• प्रियम तायल • पवन कुमार गुप्ता
• प्रद्युमन कुमार

एंड्रॉयड अनुप्रयोगों का विकास

DEVELOPMENT OF ANDROID APPLICATIONS

प्राविधिक शिक्षा परिषद् उत्तर प्रदेश द्वारा स्वीकृत, नवीन "NSQF" आधारित संशोधित पाठ्यक्रमानुसार

एंड्रॉयड अनुप्रयोगों का विकास

(DEVELOPMENT OF ANDROID APPLICATIONS)

तृतीय वर्ष, छठे सेमेस्टर (VI Sem), इंफॉर्मेशन टेक्नोलॉजी, कम्प्यूटर साइंस एवं इंजीनियरिंग ब्रांच के विद्यार्थियों के लिए

लेखकगण :

प्रियम तायल
असिस्टेंट प्रोफेसर
गेटवे इंस्टीट्यूट ऑफ इंजीनियरिंग एण्ड टेक्नोलॉजी
सोनीपत (हरियाणा)

पवन कुमार गुप्ता
विभागाध्यक्ष, सूचना प्रौद्योगिकी (आई०टी०)
चौ० मुख्तार सिंह राजकीय बालिका पॉलीटेक्निक,
दौराला, मेरठ (उत्तर प्रदेश)

प्रद्युम्न कुमार
भूतपूर्व आई०टी० विश्लेषक, टी०सी०एस० लि०
प्रवक्ता, एम०एम०आइं०टी०
संत कबीर नगर (उत्तर प्रदेश)

2021

प्रकाशकः

एशियन पब्लिशर्स, मुज़फ़्फ़रनगर ®

46/20, कम्बल वाला बाग, नई मण्डी, मुज़फ़्फ़रनगर-251 001 (उ०प्र०)

SYLLABUS

L	T	P
6	–	6

RATIONALE

Knowing the details of Mobile and their working principle are need of the every common man. Mobile Application development is the very hot business domain. Majority of the corporate have a separate division for the development of mobile applications. It is essential that diploma students must know the way to apply advanced data communicating methods and networking protocols for wireless and mobile devices. Hence this subject.

LEARNING OUTCOMES

After undergoing this subject, the students will be able to :

- Illustrate the usage of different components of Android OS in detail.
- Develop a mobile application using different components of Android.
- Choose appropriate controls to design the GUI to meet desired needs.
- Consume JSON data and call web services from Android mobile app.
- Write a program in Android to store data in databases.
- Develop Mobile applications using Android.

SUGGESTED DISTRIBUTION OF MARKS

S.No.	Topic Name	Time Alloted (Periods)	Marks Alloted (%)
1.	Introduction to Android	10	14
2.	Environment Setup and Basic Project Structure	08	08
3.	Android Fundamentals and User Interface Design	12	14
4.	Menus and Preferences	10	12
5.	Advanced UI Components	12	14
6.	Threads in Android	12	14
7.	Notifications and Services	10	12
8.	Storage and Content Provider	10	12
	Total	**84**	**100**

DETAILED CONTENTS

1. Introduction to ANDROID (10 periods)

What is Android? Dalvik Virtual Machine & .apk file extension, Fundamentals : Basic Building blocks—Activities, Services, Broadcast Receivers & Content providers, UI Components—Views

& notifications, Components for communication -Intents & Intent Filters, Android API levels (versions & version names)

2. Environment Setup and Basic Project Structure **(08 periods)**

Setting up development environment Android, Manifest.xml, Gradle, Uses-permission & uses-sdk, Resources & R.java, Assets, Layouts & Drawable Resources, First sample Application, Launching emulator, Editing emulator settings, Emulator shortcuts, Logcat usage, Introduction to DDMS, Hello World App, Creating your first project The manifest file Layout resource, Running your app on Emulator, Debugging the Android App.

3. Android Fundaments and User Interface Design **(12 periods)**

Activities and Activity lifecycle, Permission System

Basic UI Components : Text View, Button, Radio Button , Edit Text, Image View for image, Check Box, Progress Bar, Event Handling in Android.

Layouts : Liner Layout, Relative Layout, Frame Layout, Coordinate Layout, [dip, dp, sip, sp] versus px

Intents : Intents introduction and importance, Types of Intents (Explicit Intents, Implicit intents)

4. Menus and Preferences **(10 periods)**

Introduction to Menus, Types of Menus (Option menu, Context menu), Uses of Shared Preferences.

5. Advanced UI Components **(12 periods)**

Time and Date, List View, Grid View, Card View, recycler view Adaptors (Base Adaptor, Array Adaptor) & View Holder, Dialogs, Toast, Popup, Fragments, Material Design(Introduction , Navigation, Floating Button , Tool bar).

6. Threads in Android **(12 periods)**

Threads running on UI thread (run on UI Thread), Worker thread, Handlers & Runnable, AsynTask , calling web services and consuming JSON data from Web Services.

7. Notifications & Services **(10 periods)**

Broadcast Receivers (Introduction, different ways to register a broadcast receiver), Introduction to Notification, Overview & Types of services, implementing a Service, Service lifecycle

8. Storage and Content Provider **(10 periods)**

Supported Storage in Android (Internal memory, External memory, Shared Preferences and network), SQLite introduction, CRUD Operations in SQLite database (cursor, content values etc) , Basics of Content Provider

LIST OF PRACTICALS

1. Install the Android Studio and Setup the Development Environment
2. Write a program to demonstrate activity (Application Life Cycle)
3. Write a program to demonstrate different types of layouts
4. Write a program to implement simple calculator using text view, edit view, option button and button
5. Write a program to develop app having multiple activities and user should be able switch between the activities by using intents.
6. Write a program to demonstrate list view
7. Write a program to demonstrate photo gallery
8. Write a program to demonstrate Date picker and time picker
9. Develop an simple application with context menu and option menu.
10. Write a program to demonstrate the functionality of Shared Preferences.
11. Write a program to demonstrate a service
12. Write a program to demonstrate the application of intent class
13. Write a program to create a text file in a external memory
14. Write a program to store and fetch data from SQL life database.

CONTENTS

1. एंड्रॉयड का परिचय
(Introduction to Android) **1-30**

2. एनवायरनमेंट सेटअप और बेसिक प्रोजेक्ट स्ट्रक्चर
(Environment Setup and Basic Project Structure) **31-75**

3. एंड्रॉयड फंडामेंटल्स और यूजर इंटरफेस डिजाइन
(Android Fundamentals and User Interface Design) **76-114**

4. मेन्स और प्रैफरेंसेज
(Menus and Preferences) **115-130**

5. एडवांस्ड UI कंपोनेंट्स
(Advanced UI Components) **131-173**

6. एंड्रॉयड में थ्रेड
(Threads in Android) **174-209**

7. नोटिफिकेशन और सर्विसेज
(Notifications and Services) **210-242**

8. स्टोरेज और कंटेंट प्रोवाइडर
(Storage and Content Provider) **243-291**

- प्रक्टिकल
(Practicals) **292-369**

- प्रश्न–पत्र


REDMI NOTE 8
AI QUAD CAMERA

अंशुल अग्रवाल • एस० अग्रवाल

औद्योगिक प्रबन्धन एवं उद्यमियता विकास

INDUSTRIAL MANAGEMENT & ENTREPRENEURSHIP DEVELOPMENT

एशियन पब्लिशर्स, मुज़फ़्फ़रनगर

Chapter 1

एंड्रॉयड का परिचय
(INTRODUCTION TO ANDROID)

SYLLABUS

What is Android? Dalvik Virtual Machine & .apk file extension, Fundamentals: Basic Building blocks—Activities, Services, Broadcast Receivers & Content providers, UI Components—Views & notifications, Components for communication -Intents & Intent Filters, Android API levels (versions & version names)

§ 1.1 Android क्या है? (What is Android?)

