Unit 3 (Full) and Unit 4 (Bundle and SQLite)

# 1 difference between Styles and Themes in Android

- Styles:
  A style in Android is a collection of properties that define the look and feel of a UI element. A style is applied to individual views, such as TextView, Button, etc.
    - It affects only the specific UI element to which it is applied.

Example:

```
<style name="CustomTextStyle">
   <item name="android:textColor">#FF0000</item>
   <item name="android:textSize">20sp</item></style>
```

- Themes:
  A theme is a collection of styles applied globally across an entire activity or application.
    - It defines the visual appearance for all views within the activity or app (such as background color, button style, etc.).

Example:

```
 <style name="AppTheme" parent="Theme.AppCompat.Light">
   <item name="android:colorPrimary">#FF0000</item>
   <item name="android:colorPrimaryDark">#B00000</item></style>
```

2 **Broadcast Receiver in Android with Example.**

- Broadcast Receiver:
    - A BroadcastReceiver listens for system-wide or application-specific events (broadcasts). It allows your app to react to different types of notifications or events, such as when the device's battery is low, a message is received, or the screen is turned off.
- Example:
    - A BroadcastReceiver that listens for incoming SMS

```
public class SMSReceiver extends BroadcastReceiver {
   @Override
   public void onReceive(Context context, Intent intent) {
      String sender = intent.getStringExtra("sender");
      String message = intent.getStringExtra("message");
      Log.d("SMSReceiver", "Sender: " + sender + ", Message: " + message);
   }
}
```

3 **Bundle Class in Android.**

Purpose of the Bundle Class:

- The Bundle class is used for passing data between Android components, such as activities, fragments, or services. It allows you to store and transfer key-value pairs.
- Bundle objects are often used for saving state and passing data through Intent objects or fragments.

```
// Create an Intent to start Activity 2
Intent intent = new Intent(Activity1.this, Activity2.class);

// Create a Bundle to pass data
Bundle bundle = new Bundle();
bundle.putString("user_name", "John Doe");
bundle.putInt("user_age", 25);

// Attach the Bundle to the Intent
intent.putExtras(bundle);

// Start Activity2
startActivity(intent);
```

## 4. **register a BroadcastReceiver in AndroidManifest.xml.**

Register a BroadcastReceiver in AndroidManifest.xml:

```
 <receiver android:name=".SMSReceiver">
    <intent-filter>
       <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
    </intent-filter>
</receiver>
```

5 **load a URL in a WebView in Android? Write the code to do so.**

```
WebView webView = findViewById(R.id.webview);
webView.loadUrl("https://www.example.com");
```

6 **syntax to initiate a phone call with a dialer using an Intent (without directly making the call)?**

```
Intent intent = new Intent(Intent.ACTION_DIAL);
intent.setData(Uri.parse("tel:123456789"));
startActivity(intent);
```

7 **Syntax to create an SQLite database in Android.**

```
SQLiteDatabase db = openOrCreateDatabase("user_database", MODE_PRIVATE, null);
```

```
String createTableQuery = "CREATE TABLE IF NOT EXISTS users (id INTEGER PRIMARY
KEY, name TEXT, email TEXT)";
db.execSQL(createTableQuery);
```

8 **Retrieve data from an SQLite database using a Cursor in Android?**

```
SQLiteDatabase db = openOrCreateDatabase("user_database", MODE_PRIVATE, null);
Cursor cursor = db.rawQuery("SELECT * FROM users", null);
if (cursor.moveToFirst()) {
    do {
        int id = cursor.getInt(cursor.getColumnIndex("id"));
        String name = cursor.getString(cursor.getColumnIndex("name"));
        String email = cursor.getString(cursor.getColumnIndex("email"));
        Log.d("User Info", "ID: " + id + ", Name: " + name + ", Email: " + email);
    } while (cursor.moveToNext());
}
cursor.close();
```

9 **Retrieve an integer from a Bundle using the key "age"?**

```
Bundle bundle = getIntent().getExtras();
int age = bundle.getInt("age");
```

10 **Mention permission related to battery of a mobile phone.**

Permission Related to Battery:
BATTERY_STATS (System-level permission):
Allows an app to access battery statistics.
ACCESS_FINE_LOCATION / ACCESS_COARSE_LOCATION (Indirectly related to battery):
Location services can impact battery usage.
REQUEST_IGNORE_BATTERY_OPTIMIZATIONS:
Allows apps to request ignoring battery optimizations, particularly useful for background services.

