.
- Notifications can be persisted through insistent repetition, or (more commonly) by using an icon on the status bar. Status bar icons can be updated regularly or expanded to show additional information using the expanded status bar window shown in Figure.

INTRODUCING THE NOTIFICATION MANAGER

The *Notification Manager* is a system Service used to handle Notifications. Get a reference to it using the *getSystemService* method, as shown in the snippet below:

```
String svcName = Context.NOTIFICATION_SERVICE;  
NotificationManager notificationManager;
```

```
notificationManager = (NotificationManager) getSystemService(svcName);
```

Using the Notification Manager, you can trigger new Notifications, modify existing ones, or remove those that are no longer needed or wanted.

TRIGGERING NOTIFICATIONS

To fire a Notification, pass it in to the `notify` method on the Notification Manager along with an integer reference ID, as shown in the following snippet:

```
int notificationRef = 1;  
notificationManager.notify(notificationRef, notification);
```

To update a Notification that's already been fired, re-trigger, passing the same reference ID. You can pass in either the same Notification object or an entirely new one. As long as the ID values are the same, the new Notification will be used to replace the status icon and extended status window details.

You also use the reference ID to cancel Notifications by calling the `cancel` method on the Notification Manager, as shown below:

```
notificationManager.cancel(notificationRef);
```

Canceling a Notification removes its status bar icon and clears it from the extended status window.

ADVANCED NOTIFICATION TECHNIQUES

MAKING SOUNDS

Using an audio alert to notify the user of a device event (like incoming calls) is a technique that predates the mobile, and has stood the test of time. Most native phone events from incoming calls to new messages and low battery are announced by an audible ringtone. Android lets you play any audio file on the phone as a Notification by assigning a location URI to the sound property, as shown in the snippet below:

```
notification.sound = ringURI;
```

VIBRATING THE PHONE

You can use the phone's vibration function to execute a vibration pattern specific to your Notification. Android lets you control the pattern of a vibration; you can use vibration to convey information as well as get the user's attention.

To set a vibration pattern, assign an array of longs to the Notification's vibrate property. Construct the array so that every alternate number is the length of time (in milliseconds) to vibrate or pause, respectively.

Before you can use vibration in your application, you need to be granted permission. Add a uses-

permission to your application to request access to the device vibration using the following code snippet:

```
<uses-permission android:name="android.permission.VIBRATE">
```

The following example shows how to modify a Notification to vibrate in a repeating pattern of 1 second on, 1 second off, for 5 seconds total.

```
long[] vibrate = new long[] { 1000, 1000, 1000, 1000, 1000 };  
notification.vibrate = vibrate;
```

FLASHING THE LIGHTS

Notifications also include properties to configure the color and flash frequency of the device's LED. The **ledARGB** property can be used to set the LED's color, while the **ledOffMS** and **ledOnMS** properties let you set the frequency and pattern of the flashing LED. You can turn the LED on by setting the **ledOnMS** property to 1 and the **ledOffMS** property to 0, or turn it off by setting both properties to 0. Once you have configured the LED settings, you must also add the **FLAG_SHOW_LIGHTS** flag to the Notification's flags property.

The following code snippet shows how to turn on the red device LED:

```
notification.ledARGB = Color.RED;  
notification.ledOffMS = 0;
```

```
notification.ledOnMS = 1;  
notification.flags = notification.flags / Notification.FLAG_SHOW_LIGHTS;
```

ALARMS

Alarms are an application independent way of firing Intents at predetermined times. Alarms are set outside the scope of your applications, so they can be used to trigger application events or actions even after your application has been closed. They can be particularly powerful in combination with Broadcast Receivers, allowing you to set Alarms that launch applications or perform actions without applications needing to be open and active until they're required.