चित्र : 1.1

Android ऑपरेटिंग सिस्टम स्मार्टफोन और टैबलेट कम्प्यूटर जैसे मोबाइल उपकरणों के लिए ओपनसोर्स और लिनक्स-आधारित ऑपरेटिंग सिस्टम है। Android का डेवलपमेंट Google और अन्य कंपनियों के नेतृत्व में Open हैंडसेट एलायंस द्वारा किया गया था।

Android मोबाइल उपकरणों के लिए एप्लीकेशन डेवलपमेंट के लिए एक एकीकृत दृष्टिकोण प्रदान करता है, जिसका अर्थ है कि डेवलपर्स को केवल Android के लिए को विकसित करने की आवश्यकता है, और उनके एप्लिकेशन्स को जो Android द्वारा संचालित विभिन्न उपकरणों पर चलाने में सक्षम है।

Software Development Kit (SDK) का पहला बीटा संस्करण Google द्वारा 2007 में जारी किया गया था, जहाँ पहला व्यावसायिक संस्करण, Android 1.0, सितंबर 2008 में जारी किया गया था।

27 जून 2012 को, Google I / O सम्मेलन में, Google ने अगले Android संस्करण, 4.1 जेली बीन की घोषणा की। जेली बीन एक वृद्धिशील अपडेट है, जिसका मुख्य उद्देश्य कार्य क्षमता और प्रदर्शन दोनों के मामले में यूजर इंटरफ़ेस में सुधार करना है।

§ 1.2 Android ही क्यों?

मोबाइल एप्लीकेशन डेवलपमेंट के लिए Android प्लेटफॉर्म चुनने के लिए बहुत सारे कारण हैं।

1. शून्य /बहुत कम डेवलपमेंट लागत (Zero/negligible development cost)

Android Android SDK, JDK, और Eclipse IDE आदि जैसे डेवलपमेंट उपकरण Android मोबाइल एप्लीकेशन डेवलपमेंट के लिए डाउनलोड करने के लिए नि:शुल्क हैं।

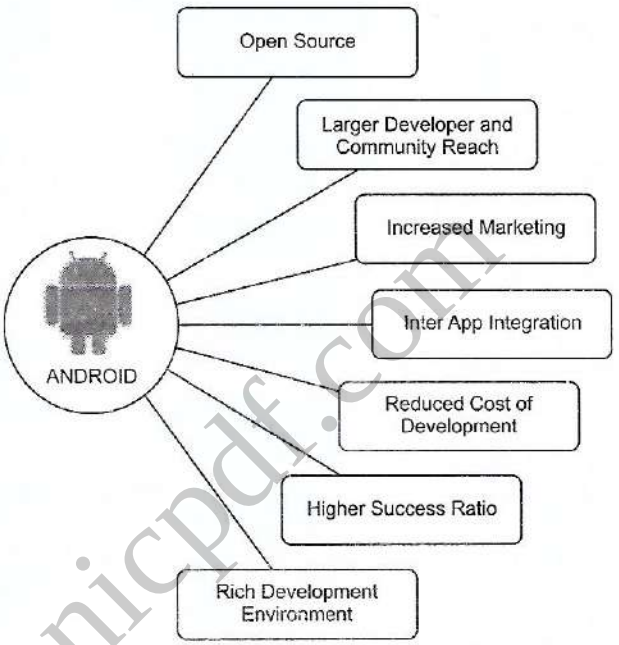
2. ओपन सोर्स (Open Source)

Android OS एक ओपन-सोर्स प्लेटफॉर्म है जो लिनक्सकर्नेल और कई ओपन-सोर्स लाइब्रेरी पर आधारित है। इस तरह डेवलपर्स जो Android डिवाइस पर मोबाइल चलते हैं उन उपकरणों के निर्माण के लिए आवश्यक प्लेटफॉर्म का विस्तार करने के लिए स्वतंत्र हैं।

3. **मल्टी-प्लेटफ़ॉर्म समर्थन**
(Multi-Platform Support)
बाजार में, Android OS द्वारा संचालित हार्डवेयर उपकरणों की एक विस्तृत श्रृंखला है, जिसमें कई अलग-अलग मोबाइल और टैबलेट शामिल हैं। Android मोबाइल एप्लीकेशन का डेवलपमेंट विंडोज, मैकओएस या लिनक्स पर भी हो सकता है।

4. **मल्टी कैरियर सपोर्ट**
(Multi-Carrier Support)
दुनिया भर में बड़ी संख्या में दूरसंचार वाहक जैसे एयरटेल, वोडाफोन, आइडिया सेल्युलर, BSNL आदि Android संचालित फोन का समर्थन कर रहे हैं।

5. **ओपन डिस्ट्रीब्यूशन मॉडल**
(Open Distribution Model)
Android Market place (Google Play store) में Android ऐप का कंटेंट या कार्यक्षमता पर बहुत कम प्रतिबंध है। इसलिए डेवलपर Google Play स्टोर के माध्यम से अपने ऐप को वितरित कर सकते हैं।

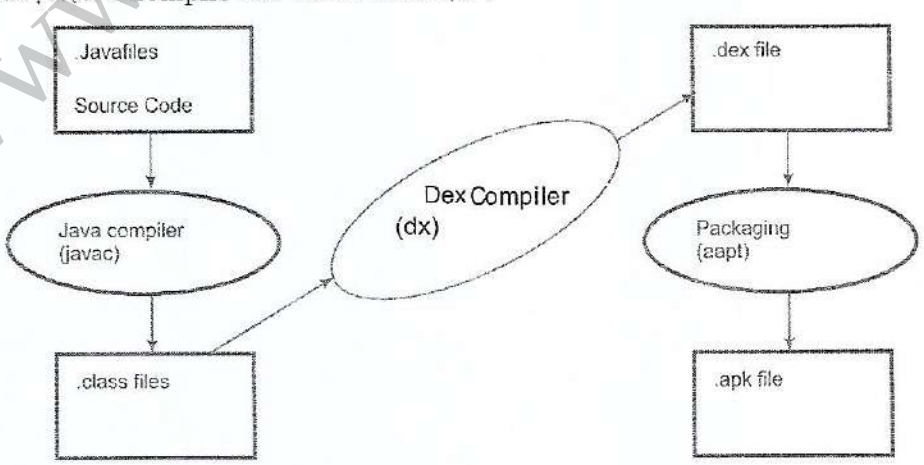


चित्र : 1.2 : Advantages of Android

§ 1.3 Dalvik वर्चुअल मशीन (Dalvik Virtual Machine)

Dalvik Virtual Machine (DVM) मोबाइल उपकरणों के लिए अनुकूलित Android वर्चुअल मशीन है। यह मेमोरी, बैटरी लाइफ और प्रदर्शन के लिए वर्चुअल मशीन को अनुकूलन करता है। Dalvik आइसलैंड में एक शहर का एक नाम है। Dalvik VM डैनबॉर्न स्टीन ने लिखा था। Dex कंपाइलर क्लास की फाइलों को .dex फाइल में परिवर्तित करता है, जो Dalvik VM पर चलती है। मल्टीपल क्लास फाइलें एक डेक्स फाइल में बदल जाती हैं।

चलो स्रोत फ़ाइल से compile और पैकेजिंग प्रोसेस देखें :



चित्र : 1.3 : Compiling and Packaging Process

Javac टूल java source फाइल को क्लास फाइल में संकलित करता है।

Dx टूल आप के एप्लीकेशन की सभी क्लास फाइल्स लेता है और एक .dex फाइल बनाता है। यह एक प्लेटफ़ॉर्म-विशिष्ट उपकरण है। **Android Assets Packaging Tool (aapt)** पैकेजिंग प्रोसेस को संभालता है।

लाभ

- DVM केवल Android ऑपरेटिंग सिस्टम का समर्थन करता है।
- DVM निष्पादन योग्य APK है।
- निष्पादन तेज है।
- Android 2.2 से SDK Dalvik में इसका अपना JIT (Just In Time) कंपाइलर है।
- DVM को इसलिए डिज़ाइन किया गया है ताकि कोई डिवाइस वर्चुअल मशीन के कई उदाहरणों को प्रभावी ढंग से चला सके।

नुकसान :

- DVM केवल Android ऑपरेटिंग सिस्टम का समर्थन करता है।
- DVM के लिए बहुत कम री-टूल्स उपलब्ध हैं।
- समान उच्च-स्तरीय कोड को लागू करने के लिए रजिस्टर मशीनों से अधिक निर्देशों की आवश्यकता होती है।
- Dex के कारण ऐप इंस्टॉलेशन में अधिक समय लगता है।
- अधिक आंतरिक (internal storage) भंडारण की आवश्यकता है।

