Ch-4 Working with Menu and Internal Application

Alarm manager (with Media Player)

AlarmManager is also a system service like other services on android, such as the Notification service. It helps us to execute some piece of code at a certain time when our application isn't in the foreground.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<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"
   tools:context="MainActivity">
   <Button
      android:id="@+id/button"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Alarm"
      android:layout_alignParentBottom="true"
      android:layout_centerHorizontal="true"
      android:layout_marginBottom="103dp" />
   <EditText
      android:id="@+id/time"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignParentTop="true"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="22dp"
      android:ems="10" />
</RelativeLayout>
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity {
   Button start;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      start = findViewById(R.id.button);
      start.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View view) {
            startAlert();
         }
      });
   }
   public void startAlert() {
      EditText text = findViewById(R.id.time);
      int i = Integer.parseInt(text.getText().toString());
      Intent intent = new Intent(this, MyBroadcastReceiver.class);
      PendingIntent pendingIntent = PendingIntent.getBroadcast(this.getApplicationContext(),
234324243, intent, 0);
      AlarmManager alarmManager = (AlarmManager)getSystemService(ALARM_SERVICE);
      alarmManager.set(AlarmManager.RTC_WAKEUP, System.currentTimeMillis() + (i * 1000),
pendingIntent);
      Toast.makeText(this, "Alarm set:: " + i + " seconds", Toast.LENGTH_LONG).show();
   }
}
```

MyBroadcastReceiver.java

```
class MyBroadcastReceiver extends BroadcastReceiver {
   @Override
   public void onReceive(Context context, Intent intent) {
      MediaPlayer mp=MediaPlayer.create(context, R.raw.alarm);
      mp.start();
      Toast.makeText(context, "Alarm....", Toast.LENGTH_LONG).show();
   }
}
```

AndroidManifest.xml

<!-- Paste after the activity tag -->

<receiver android:name=".MyBroadcastReceiver" ></receiver>

Recording video Using camera

AndroidManifest.xml

<!-- adding permissions on below line -->
<uses-feature android:name="android.hardware.camera" android:required="true"/>

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".MainActivity">
  <!-- creating a video view on below line -->
  <VideoView
     android:id="@+id/videoView"
     android:layout_centerInParent="true"
     android:layout_width="match_parent"
     android:layout_height="match_parent"
     android:layout_above="@id/idBtnRecordVideo"
     android:layout_margin="5dp" />
  <!-- creating a button to record a video on below line -->
  <Button
     android:id="@+id/idBtnRecordVideo"
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
     android:layout_alignParentBottom="true"
     android:layout_centerInParent="true"
     android:layout_margin="10dp"
     android:text="Record Video"
     android:textAllCaps="false" />
</RelativeLayout>
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity {
```

```
  // creating variables on below line.
  private Button recordVideoBtn;
  private VideoView videoView;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_main);

     // initializing variables on below line.
     recordVideoBtn = findViewById(R.id.idBtnRecordVideo);
     videoView = findViewById(R.id.videoView);

     // adding click listener for recording button.
     recordVideoBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
           // on below line opening an intent to capture a video.
           Intent i = new Intent(MediaStore.ACTION_VIDEO_CAPTURE);
           // on below line starting an activity for result.
           startActivityForResult(i, 1);
        }
     });
  }
  @Override
  protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
     super.onActivityResult(requestCode, resultCode, data);
     if (resultCode == RESULT_OK && requestCode == 1) {
        // on below line setting video uri for our video view.
        videoView.setVideoURI(data.getData());
        // on below line starting a video view
        videoView.start();
     }
  }
}
```

Handling Telephony Manager

The TelephonyManager class in Android provides information about the telephony services on a device. It allows you to retrieve details such as network type, operator name, SIM details, and call state.

Attribute	Description

getDeviceId() (deprecated)	Returns the unique device ID (IMEI/MEID). Use getImei() or getMeid() instead.
getImei(int slotIndex)	Returns the IMEI of the device for a given slot.
getMeid(int slotIndex)	Returns the MEID for a CDMA device.
getSimSerialNumber()	Returns the serial number of the SIM card.
getSimOperator()	Returns the mobile network operator code (MCC+MNC).
getSimOperatorName()	Returns the name of the SIM operator.
getSimCountryIso()	Returns the country code of the SIM provider.
getNetworkOperator()	Returns the network operator code.
getNetworkOperatorName()	Returns the name of the network operator.
getPhoneType()	Returns the phone type (GSM, CDMA, or NONE).
getCallState()	Returns the current call state (IDLE, RINGING, OFFHOOK).
getDataState()	Returns the mobile data connection state.
getAllCellInfo()	Returns the list of cell information from all radio access networks.