Alarms in Android remain active while the device is in sleep mode and can optionally be set to wake the device; however, all Alarms are canceled whenever the device is rebooted. Alarm operations are handled through the ***AlarmManager***, a system Service accessed via ***getSystemService*** as shown below:

AlarmManager alarms

```
=(AlarmManager) getSystemService(Context.ALARM_SERVICE);
```

To create a new Alarm, use the **set method** and specify an alarm type, trigger time, and a Pending Intent to fire when the Alarm triggers. If the Alarm you set occurs in the past, it will be triggered immediately.

CHOOSE AN ALARM TYPE

One of the first considerations in using a repeating alarm is what its type should be. The following four alarm types are available:

- a. RTC_WAKEUP — Wakes the device from sleep to fire the Pending Intent at the clock time specified.
 - b. RTC — Fires the Pending Intent at the time specified but does not wake the device.
 - c. ELAPSED_REALTIME — Fires the Pending Intent based on the amount of time elapsed since the device was booted but does not wake the device. The elapsed time includes any period of time the device was asleep.
 - d. ELAPSED_REALTIME_WAKEUP — Wakes the device from sleep and fires the Pending Intent after a specified length of time has passed since device boot.
-
- **Elapsed real time uses the "time since system boot" as a reference,** and real time clock uses UTC (wall clock) time. This means that elapsed real time is suited to setting an alarm based on the passage of time (for example, an alarm that fires every 30 seconds) since it isn't affected by time zone/locale. The real time clock type is

better

- suited for alarms that are dependent on current locale.
- **Both types have a "wakeup" version, which says to wake up the device's CPU if the screen is off. This ensures that the alarm will fire at the scheduled time.** This is useful if your app has a time dependency for example, if it has a limited window to perform a particular operation. If you don't use the wakeup version of your alarm type, then all the repeating alarms will fire when your device is next awake.
- If you simply need your alarm to fire at a particular interval (for example, every half hour), use one of the elapsed real time types. In general, this is the better choice.
- If you need your alarm to fire at a particular time of day, then choose one of the clockbased real time clock types. This approach can have some drawbacks the app may not translate well to other locales, and if the user changes the device's time setting, it could cause unexpected behavior in your app. Using a real time clock alarm type also does not scale well, as discussed above.

SELECTING AND REPEATING ALARMS

- Repeating alarms work in the same way as the one-shot alarms but will trigger

repeatedly at the specified interval.

- Because alarms are set outside your Application lifecycle, they are perfect for scheduling regular updates or data lookups so that they don't require a Service to be constantly running in the background.
- To set a repeating alarm, use the `setRepeating` or `setInexactRepeating` method on the Alarm Manager. Both methods support an alarm type, an initial trigger time, and a Pending Intent to fire when the alarm triggers. Use `setRepeating` when you need fine-grained control over the exact interval of your repeating alarm. The interval value passed in to this method lets you specify an exact interval for your alarm, down to the millisecond.
- The `setInexactRepeating` method helps to reduce the battery drain associated with waking the device on a regular schedule to perform updates. At run time Android will synchronize multiple inexact repeating alarms and trigger them simultaneously.
- Rather than specifying an exact interval, the `setInexactRepeating` method accepts one of the following Alarm Manager constants:
 1. `INTERVAL_FIFTEEN_MINUTES`

2. INTERVAL_HALF_HOUR

GRAPHICS AND MULTIMEDIA

BRIEFLY EXPLAIN ABOUT GRAPHICS AND MULTIMEDIA(PART B, C)

- The only modern technology that can compete with mobile phones for ubiquity is the portable digital Media player. As a result, the multimedia capabilities of portable devices are a significant consideration for many consumers.
- Android's open platform- and provider-agnostic philosophy ensures that it offers a multimedia library capable of playing and recording a wide range of media formats, both locally and streamed.
- Android exposes this library to your applications, providing comprehensive multimedia functionality including recording and playback of audio, video, and still image media stored locally, within an application, or streamed over a data connection. At the time of print, Android supported the following multimedia formats:
 - a) JPEG
 - b) PNG
 - c) OGG