§ 1.4 apk फाइल एक्सटेंशन

APK फ़ाइल एक्सटेंशन के साथ एक फाइल एक Android पैकेज फ़ाइल है जिसका उपयोग Google के Android ऑपरेटिंग सिस्टम पर एप्लीकेशन वितरित करने के लिए किया जाता है। APK फाइलें ZIP प्रारूप में सेव की जाती हैं और आमतौर पर सीधे Android डिवाइसों पर डाउनलोड की जाती हैं, आमतौर पर Google Play स्टोर के माध्यम से की जाती है।

§ 1.5 फंडामेंटल्स : बेसिक बिल्डिंग बलॉक्स (Fundamentals: Basic Building blocks)

एप्लीकेशन घटक एक Android एप्लीकेशन के आवश्यक बिल्डिंग ब्लॉक हैं। इन घटकों को एप्लीकेशन में नफ़ेस्ट फ़ाइल Android Manifest .xml द्वारा लूसेली कपल किया गया है, जो कि एप्लीकेशन के प्रत्येक घटक के बारे में बताते हैं और वे कैसे इंटरैक्ट करते हैं।

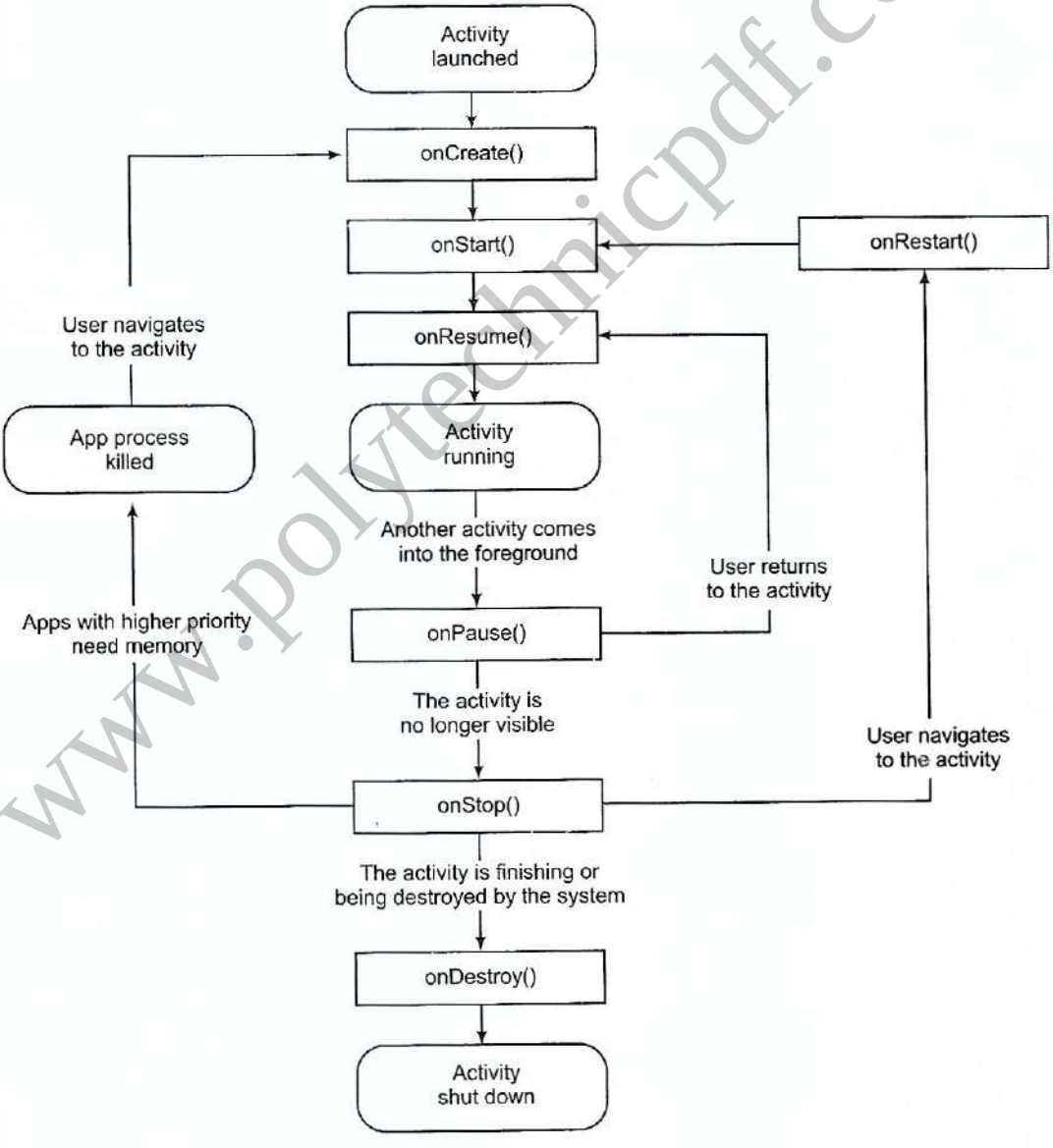
निम्नलिखित चार मुख्य घटक हैं जो एक Android एप्लीकेशन के भीतर उपयोग किए जा सकते हैं—

अनुक्रमांक	अवयव और विवरण
1.	**Activities :** वे UI को निर्देशित करते हैं और स्मार्ट फोन स्क्रीन के लिए यूजर इंटरैक्शन को संभालते हैं।
2.	**Services :** वे एक एप्लीकेशन से जुड़े बैकग्राउंड प्रसंस्करण को संभालते हैं।
3.	**Broadcast Receivers :** वे Android OS और एप्लीकेशन के बीच संचार को संभालते हैं।
4.	**Content Providers :** वे डेटा और डेटा बेस प्रबंधन समस्याओं को संभालते हैं।

Activities :

एक Activity यूजर इंटरफ़ेस के साथ एक स्क्रीन का प्रतिनिधित्व करती है, जैसे कि विंडो या जावा का फ्रेम। Android Activity, Context Theme Wrapper क्लास का सबक्लास है।

यदि आपने C, C ++ या Java प्रोग्रामिंग भाषा के साथ काम किया है, तो आपने देखा होगा कि आप का प्रोग्राम main () फ़ंक्शन से शुरू होता है। इसी तरह से, Android प्रणाली on Create () कॉल बैक मेथड पर कॉल के साथ शुरू होने वाली Activity में अपना प्रोग्राम initiate करती है। कॉल बैक मेथड्स का एक अनुक्रम है जो एक Activity शुरू करता है और कॉलबैक मेथड्स का एक क्रम होता है जो एक Activity को टिअर डाउन कर देता है जैसा कि नीचे Activity जीवन चक्र आरेख में दिखाया गया है :



चित्र : 1.4 : Activity flow chart

Activity क्लास निम्नलिखित कॉल बैक अर्थात इवेंट्स को परिभाषित करता है। आपको सभी कॉल बैक मेथड्स को लागू करने की आवश्यकता नहीं है। हालाँकि, यह महत्वपूर्ण है कि आप हर एक को समझें और उनपर अमल करें जो यह सुनिश्चित करें कि आपका ऐप यूजर्स की अपेक्षा के अनुरूप व्यवहार करे।

अनुक्रमांक	कॉल बैक और विवरण
1.	**onCreate() :** यह पहला कॉल बैक है और जब Activity पहली बार बनाई जाती है तो उसे कॉल किया जाता है।
2.	**onStart() :** यूजर को Activity दिखाई (visible) देने पर इस कॉल बैक को कॉल किया जाता है।
3.	**onResume() :** यह तब कॉल किया जाता है जब यूजर एप्लीकेशन के साथ बात चीत करना शुरू करता है।
4.	**onPause() :** रुकी हुई (paused) Activity यूजर इनपुट प्राप्त नहीं करता है और किसी भी कोड को Execute नहीं कर सकता है और यह कॉल बैक तब कॉल किया जाता है जब वर्तमान Activity को रोका जा रहा है और पिछली Activity को फिर से शुरू किया जा रहा है।
5.	**onStop() :** यह कॉल बैक तब कॉल किया जाता है जब Activity दिखाई नहीं देती है।
6.	**onDestroy() :** सिस्टम द्वारा Activity को नष्ट करने से पहले यह कॉलबैक कॉल किया जाता है।
7.	**onRestart() :** यह कॉल बैक तब कॉल किया जाता है जब Activity को रोकने के बाद फिर से शुरू होता है।