11 **Display a simple WebView in an Android application?**

To display a simple WebView, you need to add a WebView element in the XML layout and load a URL into it in the activity.
Step 1:
Add WebView in XML layout (res/layout/activity_main.xml):

```
<WebView
    android:id="@+id/webView"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

Step 2. Load a URL in the WebView in your Activity (MainActivity.java):

```
WebView webView = findViewById(R.id.webView);
webView.loadUrl("https://www.example.com");
```

**12 Basic steps to make a phone call programmatically in Android.**

To make a phone call programmatically, follow these steps:

Step 1: Add permission in AndroidManifest.xml:

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

Step 2: Initiate a phone call with an Intent:

```
Intent callIntent = new Intent(Intent.ACTION_CALL);
callIntent.setData(Uri.parse("tel:123456789"));
startActivity(callIntent);
```

**13 Syntax for Notification in Android, and why is it used?**

In Android, notifications are used to provide feedback to the user in the form of alerts, reminders, or information. These notifications are displayed outside the normal user interface, such as in the status bar, lock screen, or as a banner notification.

Basic Steps to Create and Show a Notification

1 Create a Notification Channel (Required for Android 8.0 and above)
2 Build the Notification
3 Display the Notification

Here is an example of how to create and show a simple notification in Android:

```
package com.example.simplenotification;
import android.app.Notification;
import android.app.NotificationManager;
import android.content.Context;
import android.os.Build;
import android.os.Bundle;
import android.support.v4.app.NotificationCompat;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
public class MainActivity extends AppCompatActivity {
   private static final String CHANNEL_ID = "my_channel";
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      Button showNotificationButton = findViewById(R.id.btnShowNotification);
      // Create Notification Channel (Required for Android 8.0 and above)
      if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
         NotificationChannel channel = new NotificationChannel(CHANNEL_ID, "Default",
NotificationManager.IMPORTANCE_DEFAULT);
         NotificationManager notificationManager =
getSystemService(NotificationManager.class);
         notificationManager.createNotificationChannel(channel);
```

```
        }
        // Set an onClick listener for the button
        showNotificationButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                showNotification();          }
        });
    }
    // Method to show notification
    private void showNotification() {
        // Build a simple notification
        NotificationCompat.Builder builder = new NotificationCompat.Builder(this, CHANNEL_ID)
                .setSmallIcon(R.drawable.ic_notification) // Replace with your icon
                .setContentTitle("Sample Notification")
                .setContentText("This is a simple notification example.")
                .setPriority(NotificationCompat.PRIORITY_DEFAULT)
                .setAutoCancel(true); // Notification disappears once clicked
        // Get the NotificationManager system service
        NotificationManager notificationManager = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
        // Show the notification with a unique ID
        notificationManager.notify(1, builder.build());
    }
}
```

**14. Purpose of AndroidManifest.xml**

The AndroidManifest.xml file is an essential part of every Android application. It provides important information to the Android system about the app, such as components, permissions, features, and other settings needed for the app to function properly.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapp">
    <!-- Declare permissions -->
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <!-- Declare application details -->
    <application
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/Theme.MyApp">
        <!-- Declare main activity -->
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"
            android:launchMode="singleTop">
```

```
        <!-- Intent filter to make MainActivity the entry point of the app -->
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>        </activity>    </application></manifest>
```

**15 Write the code to create a notification with a "Dismiss" button in Android.**

```
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.os.Build;
import androidx.core.app.NotificationCompat;

public class NotificationHelper {
        private static final String CHANNEL_ID = "notification_channel";

        public static void showNotification(Context context) {
        // Create the NotificationChannel for Android 8.0+
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        NotificationChannel channel = new NotificationChannel(
                CHANNEL_ID,
                "Notification Channel",
                NotificationManager.IMPORTANCE_DEFAULT
        );
        NotificationManager notificationManager =
                context.getSystemService(NotificationManager.class);
        notificationManager.createNotificationChannel(channel);
        }

        // Create an intent for the dismiss action
        Intent dismissIntent = new Intent(context, DismissReceiver.class);
        PendingIntent dismissPendingIntent = PendingIntent.getBroadcast(
        context, 0, dismissIntent, PendingIntent.FLAG_UPDATE_CURRENT |
PendingIntent.FLAG_IMMUTABLE
        );

        // Build the notification
        NotificationCompat.Builder builder = new NotificationCompat.Builder(context,
CHANNEL_ID)
        .setSmallIcon(android.R.drawable.ic_dialog_info)
        .setContentTitle("Sample Notification")
        .setContentText("This is a notification with a Dismiss button.")
```

```
        .setPriority(NotificationCompat.PRIORITY_DEFAULT)
        .addAction(android.R.drawable.ic_menu_close_clear_cancel, "Dismiss",
dismissPendingIntent)
        .setAutoCancel(true);

        // Show the notification
        NotificationManager notificationManager =
        (NotificationManager) context.getSystemService(Context.NOTIFICATION_SERVICE);
        notificationManager.notify(1, builder.build());
        }
}


import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;

public class DismissReceiver extends BroadcastReceiver {
        @Override
        public void onReceive(Context context, Intent intent) {
        // Handle the dismiss action
        NotificationManager notificationManager =
        (NotificationManager) context.getSystemService(Context.NOTIFICATION_SERVICE);
        notificationManager.cancel(1);
        }
}
```

Step 1 : Create a NotificationChannel for Android 8.0+.
Step 2: Use NotificationCompat.Builder to build the notification.
Step 3 : Add a "Dismiss" action using .addAction() and a PendingIntent.
Step 4 : Handle the dismiss action in a BroadcastReceiver.