- d) Mpeg 4
- e) 3GP
- f) MP3
- g) Bitmap

PLAYING MEDIA RESOURCES

Multimedia playback in Android is handled by the MediaPlayer class. You can play back media stored as application resources, local files, or from a network URI. To play a media resource, create a new Media Player instance, and assign it a media source to play using the *setDataSource* method. Before you can start playback, you need to call prepare, as shown in the following code snippet:

```
String MEDIA_FILE_PATH =  
Settings.System.DEFAULT_RINGTONE_URI.toString();  
MediaPlayer mpFile = new MediaPlayer();  
try {  
    mpFile.setDataSource(MEDIA_FILE_PATH);  
    mpFile.prepare();  
    mpFile.start();  
}
```

```
        catch (IllegalArgumentException e) {}  
        catch (IllegalStateException e) {}  
        catch (IOException e) {}
```

Alternatively, the static create methods work as shortcuts, accepting media resources as a parameter and preparing them for playback, as shown in the following example, which plays back an application resource:

```
MediaPlayer mpRes = MediaPlayer.create(context, R.raw.my_sound);
```

Note that if you use a create method to generate your MediaPlayer object, prepare is called for you. Once a Media Player is prepared, call start as shown below to begin playback of the associated media resource.

```
mpRes.start();  
mpFile.start();
```

The Android Emulator simulates audio playback using the audio output of your development platform.

The Media Player includes stop, pause, and seek methods to control playback, as well as methods to find the duration, position, and image size of the associated media. To loop or repeat playback, use the **setLooping** method. When playing video resources, **getFrame** will take a screen grab of video media at the specified frame

and return a bitmap resource.

Once you've finished with the Media Player, be sure to call release to free the associated resources, as shown below:

```
mpRes.release();  
mpFile.release();
```

Since Android only supports a limited number of simultaneous Media Player objects, not releasing them can cause runtime exceptions.

Using the Camera

The popularity of digital cameras (particularly within phone handsets) has caused their prices to drop just as their size has shrunk dramatically. It's now becoming difficult to even find a mobile phone without a camera, and Android devices are unlikely to be exceptions.

To access the camera hardware, you need to add the CAMERA permission to your application manifest, as shown here:

```
<uses-permission android:name="android.permission.CAMERA"/>
```

This grants access to the Camera Service. The Camera class lets you adjust camera settings, take pictures, and manipulate streaming camera previews. To access the Camera Service, use the static open method on the Camera class. When your application has finished with the camera, remember to relinquish your hold on the

Service by calling release following the simple use pattern shown in the code snippet below:

```
Camera camera = Camera.open();
[ ... Do things with the camera ... ]
camera.release();
```

ANIMATIONS

Android supports three types of animation:

- **Property animations** — A tweened animation that can be used to potentially animate any property on the target object by applying incremental changes between two values. This can be used for anything from changing the color or opacity of a View to gradually fade it in or out, to changing a font size, or increasing a character's hit points.
- **View animations** — Tweened animations that can be applied to rotate, move, and stretch a View.
- **Frame animations** — Frame-by-frame —cell|| animations used to display a sequence of Drawable images.

Defining animations as external resources enables you to reuse the same sequence in

multiple places and provides you with the opportunity to present different animations based on device hardware or orientation.

PROPERTY ANIMATIONS

- ❖ Property animators were introduced in Android 3.0 (API level 11). It is a powerful framework that can be used to animate almost anything.
- ❖ Each property animation is stored in a separate XML file in the project's res/Animator folder. As with layouts and Drawable resources, the animation's filename is used as its resource identifier.
- ❖ You can use a property animator to animate almost any property on a target object. You can define animators that are tied to a specific property, or a generic value animator that can be allocated to any property and object.

VIEW ANIMATIONS

- Each view animation is stored in a separate XML file in the project's res/animfolder. As with layouts and Drawable resources, the animation's filename is used as its resource identifier.
- An animation can be defined for changes in alpha (fading), scale (scaling), translate

(movement), or rotate(rotation).