उदाहरण :

यह उदाहरण आपको Android एप्लीकेशन Activity जीवन चक्र दिखाने के लिए सरल चरणों के माध्यम से ले जाएगा। हैलो वर्ल्ड उदाहरण में हमारे द्वारा बनाए गए Android एप्लीकेशन को संशोधित करने के लिए निम्नलिखित चरणों का पालन करें—

स्टेप्स	विवरण
1.	आप Android एप्लीकेशन बनाने के लिए Android स्टूडियो का उपयोग करेंग और एक पैकेज com.example.helloworld के तहत इसे हैलो वर्ल्ड नाम दें।
2.	नीचे बताए अनुसार Main Activity फ़ाइल MainActivity.java को संशोधित करें। बाकी फाइलें अपरिवर्तित रखें।
3.	Android एमुलेटर लॉन्च करने के लिए एप्लीकेशन चलाएँ और एप्लीकेशन में किए गए परिवर्तनों के परिणाम की पुष्टि करें।

निम्नलिखित संशोधित Main Activity फ़ाइल src / com.example.helloworld / MainActivity.java की कंटेंट है। इस फ़ाइल में मौलिक जीवन चक्र मेथड्स में से प्रत्येक शामिल है। लॉग संदेश बनाने के लिए Log.d () मेथड का उपयोग किया गया है—

```
package com.example.helloworld;

import android.os.Bundle;
import android.app.Activity;
import android.util.Log;

public class MainActivity extends Activity {
```

```
String msg = "Android : ";

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);
   Log.d(msg, "The onCreate() event");
}

/** Called when the activity is about to become visible. */
@Override
protected void onStart() {
   super.onStart();
   Log.d(msg, "The onStart() event");
}

/** Called when the activity has become visible. */
@Override
protected void onResume() {
   super.onResume();
   Log.d(msg, "The onResume() event");
}

/** Called when another activity is taking focus. */
@Override
protected void onPause() {
   super.onPause();
   Log.d(msg, "The onPause() event");
}

/** Called when the activity is no longer visible. */
@Override
protected void onStop() {
   super.onStop();
   Log.d(msg, "The onStop() event");
}

/** Called just before the activity is destroyed. */
@Override
```

```
    public void onDestroy() {
        super.onDestroy();
        Log.d(msg, "The onDestroy() event");
    }
}
```

एक Activity क्लास प्रोजेक्ट के ले-आउट / ले-आउट फ़ोल्डर में उपलब्ध XML फ़ाइल का उपयोग कर के सभी UI घटक को लोड करता है। निम्नलिखित कथन res / ले-आउट / activity_main.xml फ़ाइल से UI घटकों को लोड करता है :

```
setContentView(R.layout.activity_main);
```

एक एप्लीकेशन में बिना किसी प्रतिबंध के एक या अधिक activities हो सकती हैं। आपके एप्लीकेशन के लिए आपके द्वारा परिभाषित प्रत्येक Activity को आपके Android Manifest.xml फ़ाइल में घोषित किया जाना चाहिए और आपके ऐप की Main Activity को एक < intent-filter > के साथ manifest में घोषित किया जाना चाहिए, जिसमें MAIN action और लॉन्चर श्रेणी निम्नानुसार है :

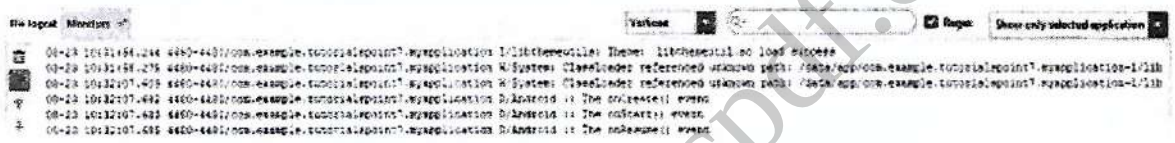
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.asianpublishers7.myapplication">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

यदि आपकी किसी एक Activity के लिए MAIN एक्शन या LAUNCHER श्रेणी घोषित नहीं की जाती है, तो आपका ऐप आइकन होम स्क्रीन की ऐप्स की सूची में दिखाई नहीं देगा।

आइए हमारी संशोधित हैलो वर्ल्ड को चलाने की कोशिश करें! । मैं मानता हूँ कि आपने एनवायरनमेंट सेटअप करते समय अपना AVD बनाया था। Android स्टूडियो से ऐप को चलाने के लिए, अपने प्रोजेक्ट की Activity फ़ाइलों में से एक खोलें और टूलबार से आइकन ▶ रन पर क्लिक करें । Android स्टूडियो आपके AVD पर ऐप इंस्टॉल करता है और इसे शुरू

करता है और अगर आपके सेटअप और एप्लीकेशन के साथ सब कुछ ठीक है, तो यह एमुलेटर विंडो प्रदर्शित करेगा और आपको Android स्टूडियो में लॉगकट विंडो में लॉग संदेशों का पालन करना चाहिए–

```
08-23 10:32:07.682 4480-4480/com.example.helloworld D/Android :: The
onCreate() event
08-23 10:32:07.683 4480-4480/com.example.helloworld D/Android :: The
onStart() event
08-23 10:32:07.685 4480-4480/com.example.helloworld D/Android :: The
onResume()
```



चित्र : 1.5 : Logcat window

आइए हम Android एमुलेटर पर लॉकस्क्रीन बटन पर क्लिक करने का प्रयास करें और यह Android स्टूडियों में लॉगकट विंडो में निम्नलिखित इवेंट्स के संदेश उत्पन्न करेगा :

```
08-23 10:32:53.230 4480-4480/com.example.helloworld D/Android :: The
onPause() event
08-23 10:32:53.294 4480-4480/com.example.helloworld D/Android :: The
onStop() event
```

आइए हम फिरसे Android एमुलेटर पर अपनी स्क्रीन को अनलॉक करने का प्रयास करें और यह Android स्टूडियो में लॉगकैट विंडो में निम्नलिखित इवेंट्स के संदेश उत्पन्न करेगा :

```
08-23 10:34:41.390 4480-4480/com.example.helloworld D/Android :: The
onStart() event
08-23 10:34:41.392 4480-4480/com.example.helloworld D/Android :: The
onResume() event
```

अगला, Android एमुलेटर पर फिर से बैक बटन पर क्लिक करने का प्रयास करते हैं और यह Android स्टूडियो में लॉगकैट विंडो में निम्न ईवेंट संदेशों को उत्पन्न करेगा और यह Android एप्लीकेशन के लिए Activity जीवन चक्र पूरा करता है।

```
08-23 10:37:24.806 4480-4480/com.example.helloworld D/Android :: The
onPause() event
08-23 10:37:25.668 4480-4480/com.example.helloworld D/Android :: The
onStop() event
08-23 10:37:25.669 4480-4480/com.example.helloworld D/Android :: The
onDestroy()
```