**16 open and close a Cursor object after use in SQLite in Android?**

```
// Code to open and close a Cursor object
// Assume you have a database helper instance `dbHelper`
// Open a readable database
SQLiteDatabase db = dbHelper.getReadableDatabase();
// Perform a query to retrieve data
Cursor cursor = db.rawQuery("SELECT * FROM contacts", null);
// Loop through the cursor and retrieve data
if (cursor != null) {
   while (cursor.moveToNext()) {
      String contactName = cursor.getString(cursor.getColumnIndex("contact_name"));
```

```
        String contactNumber = cursor.getString(cursor.getColumnIndex("contact_number"));
            // Use the data (e.g., display it in a list)
    }
    // Close the cursor after use
    cursor.close();
}

// Close the database connection after use
db.close();
```

**17 Syntax to define a table in SQLite using SQL in Android?**

Answer:

To define a table in SQLite using SQL in Android, you use the CREATE TABLE statement. Here's the basic syntax:

```
CREATE TABLE table_name (
    column1_name column1_data_type,
    column2_name column2_data_type,
    ...
);
```

18 **retrieve data from a Bundle in an Activity?**

```
@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Retrieve the Bundle from the Intent
        Bundle bundle = getIntent().getExtras();
        if (bundle != null) {
        // Extract data using keys
        String data = bundle.getString("key");
        int number = bundle.getInt("anotherKey");

        // Use the data
        System.out.println("Data: " + data + ", Number: " + number);
        }
}
```

19 **Bundle object and add an integer with the key "score" in it.**

```
import android.os.Bundle;
```

```
public class Main {
	public static void main(String[] args) {
	// Create a Bundle object
	Bundle bundle = new Bundle();

	// Add an integer to the Bundle with the key "score"
	bundle.putInt("score", 100);

	// Example: Retrieve the integer back from the Bundle
	int score = bundle.getInt("score");
	System.out.println("Score: " + score);
	}
}
```

20 **Permission related to location of a mobile phone.**

To access the location of a mobile phone in Android, you need to request specific permissions in your app. There are two main types of location-related permissions:

## 1. ACCESS_FINE_LOCATION

- This permission allows access to precise location data using GPS, Wi-Fi, or cellular networks.
- It is suitable when your app requires high accuracy for location data.

## 2. ACCESS_COARSE_LOCATION

- This permission allows access to approximate location data using Wi-Fi or cellular networks.
- It is less accurate than ACCESS_FINE_LOCATION but sufficient for many use cases.

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

21. APPLICATIONS OF BroadcastReceiver

1. Battery Status

	Monitor changes in the battery level or charging status using broadcasts like Intent.ACTION_BATTERY_CHANGED.

2. Network Connectivity

	Detect changes in network connectivity, such as Wi-Fi or mobile data, using broadcasts like ConnectivityManager.CONNECTIVITY_ACTION.

3. Boot Completed

	Perform actions when the device finishes booting using Intent.ACTION_BOOT_COMPLETED. Useful for starting services or scheduling tasks.

4. Time and Date Changes

Respond to events like time zone changes (Intent.ACTION_TIMEZONE_CHANGED) or time updates (Intent.ACTION_TIME_CHANGED).

5. Incoming Calls or SMS

Listen for incoming calls (TelephonyManager.ACTION_PHONE_STATE_CHANGED) or SMS messages (Telephony.SMS_RECEIVED).

6. Package Installed or Removed

Get notified when apps are installed, updated, or removed using Intent.ACTION_PACKAGE_ADDED, Intent.ACTION_PACKAGE_REMOVED, etc.

7. Alarm Notifications

Handle scheduled alarms triggered using the AlarmManager.

8. Custom Broadcasts

Communicate between different components of your app by sending and receiving custom broadcasts.

9. Media Events

Listen for media events, such as headphones plugged in (Intent.ACTION_HEADSET_PLUG) or screen off/on events (Intent.ACTION_SCREEN_OFF, Intent.ACTION_SCREEN_ON).

10. Location Updates

Respond to location-related broadcasts from location services.