ANIMATION TYPE	ATTRIBUTES	VALID VALUES
Alpha	fromAlpha/toAlpha	Float from 0 to 1
Scale	fromXScale/toXScale	Float from 0 to 1
	fromYScale/toYScale	Float from 0 to 1
	pivotX/pivotY	String of the percentage of graphic width/height from 0% to 100%
Translate	fromX/toX	Float from 0 to 1
	fromY/toY	Float from 0 to 1
Rotate	fromDegrees/toDegrees	Float from 0 to 360
	pivotX/pivotY	String of the percentage of graphic width/height from 0% to 100%

The following list shows some of the settags available:

- duration— Duration of the full animation in milliseconds.
- startOffset— Millisecond delay before the animation starts.

- `fillBeforetrue`— Applies the animation transformation before it begins.
- `fillAftertrue` — Applies the animation transformation after it ends.
- `interpolator`— Sets how the speed of this effect varies over time.`:anim/interpolatorName`.

FRAME-BY-FRAME ANIMATIONS

- ❖ Frame-by-frame animations produce a sequence of Drawables, each of which is displayed for a specified duration.
- ❖ Because frame-by-frame animations represent animated Drawables, they are stored in the `res/ drawable` folder and use their filenames (without the `.xml` extension) as their resource Ids.
- ❖ The following XML snippet shows a simple animation that cycles through a series of bitmap resources, displaying each one for half a second. To use this snippet, you need to create new imageresources `android1`through `android3`:

```
<animation-list      xmlns:android="http://schemas.android.com/apk/res/android"  
    android:oneshot="false">  
    <item  android:drawable="@drawable/android1"  android:duration="500" />  
    <item  android:drawable="@drawable/android2"  android:duration="500" />
```

```
<item android:drawable="@drawable/android3" android:duration="500" />
</animation-list>
```

To play the animation, start by assigning the resource to a host View before getting a reference to the Animation Drawable object and starting it:

```
ImageView androidIV = (ImageView)findViewById(R.id.iv_android);
androidIV.setBackgroundResource(R.drawable.android_anim);
```

```
AnimationDrawable androidAnimation
= (AnimationDrawable)
    androidIV.getBackground();
    androidAnimation.start();
```

- ❖ Typically, this is done in two steps; assigning the resource to the background should be done within the `onCreate` handler.
- ❖ Within this handler the animation is not fully attached to the window, so the animations can't be started; instead, this is usually done as a result of user action (such as a button press) or within the `onWindowFocusChanged` handler.

EVENT HANDLING AND GRAPHICS SERVICES

DESCRIBE ABOUT EVENT HANDLING (PART B)

EXPLAIN THE GRAPHICS SERVICES ASSOCIATED WITH EVENTS(PART B, C)

Events are a useful way to collect data about a user's interaction with interactive components of Applications. Like button presses or screen touch etc. The Android framework maintains an event queue as first-in, first-out (FIFO) basis. You can capture these events in your program and take appropriate action as per requirements.

There are following three concepts related to Android Event Management –

- **Event Listeners** – An event listener is an interface in the View class that contains a single callback method. These methods will be called by the Android framework when the View to which the listener has been registered is triggered by user interaction with the item in the UI.
- **Event Listeners Registration** – Event Registration is the process by which an Event Handler gets registered with an Event Listener so that the handler is called when the Event Listener fires the event.
- **Event Handlers** – When an event happens and we have registered an event listener for the event, the event listener calls the Event Handlers, which is the method that actually handles the event.