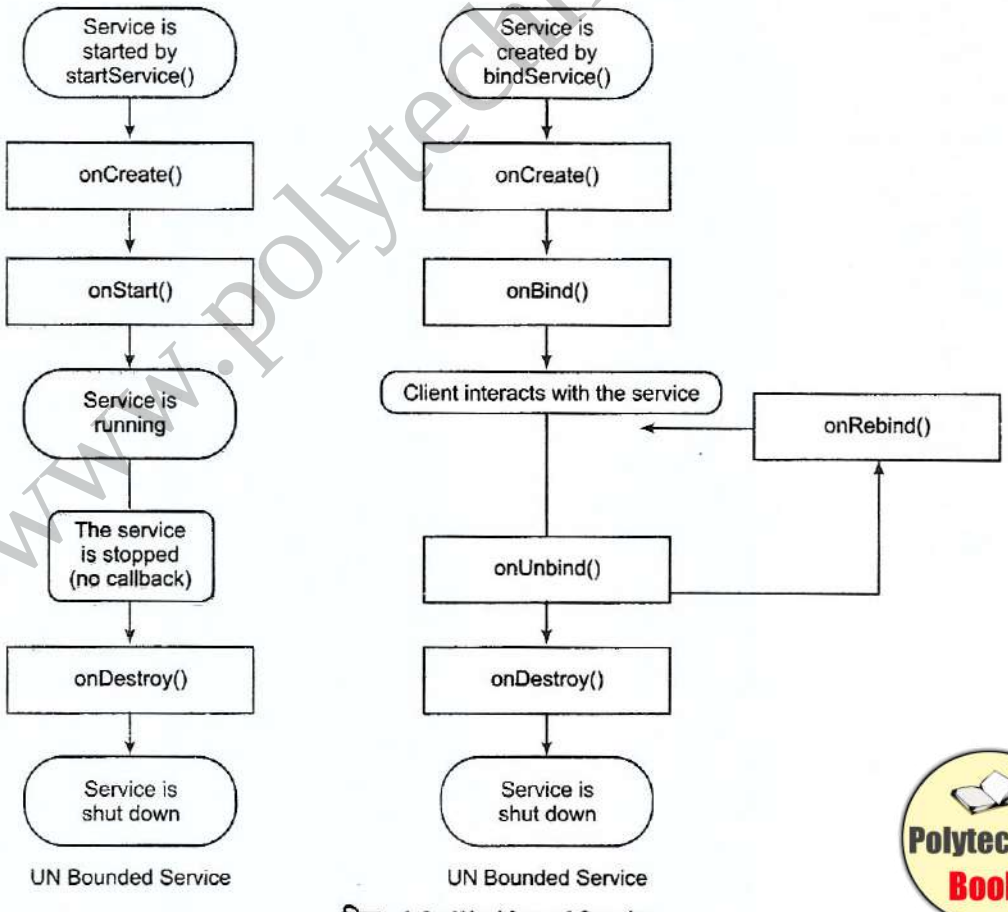
सर्विसेज (Services)

एक सर्विस एक घटक है जो यूजर के साथ बात चीत करने की आवश्यकता के बिना लंबे समय तक चलने वाले संचालन करने के लिए बैकग्राउंड में चलती है और यह तब भी काम करती है जब एप्लीकेशन नष्ट हो जाता है।

एक सर्विस अनिवार्य रूप से दो स्टेट ले सकती है—

अनुक्रमांक	स्टेट और विवरण
1.	**Started :** एकसर्विस Started है जब कोई एप्लीकेशन घटक, जैसे कि एक Activity, जिसे start Service () कहकर शुरू करता है। एक बार शुरू होने के बाद, एक सर्विस अनिश्चित काल के लिए बैकग्राउंड में चल सकती है, भले ही यह शुरू करने वाले घटक को नष्ट कर दिया गया हो।
2.	**Bound :** एक सर्विस तब Bound होती है जब कोई एप्लीकेशन घटक उसे bind Service() कॉल करता है। एक bound सर्विस एक क्लाइंट-सर्वर इंटरफ़ेस प्रदान करती है जो घटकों को सर्विस के साथ बातचीत करने, रिक्वेस्ट भेजने, परिणाम प्राप्त करने और यहाँ तक कि interprocess communication (IPC) के साथ प्रोसेसेज के लिए ऐसा करने की अनुमति देता है।

एक सर्विस में जीवन चक्र कॉलबैक मेथड्स हैं जिन्हें आप सर्विस की स्थिति में परिवर्तन की निगरानी के लिए लागू कर सकते हैं और आप उचित स्तर पर काम कर सकते हैं। निम्नलिखित आरेख जीवन चक्र को दिखाता है जब सर्विस startService () के साथ बनाई गई है और दाईं ओर आरेख जीवन चक्र को दिखाता है जब सर्विस bindService () के साथ बनाई गई है :



चित्र : 1.6 : Working of Service

एक सर्विस बनाने के लिए, आप एक जावा क्लास बनाते हैं जो सर्विस आधार क्लास या उसके एक मौजूदा सब क्लास का विस्तार करता है। सर्विस आधार क्लास विभिन्न कॉलबैक मेथड्स को परिभाषित करता है और सबसे महत्वपूर्ण नीचे दिया गया है। आपको सभी कॉलबैक मेथड्स को लागू करने की आवश्यकता नहीं है। हालाँकि, यह महत्वपूर्ण है कि आप हर एक को समझें और उन पर अमल करें जो यह सुनिश्चित करें कि आपका ऐप यूजर्स की अपेक्षा के अनुरूप व्यवहार करे।

अनुक्रमांक	कॉलबैक और विवरण
1.	**onStartCommand() :** सिस्टम इस मेथड को कॉल करता है जब एक अन्य घटक, जैसे कि एक Activity, रिक्वेस्ट करता है कि सर्विस शुरू की जाए, startService () को कॉल करके। यदि आप इस मेथड को कार्यन्वित करते हैं, तो आपकी जिम्मेदारी है कि जब सर्विस बंद हो जाए तो सर्विस को रोक दें, stopSelf() या stopService() मेथड्स को कॉल करके।
2.	**onBind() :** सिस्टम इस मेथड को कॉल करता है जब कोई अन्य घटक bindService()) को कॉल करके सर्विस के साथ bind करना चाहता है। यदि आप इस मेथड को कार्यन्वित करते हैं, तो आपको एक इंटरफ़ेस प्रदान कर ना होगा जो क्लाइंट IBinder ऑब्जेक्ट को वापस करके, सर्विस के साथ संचार करने के लिए उपयोग करता है।
3.	**onUnbind() :** सिस्टम इस मेथड को कॉल करता है जब सभी क्लाइंट सर्विस द्वारा प्रकाशित किसी विशेष इंटरफेस से डिस्कनेक्ट हो जाते हैं।
4.	**onRebind() :** जब नए क्लाइंट सर्विस से जुड़ जाते हैं, तो सिस्टम इस मेथड को कॉल करता है, क्योंकि पहले यह सूचित किया गया था कि सभी ने अपने onUnbind(Intent) में डिस्कनेक्ट कर दिया था।
5.	**onCreate() :** जब सिस्टम पहली बार onStartCommand () या onBind () का उपयोग करके बनाया जाता है, तो सिस्टम इस मेथड को कॉल करता है। यह कॉल एक बार सेट-अप करने के लिए आवश्यक है।
6.	**onDestroy() :** सिस्टम इस मेथड को कॉल करता है जब सर्विस अब उपयोग नहीं की जाती है और नष्ट हो रही है। आपकी सर्विस को किसी भी रिसोर्सेज जैसेकि थ्रेड्स, रजिस्टर्ड लिस्टनर्स, रिसीवर, आदि को साफ करने के लिए इसे लागू करना चाहिए।

निम्नलिखित स्केलेटन सर्विस जीवन चक्र मेथड्स में से प्रत्येक को प्रदर्शित करती है–

```
package com.asianpublishers;

import android.app.Service;
import android.os.IBinder;
import android.content.Intent;
import android.os.Bundle;

public class HelloService extends Service {

   /** indicates how to behave if the service is killed */
```

```
    int mStartMode;

    /** interface for clients that bind */
    IBinder mBinder;

    /** indicates whether onRebind should be used */
    boolean mAllowRebind;

    /** Called when the service is being created. */
    @Override
    public void onCreate() {

    }

    /** The service is starting, due to a call to startService() */
    @Override
   public int onStartCommand(Intent intent, int flags, int startId) {
       return mStartMode;
    }

    /** A client is binding to the service with bindService() */
    @Override
    public IBinder onBind(Intent intent) {
       return mBinder;
    }

    /** Called when all clients have unbound with unbindService() */
    @Override
    public boolean onUnbind(Intent intent) {
       return mAllowRebind;
    }

      /** Called when a client is binding to the service with
bindService()*/
    @Override
    public void onRebind(Intent intent) {

    }

   /** Called when The service is no longer used and is being destroyed */
```

```
    @Override
    public void onDestroy() {

    }
}
```

उदाहरण :

यह उदाहरण आपको सरल चरणों के माध्यम से दिखाएगा कि कैसे अपनी खुदकी Android सर्विस बनाएँ। हैलो वर्ल्ड उदाहरण में हमारे द्वारा बनाए गए Android एप्लीकेशन को संशोधित करने के लिए निम्नलिखित चरणों का पालन करें—