22. Code to Check Battery Status

```
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.BatteryManager;

public class BatteryStatusReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
    if (intent != null && Intent.ACTION_BATTERY_CHANGED.equals(intent.getAction())) {
    // Get battery level
    int level = intent.getIntExtra(BatteryManager.EXTRA_LEVEL, -1);
    int scale = intent.getIntExtra(BatteryManager.EXTRA_SCALE, -1);
    int batteryPercent = (int) ((level / (float) scale) * 100);

    // Get charging status
    int status = intent.getIntExtra(BatteryManager.EXTRA_STATUS, -1);
    boolean isCharging = status == BatteryManager.BATTERY_STATUS_CHARGING
            || status == BatteryManager.BATTERY_STATUS_FULL;

    // Display information
    System.out.println("Battery Level: " + batteryPercent + "%");
```

```
        System.out.println("Is Charging: " + isCharging);
        }
        }
}
```

23 .Service Registry in Android Purpose and Applications

The **Service Registry** in Android serves as a mechanism to facilitate interaction between different components of the Android framework or application. It acts as a centralized system for registering and accessing services within the Android system.

**Example:**

```
LocationManager locationManager =
        (LocationManager) context.getSystemService(Context.LOCATION_SERVICE);
```

## Benefits of Using Service Registry:

- Simplifies access to system and custom services.
- Provides a standard and consistent interface for interacting with Android services.
- Promotes reusability and modularity within the application.

24. Simple Program Location Updates using BroadcastReceiver

Step 1: Register the BroadcastReceiver

In your MainActivity, you can register your LocationBroadcastReceiver to handle location updates:

```
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.location.Location;
import android.os.Bundle;
import android.widget.Toast;

public class LocationBroadcastReceiver extends BroadcastReceiver {
        @Override
        public void onReceive(Context context, Intent intent) {
        if (intent != null && intent.getExtras() != null) {
        // Extract location data from Intent
        Location location = intent.getParcelableExtra("location");
        if (location != null) {
                double latitude = location.getLatitude();
                double longitude = location.getLongitude();
```

```
                Toast.makeText(context, "Location Updated: Lat=" + latitude + ", Lon=" +
longitude, Toast.LENGTH_SHORT).show();
        }
        }
        }
}
```

Step 2: Use LocationManager for Location Updates
Set up your MainActivity to request location updates and send them via broadcast:

```
import android.Manifest;
import android.content.Intent;
import android.content.IntentFilter;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;

public class MainActivity extends AppCompatActivity {

        private LocationBroadcastReceiver locationBroadcastReceiver = new
LocationBroadcastReceiver();
        private LocationManager locationManager;

        @Override
        protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Register BroadcastReceiver
        registerReceiver(locationBroadcastReceiver, new
IntentFilter("com.example.LOCATION_UPDATE"));

        // Request Location Permissions (Runtime Permission for Android 6.0+)
        ActivityCompat.requestPermissions(this, new
String[]{Manifest.permission.ACCESS_FINE_LOCATION}, 1);

        // Get LocationManager and set up location updates
        locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);

        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED) {
```

```
        locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 5000,
10, new LocationListener() {
                @Override
                public void onLocationChanged(Location location) {
                // Send Broadcast with updated location
                Intent intent = new Intent("com.example.LOCATION_UPDATE");
                intent.putExtra("location", location);
                sendBroadcast(intent);
                }

                @Override
                public void onStatusChanged(String provider, int status, Bundle extras) {
                }

                @Override
                public void onProviderEnabled(String provider) {
                }

                @Override
                public void onProviderDisabled(String provider) {
                }
        });
        }
        }

        @Override
        protected void onDestroy() {
        super.onDestroy();
        // Unregister BroadcastReceiver
        unregisterReceiver(locationBroadcastReceiver);
        }
}
```

25. Code to Handle Links in WebView

```
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.webkit.WebSettings;
import android.webkit.WebView;
import android.webkit.WebViewClient;
import androidx.appcompat.app.AppCompatActivity;

public class WebViewActivity extends AppCompatActivity {
```

```
	@Override
	protected void onCreate(Bundle savedInstanceState) {
	super.onCreate(savedInstanceState);
	setContentView(R.layout.activity_webview);

	WebView webView = findViewById(R.id.webView);

	// Configure WebView settings
	WebSettings webSettings = webView.getSettings();
	webSettings.setJavaScriptEnabled(true); // Enable JavaScript if needed
	webSettings.setAllowFileAccess(false);  // Disable file access for security
	webSettings.setAllowUniversalAccessFromFileURLs(false); // Restrict universal file
access

	// Set WebViewClient to handle URL loading
	webView.setWebViewClient(new WebViewClient() {
	@Override
	public boolean shouldOverrideUrlLoading(WebView view, String url) {
		if (url.startsWith("https://example.com")) {
		view.loadUrl(url); // Internal links stay in WebView
		return true;
		} else {
		// Open external links in default browser
		Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(url));
		startActivity(intent);
		return true;
		}
	}
	});

	// Load the initial webpage
	webView.loadUrl("https://example.com");
	}
}
```