Event Listeners & Event Handlers

Event Handler	Event Listener & Description
onClick()	<p>OnClickListener()</p> <p>This is called when the user either clicks or touches or focuses upon any widget like button, text, image etc. You will use onClick() event handler to handle such event.</p>
onLongClick()	<p>OnLongClickListener()</p> <p>This is called when the user either clicks or touches or focuses upon any widget like button, text, image etc. for one or more seconds. You will use onLongClick() event handler to handle such event.</p>
onFocusChange()	<p>OnFocusChangeListener()</p> <p>This is called when the widget loses its focus ie. user goes away from the view item. You will use onFocusChange() event handler to handle such event.</p>

onKey()	<p>OnFocusChangeListener()</p> <p>This is called when the user is focused on the item and presses or releases a hardware key on the device. You will use onKey() event handler to handle such event.</p>
onTouch()	<p>OnTouchListener()</p> <p>This is called when the user presses the key, releases the key, or any movement gesture on the screen. You will use onTouch() event handler to handle such event.</p>
onMenuItemClick()	<p>OnMenuItemClickListener()</p> <p>This is called when the user selects a menu item. You will use onMenuItemClick() event handler to handle such event.</p>
onCreateContextMenu()	<p>onCreateMenuItemListener()</p> <p>This is called when the context menu is being built(as the result of a sustained "long click")</p>

GRAPHICS SERVICES

CGPOINT

A `CGPoint` is the simplest Core Graphics structure, and contains two floating-point values corresponding to horizontal (X) and vertical (Y) coordinates on a display. To create a `CGPoint`, use the `CGPointMake` method:

```
CGPoint point = CGPointMake (320.0, 480.0);
```

Whenever an event is received, the object communicates with Graphics Services to get the specifics of the event. The Graphics Services framework provides many different decoding functions to extract the event's details.

EVENT LOCATION

For one-fingered events, the `GSEventGetLocationInWindow` function returns a `CGPoint` structure containing the X, Y coordinates where the event occurred. These coordinates are generally offset to the position of the window that received the event. For example, if a window located at the bottom half of the screen, whose origin was `0x240`, received an event at `0x0`, this means the event actually took place at `0x240` on the screen, which is where the window's `0x0` origin is located.

EVENT TYPE

The event type identifies whether a single finger or gesture was used, and whether the event involved a finger being placed down or raised from the screen.

Most events can be easily identified through the method that was notified. For example, if a finger is pressed down, this notifies the `mouseDown` method, whereas a two-finger gesture results in the `gestureStarted` method being notified:

```
unsigned int eventType = GSEventGetType(event);
```

- ❖ For a one-fingered event, the event type would begin as a single finger down, followed by all fingers up (when the user released).
- ❖ A two-fingered gesture is a little more complex. It begins life with a single finger down, and then changes to a two-finger gesture. If the user lifts one finger, the event type then changes to a one-finger-up event, followed by an all-fingers-up event when the user removes the second finger.

The following events are tied to event type values:

Type	Description
1	One finger down, including first finger of a

	gesture
2	All fingers up
5	One finger up in a two-fingered gesture
6	Two-finger gesture

EVENT CHORDING (MULTIPLE-FINGER EVENTS)

- ❖ When more than one note is played on a piano, it's considered a chord. The same philosophy is used in processing gestures. A single finger represents one note being played on the screen, whereas two are considered a chord.
- ❖ The `GSEventIsChordingHandEvent` method can be used to determine how many fingers are down on the screen at the time of the event:

```
int eventChording = GSEventIsChordingHandEvent(event);
```

MOUSE EVENTS

Mouse events are considered to be any touch screen event where a single finger is used. All classes derived from the `UIResponder` class inherit mouse events, and some classes override these to form new types of events, such as row selection within a table or changing the position of switches and sliders in a control.

- Mouse down
- Mouse up
- Mouse dragged

MOUSEENTERED, MOUSEEXITED, MOUSEMOVED

- These methods notify the application when the mouse is scrolled into, out of, and within an object's frame in the absence of any click event.
- The iPhone doesn't have a real mouse, so when you tap with your finger it's treated as a mouse down. These methods are relatively useless for the iPhone, but future mobile devices from Apple might make use of a real mouse or trackball.

GESTURE EVENTS

- When the user switches from a one-fingered tap to using two fingers, it's considered the beginning of a gesture.
- This causes gesture events to be created, which can be intercepted by overriding the appropriate methods.