कदम	विवरण
1.	आप Android StudioIDE का उपयोग Android एप्लीकेशन बनाने के लिए और एक पैकेज com.example.asianpublishers7.myapplication के तहत My Application के रूप में नाम के रूप में हैलो वर्ल्ड उदाहरण में समझाया गया है।
2.	StartService () और stopService () मेथड्स को जोड़ने के लिए Main Activity फ़ाइल MainActivity.java को संशोधित करें।
3.	पैकेज com.example.My Application के तहत एक नई जावा फ़ाइल MyService.java बनाएँ। इस फाइल में Android सर्विस से संबंधित मेथड्स का कार्यन्वियन होगा।
4.	<service.../> टैग का उपयोग करके AndroidManifest.xml फ़ाइल में अपनी सर्विस को परिभाषित करें। एक एप्लीकेशन में बिना किसी प्रतिबंध के एक या अधिक सर्विसेज हो सकती हैं।
5.	Linear ले-आउट में दो बटन शामिल करने के लिए res/layout/activity_main.xml फ़ाइल की डिफ़ॉल्ट कंटेंट को संशोधित करें।
6.	in res/values/strings.xml file फ़ाइल में किसी भी स्थिरांक को बदलने की आवश्यकता नहीं है। Android स्टूडियो स्ट्रिंग मानों का ख्याल रखता है।
7.	Android एमुलेटर लॉन्च करने के लिए एप्लीकेशन चलाएँ और एप्लीकेशन में किए गए परिवर्तनों के परिणाम की पुष्टि करें।

निम्नलिखित Main Activity फ़ाइल MainActivity.java की संशोधित कंटेंट है। यह फ़ाइल मूलभूत जीवन चक्र मेथड्स में से प्रत्येक को शामिल कर सकती है। हमने सर्विस शुरू करने और रोकने के लिए startService () और stopService () जोड़ा हैं।

```
package com.example.asianpublishers7.myapplication;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

import android.os.Bundle;
import android.app.Activity;
import android.util.Log;
```

```
import android.view.View;

public class MainActivity extends Activity {
   String msg = "Android : ";

   /** Called when the activity is first created. */
   @Override
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      Log.d(msg, "The onCreate() event");
   }

   public void startService(View view) {
      startService(new Intent(getBaseContext(), MyService.class));
   }

   // Method to stop the service
   public void stopService(View view) {
      stopService(new Intent(getBaseContext(), MyService.class));
   }
}
```

निम्नलिखित MyService.java की कंटेंट है। इस फ़ाइल में आवश्यकताओं के आधार पर सर्विस से जुड़े एक या अधिक मेथड्स का कार्यान्वयन हो सकता है। अभी के लिए हम केवल दो मेथड्स को लागू करने जा रहे हैं। onStartCommand () और onDestroy ()—

```
package com.example.asianpublishers7.myapplication;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.support.annotation.Nullable;
import android.widget.Toast;

/**
   * Created by Asianpublishers7 on 8/23/2016.
*/

public class MyService extends Service {
   @Nullable
```

```
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        // Let it continue running until it is stopped.
        Toast.makeText(this, "Service Started", Toast.LENGTH_LONG).show();
        return START_STICKY;
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        Toast.makeText(this, "Service Destroyed", Toast.LENGTH_LONG).show();
    }
}
```


AndroidManifest.xml फ़ाइल की संशोधित कंटेंट निम्नलिखित होगी। यहाँ हमने अपनी सर्विस को शामिल करने के लिए <service.../> टैग जोड़ा है—

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.asianpublishers7.myapplication">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
```

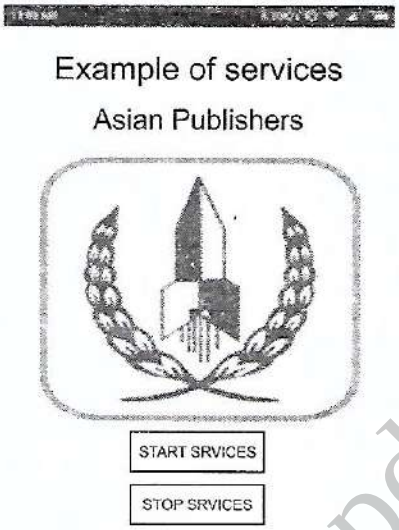
```
        <service android:name=".MyService" />
    </application>

</manifest>
```

निम्नलिखित दो बटन शामिल करने के res/layout/activity_main.xml फ़ाइल की कंटेंट होगी—

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
                        xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
                                android:layout_height="match_parent"
android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
            android:paddingBottom="@dimen/activity_vertical_margin"
tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Example of services"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:textSize="30dp" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Asian publishers "
        android:textColor="#ff87ff09"
        android:textSize="30dp"
        android:layout_above="@+id/imageButton"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="40dp" />

    <ImageButton
        android:layout_width="wrap_content"
```

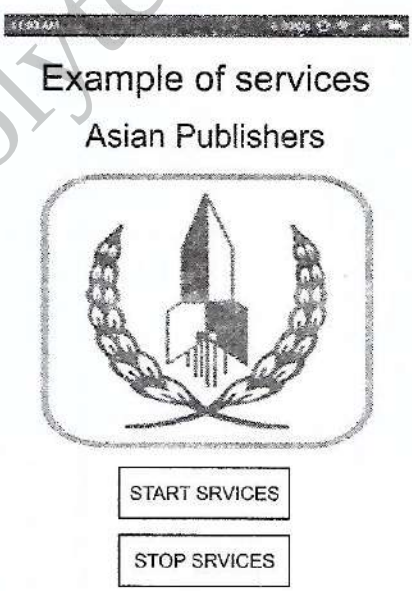
```
        android:layout_height="wrap_content"
        android:id="@+id/imageButton"
        android:src="@drawable/abc"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/button2"
        android:text="Start Services"
        android:onClick="startService"
        android:layout_below="@+id/imageButton"
        android:layout_centerHorizontal="true" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Stop Services"
        android:id="@+id/button"
        android:onClick="stopService"
        android:layout_below="@+id/button2"
        android:layout_alignLeft="@+id/button2"
        android:layout_alignStart="@+id/button2"
        android:layout_alignRight="@+id/button2"
        android:layout_alignEnd="@+id/button2" />

</RelativeLayout>
```

आइए हमारी संशोधित हैलो वर्ल्ड को चलाने की कोशिश करें! । मैं मानता हूँ कि आपने एनवायरनमेंट सेटअप करते समय अपना AVD बनाया था। Android स्टूडियो से ऐप को चलाने के लिए, अपने प्रोजेक्ट की Activity फ़ाइलों में से एक खोलें और टूलबार से आइकन ▶ रन पर क्लिक करें । Android स्टूडियो आपके AVD पर ऐप इंस्टॉल करता है और इसे शुरू करता है और अगर आपके सेट अप और एप्लीकेशन के साथ सबकुछ ठीक है, तो यह एमुलेटर विंडो प्रदर्शित करेगा—



चित्र : 1.7

अब अपनी सर्विस शुरू करने के लिए, स्टार्ट सर्विस बटन पर क्लिस करें, यह सर्विस शुरू करेगा और onStartCommand() मेथड में हमारी प्रोग्रामिंग के अनुसार, सिम्युलेटर के नीचे एक संदेश सर्विस शुरू होगी, जो निम्नानुसार सिम्युलेटर के नीचे दिखाई देगी–



चित्र : 1.8

सर्विस को रोकने के लिए, Stop Service बटन पर क्लिक कर सकते हैं।

Broadcast Receivers

Broadcast Receivers अन्य अनुप्रयोगों से या सिस्टम से संदेशों को प्रसारित करने के लिए प्रतिक्रिया देता है। ये संदेश कभी-कभी ईवेंट या इंटेंट्स कहलाते हैं। उदाहरण के लिए, एप्लीकेशन अन्य अनुप्रयोगों को यह बताने के लिए प्रसारण भी आरंभ कर सकते हैं कि कुछ डेटा डिवाइस में डाउनलोड कर लिए गए हैं और उनका उपयोग करने के लिए उपलब्ध है, इसलिए यह ब्रॉडकास्ट रिसीवर है जो इस संचार को बाधित करेगा और उचित एक्शन initiate करेगा।

ब्रॉडकास्टेड सिस्टम इंटेंट्स के लिए ब्रॉडकास्टर रिसीवर काम करने के लिए दो महत्वपूर्ण चरणों का पालन कर रहे हैं—

- ब्रॉडकास्टर रिसीवर बनाना।
- ब्रॉडकास्टर रिसीवर को रजिस्टर करना।

यदि आप अपने कस्टम इंटेंट्स को लागू करने जा रहे हैं तो एक अतिरिक्त कदम है फिर आपको उन इंटेंट्स को बनाना और प्रसारित करना होगा।