- Gestures are provided in the `UIView` base class, and only objects that inherit from it support them.
- The order of events is this: as a gesture is started, the `gestureStarted` method is invoked. Should the user change his finger position, the `gestureChanged` method gets called to notify the object of each new gesture position. When the user has ended his gesture, `gestureEnded` is finally called.

gestureStarted

The `gestureStarted` method is notified when the user makes screen contact with two fingers or transitions from using one finger to two. This method is the two-fingered version of `mouseDown`. The inner and outer coordinates correspond to the first point of contact on the screen. The event locations in the `CGPoint` structures returned represent the coordinates at which each finger was pressed:

```
- (void)gestureStarted:(struct _GSEvent)event {  
    CGPoint leftFinger = GSEventGetInnerMostPathPosition(event);  
    CGPoint rightFinger = GSEventGetOuterMostPathPosition(event);
```

```
[ super gestureStarted: event ];  
  
/* Handle gesture started event */  
}
```

gestureEnded

The gestureEnded method is the two-fingered equivalent to mouseUp, and lets the application know that the user has removed at least one finger from the screen. If the user removes both fingers at the same time, the iPhone sends both events to the gestureEnded and mouseUp methods. If the user lifts fingers separately, the first finger to be removed causes a call to gestureEnded and the second causes a call to mouseUp.

The screen coordinates provided with the gestureEnded method identify the point, if any, that remains in a pressed-down state when just one finger was removed and the other finger is still down. When the second finger is removed, the mouseUp method will be notified, as the gesture will have been demoted to a mouse event when the first finger was removed:

```
- (void)gestureEnded:(struct _GSEvent)event {  
    CGPoint leftFinger = GSEventGetInnerMostPathPosition(event);  
    CGPoint rightFinger = GSEventGetOuterMostPathPosition(event);  
  
    [ super gestureEnded: event ];  
  
    /* Handle gesture ended event */  
}
```

gestureChanged

Whenever the user moves his fingers while in a two-fingered gesture, the gestureChanged method is called. This is the two-fingered equivalent of mouseDragged. When it occurs, the application should reevaluate the finger positions of the gesture and make the appropriate response. The event locations represent the new coordinates to which the fingers have been dragged:

```
- (void)gestureChanged:(struct _GSEvent)event {  
    CGPoint leftFinger = GSEventGetInnerMostPathPosition(event);
```

```
CGPoint rightFinger = GSEventGetOuterMostPathPosition(event);  
  
[ super gestureChanged: event ];  
  
/* Handle gesture change event */
```

TELEPHONY

EXPLAIN IN SHORT NOTES ABOUT TELEPHONY (PART B, C)

ANDROID TELEPHONY

The telephony APIs let your applications access the underlying telephone hardware, making it possible to create your own dialer or integrate call handling and phone state monitoring into your applications.

MAKING PHONE CALLS

The best practice is to use Intents to launch a dialer application to initiate new phone calls. There are two Intent actions you can use to dial a number.

Intent:

- Intent.ACTION_CALL** Automatically initiates the call, displaying the in-

call application. You should only use this action if you are replacing the native dialer. Your application must have the CALL_PHONE permission granted to broadcast this action.

I Intent.ACTION_DIAL Rather than dial the number immediately, this action starts a dialer application, passing in the specified number but allowing the dialer application to manage the call initialization (the default dialer asks the user to explicitly initiate the call).
This action

doesn't require any permissions and is the standard way applications should initiate calls.