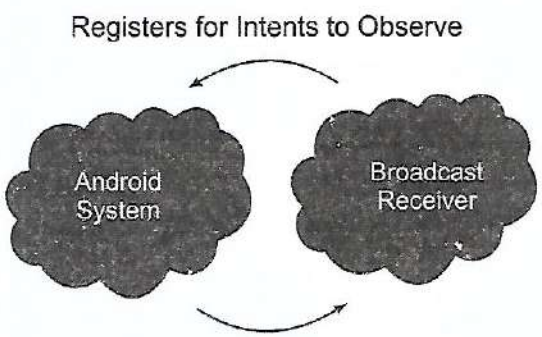
ब्रॉडकास्टर रिसीवर बनाना (Creating the Broadcast Receiver)

ब्रॉडकास्ट रिसीवर के सबक्लास के रूप में एक ब्रॉडकास्ट रिसीवर को इम्प्लीमेंट किया जाता है और onReceive() मेथड को ओवरराइड करके जहाँ प्रत्येक संदेश को Intent ऑब्जेक्ट पैरामीटर के रूप में प्राप्त किया जाता है।

```
public class MyReceiver extends BroadcastReceiver {
   @Override
   public void onReceive(Context context, Intent intent) {
    Toast.makeText(context, "Intent Detected.", Toast.LENGTH_LONG).show();
   }
}
```

ब्रॉडकास्ट रिसीवर को पंजीकृत करना (Registering Broadcast Receiver)

AndroidManifest.xml फ़ाइल में एक ब्रॉडकास्ट रिसीवर दर्ज करके विशिष्ट ब्रॉडकास्ट के लिए एक एप्लीकेशन सुनता है। विचार करें कि हम सिस्टम जनरेटेड इवेंट ACTION_BOOT_COMPLETED के लिए MyReceiver को पंजीकृत करने जा रहे हैं, जिसे सिस्टम द्वारा निकाल दिया जाता है क्योंकि Android सिस्टम ने बूट प्रोसेस पूरी कर ली है।



चित्र : 1.9 : ब्रॉडकास्टर रिसीवर (Broadcast Receiver)

```
<application
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    <receiver android:name="MyReceiver">

        <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED">
            </action>
        </intent-filter>

    </receiver>
</application>
```

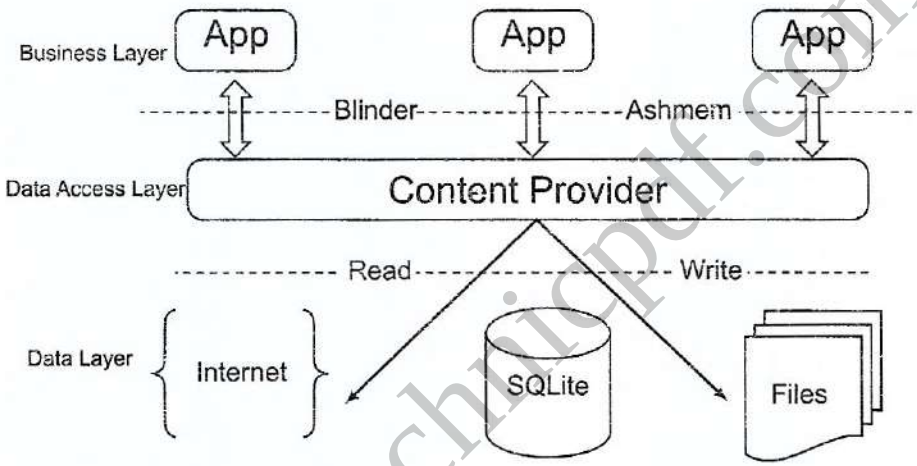
अब जब भी आपका Android डिवाइस बूट हो जाता है, तो इसे ब्रॉडकास्टर MyReceiver द्वारा इंटरसेप्ट किया जाएगा और OnReceive () के अंदर कार्यान्वित लॉजिक Execute किया जाएगा।

Intent क्लास में स्टैटिक क्षेत्रों के रूप में परिभाषित कई सिस्टम जनरेट किए गए इवेंट हैं। निम्न तालिका कुछ महत्वपूर्ण सिस्टम इवेंट्स को सूचीबद्ध करती है।

अनुक्रमांक	इवेंट कांस्टेंट और विवरण
1.	**android.intent.action.BATTERY_CHANGED :** बैटरी के बारे में चार्जिंग स्टेट, स्तर और अन्य जानकारी युक्त प्रसारण।
2.	**android.intent.action.BATTERY_LOW :** डिवाइस पर कम बैटरी की स्थिति का संकेत देता है।
3.	**android.intent.action.BATTERY_OKAY :** बैटरी कम होने के बाद अब ठीक होने का संकेत देती है।
4.	**android.intent.action.BOOT_COMPLETED :** यह एक बार प्रसारित होता है, सिस्टम के बूटिंग समाप्त होने के बाद।
5.	**android.intent.action.BUG_REPORT :** बग की रिपोर्टिंग के लिए Activity दिखाता है।
6.	**android.intent.action.CALL :** डेटा द्वारा निर्दिष्ट किसी को कॉल करें।
7.	**android.intent.action.CALL_BUTTON :** यूजर ने कॉल करने के लिए डायलर या अन्य उपयुक्त UI पर जाने के लिए 'कॉल' बटन दबाया।
8.	**android.intent.action.DATE_CHANGED :** तारीख बदल गई है।
9.	**android.intent.action.REBOOT :** डिवाइस रिबूट करता है ।

कंटेंट प्रोवाइडर (Content provider)

एक कंटेंट प्रोवाइडर घटक रिक्वेस्ट पर एक एप्लीकेशन से दूसरे में डेटा की आपूर्ति करता है। ऐसे रिक्वेस्ट ContentResolver क्लास के मेथड्स द्वारा नियंत्रित किए जाते हैं। एक कंटेंट प्रोवाइडर अपने डेटा को संग्रहीत करने के लिए विभिन्न मेथड्स का उपयोग कर सकता है और डेटा को डेटाबेस में, फाइलों में, या यहाँ तक कि एक नेटवर्क पर संग्रहीत किया जा सकता है।



चित्र : 1.10 : कंटेंट प्रोवाइडर (ContentProvider)

कभी-कभी अनुप्रयोगों में डेटा साझा करना आवश्यक होता है। यह वह जगह है जहाँ कंटेंट प्रोवाइडर बहुत उपयोगी हो जाते हैं।

कंटेंट प्रोवाइडर आपको एक ही स्थान पर कंटेंट को केंद्रीकृत करने की अनुमति देते हैं और कई अलग-अलग एप्लीकेशन इसे आवश्यकतानुसार एक्सेस करते हैं। एक कंटेंट प्रोवाइडर एक डेटाबेस की तरह बहुत व्यवहार करता है जहां आप इसे क्वेरी कर सकते हैं, इसकी कंटेंट संपादित कर सकते हैं, साथ ही insert(), update(), delete(), और query() मेथड्स का उपयोग करके कंटेंट को जोड़ या हटा सकते हैं। ज्यादातर मामलों में यह डेटा एक SQlite डेटाबेस में संग्रहीत होता है।

एक कंटेंट प्रोवाइडर को ContentProvider क्लास के सबक्लास के रूप में इम्प्लीमेंट किया जाता है और उसे API के एक मानक सेट को लागू करना चाहिए जो अन्य एप्लीकेशन को ट्रांसक्शन करने में सक्षम बनाता है।

```
public class My Application extends  ContentProvider {
}
```

कंटेंट URIs (Content URIs)

किसी कंटेंट प्रोवाइडर को क्वेरी करने के लिए, आप एक URL के रूप में क्वेरी स्ट्रिंग को निर्दिष्ट करते हैं जिसमें निम्न प्रारूप होता है—

```
<prefix>://<authority>/<data_type>/<id>
```

यहाँ URL के विभिन्न भागों का विवरण दिया गया है–

अनुक्रमांक	भाग विवरण
1.	**prefix :** यह हमेशा कंटेंट पर सेट होता है : //
2.	**authority :** यह कंटेंट प्रोवाइडर का नाम निर्दिष्ट करता है, उदाहरण के लिए कॉन्टेक्ट्स, ब्राउज़र आदि। तृतीय-पक्ष कंटेंट प्रोवाइडर के लिए, यह पूरी तरह से योग्य नाम हो सकता है, जैसे com.asianpublishers.statusprovider
3.	**data_type :** यह उस डेटा के प्रकार को इंगित करता है जो यह विशेष प्रोवाइडर प्रदान करता है। उदाहरण के लिए, यदि आप कॉन्टेक्ट्स कंटेंट प्रोवाइडर से सभी संपर्क प्राप्त कर रहे हैं, तो डेटापथ people होंगे और URL इस प्रकार होगा: this*content://contacts/people*
4.	**id :** यह अनुरोधित विशिष्ट रिकॉर्ड को निर्दिष्ट करता है। उदाहरण के लिए, यदि आप संपर्क कॉन्टेक्ट्स कंटेंट प्रोवाइडर में संपर्क नंबर 5 की तलाश कर रहे हैं तो URL इस कंटेंट प्रकार होगा : content://contacts/people/5.

कंटेंट प्रोवाइडर बनाएं (Create Content Provider)

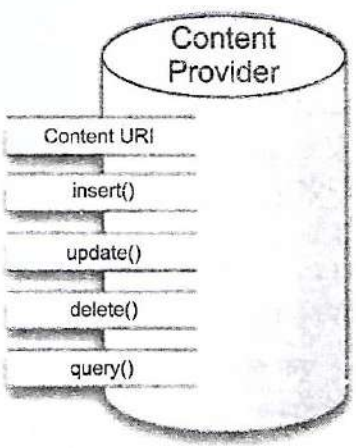
इसमें अपने स्वयं के कंटेंट प्रोवाइडर बनाने के लिए निम्नलिखित सरल स्टेप शामिल है।

- सबसे पहले आपको एक कंटेंट प्रोवाइडर क्लास बनाने की जरूरत है जो कि Content Provider base class को एक्सटेंट करे ।
- दूसरा, आपको अपने कंटेंट प्रोवाइडर URL पते को परिभाषित करने की आवश्यकता है जिसका उपयोग कंटेंट तक पहुंचने के लिए किया जाएगा।
- आगे आपको कंटेंट रखने के लिए अपना खुद का डेटाबेस बनाना होगा। आमतौर पर, Android SQLite डेटाबेस का उपयोग करता है और फ्रेमवर्क को onCreate() मेथड को ओवरराइड करने की आवश्यकता होती है जो प्रोवाइडर के डेटाबेस को बनाने या खोलने के लिए SQLite Open Helper मेथड का उपयोग करेगा। जब आपका एप्लीकेशन लॉन्च किया जाता है, तो उसके प्रत्येक कंटेंट प्रोवाइडर के onCreate() हैंडलर को मुख्य एप्लीकेशन थ्रेड पर बुलाया जाता है।
- आगे आपको विभिन्न डेटाबेस विशिष्ट संचालन करने के लिए कंटेंट प्रोवाइडर प्रश्नों को लागू करना होगा।

अंत में < provider > टैग का उपयोग करके अपनी कंटेंट प्रोवाइडर को अपनी Activity फ़ाइल में पंजीकृत करना होगा।

यहाँ उन मेथड्स की सूची दी गई है, जिन्हें आपको अपने कंटेंट प्रोवाइडर के काम करने के लिए कंटेंट प्रोवाइडर क्लास में ओवरराइड करने की आवश्यकता है–

- **onCreate() :** जब प्रोवाइडर शुरू किया जाता है तो इस मेथड को कॉल करता है।
- **query() :** यह मेथड क्लाइंट से रिक्वेस्ट प्राप्त करता है। परिणाम को Cursor ऑब्जेक्ट के रूप में लौटाया जाता है।



चित्र : 1.11 : कंटेंट प्रोवाइडर (ContentProvider)

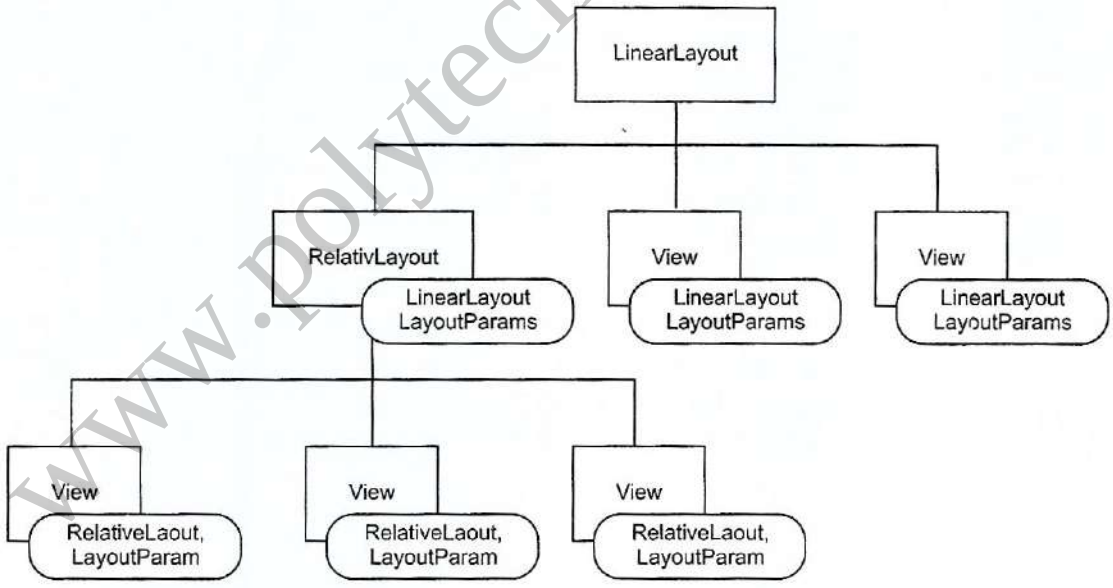
- **insert() :** यह मेथड कंटेंट प्रोवाइडर में एक नया रिकॉर्ड सम्मिलित करता है।
- **delete() :** यह मेथड कंटेंट प्रोवाइडर से मौजूदा रिकॉर्ड को हटा देता है।
- **update() :** यह मेथड कंटेंट प्रोवाइडर से मौजूदा रिकॉर्ड को अपडेट करता है।
- **getType() :** यह मेथड दिए गए URL में MIME प्रकार का डेटा लौटाता है।

§ 1.6 UI घटक (Views & notifications)

यूजर इंटरफ़ेस के लिए बुनियादी बिल्डिंग ब्लॉक एक Veiw ऑब्जेक्ट है जो Veiw क्लास से बनाया गया है और स्क्रीन पर एक आयताकार क्षेत्र (rectangular area) पर कब्जा कर लेता है और ड्राइंग और इवेंट हैंडलिंग के लिए जिम्मेदार है। विगेट्स के लिए Veiw बेस क्लास है, जो कि इंटर एक्टिव UI घटक जैसे बटन, टेक्स्ट फील्ड आदि बनाने के लिए उपयोग किया जाता है।

VeiwGroupVeiw का एक सबक्लास है और अदृश्य कंटेनर प्रदान करता है जो अन्य दृश्य या अन्य VeiwGroup होल्ड करते हैं और उनके ले-आउट गुणों को परिभाषित करते हैं।

तीसरे स्तर पर हमारे पास अलग-अलग ले-आउट हैं जो VeiwGroup क्लास के सबक्लास हैं और एक विशिष्ट ले-आउट एक Android यूजर इंटरफ़ेस के लिए दृश्य संरचना को परिभाषित करता है और Veiw / VeiwGroup ऑब्जेक्ट्स का उपयोग करके रन टाइम पर बनाया जा सकता है या आप अपने ले-आउट को XML फ़ाइल main_layout के साथ घोषित कर सकते हैं .xml जो आपके प्रोजेक्ट के लेआउट फ़ोल्डर में स्थित है।



चित्र : 1.12 : ले-आउट परम (Linear Layout)

एक लेआउट में किसी भी प्रकार के विगेट्स जैसे बटन, लेबल, टेक्स्टबॉक्स इत्यादि हो सकते हैं। निम्नलिखित LinearLayout XML फ़ाइल का एक सरल उदाहरण है—

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
```