The following skeleton code shows the basic technique for dialing a number:

```
Intent intent = new Intent(Intent.ACTION_DIAL, Uri.parse("tel:1234567"));  
startActivity(intent);
```

MONITORING PHONE CALLS

One of the most popular reasons for monitoring phone state changes is to detect, and react to, incoming and outgoing phone calls. Calls can be detected through changes in the phone's call state. Override the **onCallStateChanged**

Changed method in a Phone State Listener implementation, and register it as shown below to

receive notifications when the call state changes:

```
PhoneStateListener callStateListener = new PhoneStateListener() {  
    public void onCallStateChanged(int state, String incomingNumber) {  
        // TODO React to incoming call.  
    }  
};  
telephonyManager.listen(callStateListener,  
    PhoneStateListener.LISTEN_CALL_STATE);
```

The **onCallStateChanged** handler receives the phone number associated with incoming calls, and the state parameter represents the current call state as one of the following three values:

- a) TelephonyManager.CALL_STATE_IDLE When the phone is neither ringing nor in a call
- b) TelephonyManager.CALL_STATE_RINGING When the phone is ringing
- c) TelephonyManager.CALL_STATE_OFFHOOK If the phone is currently on a call

TRACKING CELL LOCATION CHANGES

We can get notifications whenever the current cell location changes by overriding onCellLocationChanged on a Phone State Listener

implementation. Before you can register to listen for cell location changes, you need to add the ACCESS_COARSE_LOCATION permission to your application manifest.

<uses-permission

android:name="android.permission.ACCESS_COARSE_LOCATION"/>

The onCellLocationChanged handler receives a CellLocation object that includes methods for extracting the **cell ID (getCID)** and the **current LAC (getLac)**. The following code snippet shows how to implement a Phone State Listener to monitor cell location changes, displaying a Toast that includes the new location's cell ID:

```
PhoneStateListener cellLocationListener = new PhoneStateListener()
{
    public void onCellLocationChanged(CellLocation location)
    {
        GsmCellLocation gsmLocation = (GsmCellLocation)location;
        Toast.makeText(getApplicationContext(),
        String.valueOf(gsmLocation.getCid()),
        Toast.LENGTH_LONG).show();
    }
};
```

```
telephonyManager.listen(cellLocationListener,  
PhoneStateListener.LISTEN_CELL_LOCATION);
```

TRACKING SERVICE CHANGES

The onServiceStateChanged handler tracks the service details for the device's cell service. Use the ServiceState parameter to find details of the current service state.

The getState method on the Service State object returns the current service state as one of:

- ❖ ServiceState.STATE_IN_SERVICE Normal phone service is available.
- ❖ ServiceState.STATE_EMERGENCY_ONLY Phone service is available but only for emergency calls.
- ❖ ServiceState.STATE_OUT_OF_SERVICE No cell phone service is currently available.
- ❖ ServiceState.STATE_POWER_OFF The phone radio is turned off (usually when airplane mode is enabled).

A series of getOperator* methods is available to retrieve details on the operator supplying the cell phone service, while getRoaming tells you if the device is currently using a roaming profile. The following example shows how to register for service state changes and displays a Toast showing the operator name of the current phone service:

```
PhoneStateListener serviceStateListener = new PhoneStateListener()
{
    public void onServiceStateChanged(ServiceState serviceState)
    {
        if (serviceState.getState() == ServiceState.STATE_IN_SERVICE)
        {
            String toastText = serviceState.getOperatorAlphaLong();
            Toast.makeText(getApplicationContext(), toastText, Toast.LENGTH_SHORT);
        }
    }
};

telephonyManager.listen(serviceStateListener,
    PhoneStateListener.LISTEN_SERVICE_STATE);
```

LOCATION BASED SERVICES

DISCUSS IN DETAIL ABOUT LOCATION BASED SERVICES (PART B, C)

USING LOCATION-BASED SERVICES

-Location-based services| is an umbrella term that describes the different technologies you can use to find a device's current location. The two main LBS elements are:

- ▶ **Location Manager** — Provides hooks to the location-based services.
- ▶ **Location Providers** — Each of these represents a different location-finding technology used to determine the device's current location.

Using the Location Manager, you can do the following:

- **Obtain your current location**
- **Follow movement**
- **Set proximity alerts for detecting movement into and out of a specified area**
- **Find available Location Providers**

Monitor the status of the GPS receiver Before you can use the location-based services, you need to add one or more uses-permission tags to your manifest.

The following snippet shows how to request the *fine* and *coarse* permissions in your application manifest:

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
```

SELECTING A LOCATION PROVIDER

Depending on the device, you can use several technologies to determine the current location. Each technology, available as a Location Provider, offers different capabilities — including differences in power consumption, accuracy, and the ability to determine altitude, speed, or heading information

FINDING LOCATION PROVIDERS

The `LocationManager` class includes static string constants that return the provider name for three Location Providers:

- `LocationManager.GPS_PROVIDER`
- `LocationManager.NETWORK_PROVIDER`
- `LocationManager.PASSIVE_PROVIDER`

If more than one Location Provider matches your Criteria, the one with the greatest accuracy is returned. If no Location Providers meet your requirements, the Criteria are loosened, in the following order, until a provider is found:

- Power use
- Accuracy of returned location
- Accuracy of bearing, speed, and altitude
- Availability of bearing, speed, and altitude

FINDING YOUR CURRENT LOCATION

One of the most powerful uses of location-based services is to find the physical allocation of the device. The accuracy of the returned location is dependent on the hardware available and the permissions requested by your application.

LOCATION PRIVACY

Privacy is an important consideration when your application uses the user's location particularly when it is regularly updating their current position. Ensure that your application uses the device location data in a way that respects the

user's privacy by:

- Only using and updating location when necessary for your application
- Notifying users of when you track their locations, and if and how that location information is used, transmitted, and stored
- Allowing users to disable location updates, and respecting the system settings for LBS preferences

FINDING THE LAST KNOWN LOCATION

We can find the last location fix obtained by a particular Location Provider using the getLastKnownLocation method, passing in the name of the Location Provider. The following example finds the last location fix taken by the GPS provider:

```
String provider = LocationManager.GPS_PROVIDER;  
Location location = locationManager.getLastKnownLocation(provider);
```

The Location object returned includes all the position information available from the provider that supplied it. This can include the time it was obtained, the accuracy of the location found, and its latitude, longitude, bearing, altitude, and speed. All these properties are available via

getmethods on the Location object.

BEST PRACTICE FOR LOCATION UPDATES

When using Location within your application, consider the following factors:

- ❖ **Battery life versus accuracy** — The more accurate the Location Provider, the greater its drainon the battery.
- ❖ **Startup time** — In a mobile environment the time taken to get an initial location can have a dramatic effect on the user experience — particularly if your app requires a location to be used. GPS, for example, can have a significant startup time, which you may need to mitigate.
- ❖ **Update rate** — The more frequent the update rate, the more dramatic the effect on batterylife. Slower updates can reduce battery drain at the price of less timely updates.
- ❖ **Provider availability** — Users can toggle the availability of providers, so your application needs to monitor changes in provider status to ensure the best alternative is used at all times

USING THE GEOCODER

- ❖ Geocoding enables you to translate between street addresses and longitude/latitude map coordinates. This can give you a recognizable context for the locations and coordinates used in location-based services and map-based Activities.
- ❖ The Geocoder classes are included as part of the Google Maps library, so to use them you need to import it into your application by adding a uses-library node within the application node as shown here:

```
<uses-library android:name="com.google.android.maps"/>
```

As the geocoding lookups are done on the server, your application also requires the Internet uses-permission in your manifest:

```
<uses-permission android:name="android.permission.INTERNET"/>
```

The Geocoder class provides access to two geocoding functions:

1. **Forward geocoding** — Finds the latitude and longitude of an address
2. **Reverse geocoding** — Finds the street address for a given latitude and longitude

REFERENCE

1. Reto Meier, Professional Android 4 Application Development:, Wiley, 1st edition,2012
2. ZigurdMednieks, Laird Dominm G. Blake Meike, Masumi Nakamura, -Programming Android, O'Reilly , 2nd edition,2012
3. Alasdair allan, iPhone Programming, O'Reilly , 1st edition,2010.
4. Android in action 2nd Edition, W. Frank Ableson Robi Sen Chris King

*******UNIT V COMPLETED*******