1. AndroidManifest.xml

Before accessing TelephonyManager, add the necessary permissions in AndroidManifest.xml:

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

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   android:padding="16dp">

   <Button
      android:id="@+id/btnGetInfo"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:text="Get Telephony Info" />

   <TextView
      android:id="@+id/tvInfo"
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Telephony Info will be displayed here."
        android:paddingTop="20dp"/>
</LinearLayout>
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity {

    private static final int PERMISSION_REQUEST_CODE = 1;
    private TelephonyManager telephonyManager;
    private TextView tvInfo;

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

        tvInfo = findViewById(R.id.tvInfo);
        Button btnGetInfo = findViewById(R.id.btnGetInfo);

        telephonyManager = (TelephonyManager) getSystemService(TELEPHONY_SERVICE);

        btnGetInfo.setOnClickListener(view -> {
            if (ActivityCompat.checkSelfPermission(this, Manifest.permission.READ_PHONE_STATE)
                    != PackageManager.PERMISSION_GRANTED) {
                ActivityCompat.requestPermissions(this,
                        new String[]{Manifest.permission.READ_PHONE_STATE}, PERMISSION_REQUEST_CODE);
            } else {
                displayTelephonyInfo();
            }
        });
    }

    private void displayTelephonyInfo() {
        StringBuilder info = new StringBuilder();
        if (ActivityCompat.checkSelfPermission(this, Manifest.permission.READ_PHONE_STATE)
                == PackageManager.PERMISSION_GRANTED) {

            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
                info.append("IMEI: ").append(telephonyManager.getImei()).append("\n");
            }
            info.append("SIM Operator: ").append(telephonyManager.getSimOperatorName()).append("\n");
            info.append("SIM Country: ").append(telephonyManager.getSimCountryIso()).append("\n");
            info.append("Network Operator:
").append(telephonyManager.getNetworkOperatorName()).append("\n");
           info.append("Is Roaming: ").append(telephonyManager.isNetworkRoaming()).append("\n");
            tvInfo.setText(info.toString());
```

```
    }
  }
 }
}
```

Media Player in Android

MediaPlayer in Android – Attributes Table & Implementation

The `MediaPlayer` class in Android is used for playing audio and video files. It provides methods to start, pause, stop, and control media playback.

Attributes of MediaPlayer

Example Implementation of MediaPlayer
**1. Add Required Permissions in `AndroidManifest.xml`

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

Layout File (activity_main.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical"
   android:gravity="center"
   android:padding="20dp">

   <Button
      android:id="@+id/btnPlay"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Play" />

   <Button
      android:id="@+id/btnPause"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Pause" />

   <Button
      android:id="@+id/btnStop"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
```

```
        android:text="Stop" />

</LinearLayout>
```

Java Code (MainActivity.java)

```
package com.example.mediaplayerdemo;

import android.media.MediaPlayer;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    private MediaPlayer mediaPlayer;

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

        Button btnPlay = findViewById(R.id.btnPlay);
        Button btnPause = findViewById(R.id.btnPause);
        Button btnStop = findViewById(R.id.btnStop);

        // Initialize MediaPlayer with a local file (Replace with your file)
        mediaPlayer = MediaPlayer.create(this, R.raw.sample_audio);

        // Play button functionality
        btnPlay.setOnClickListener(view -> {
            if (mediaPlayer != null && !mediaPlayer.isPlaying()) {
                mediaPlayer.start();
            }
        });

        // Pause button functionality
        btnPause.setOnClickListener(view -> {
            if (mediaPlayer != null && mediaPlayer.isPlaying()) {
                mediaPlayer.pause();
            }
        });

        // Stop button functionality
        btnStop.setOnClickListener(view -> {
            if (mediaPlayer != null) {
                mediaPlayer.stop();
                mediaPlayer.reset();
```

```
            mediaPlayer = MediaPlayer.create(this, R.raw.sample_audio);
        }
    });

    // Release resources when playback is completed
    mediaPlayer.setOnCompletionListener(mp -> {
        mediaPlayer.reset();
        mediaPlayer = MediaPlayer.create(this, R.raw.sample_audio);
    });
  }

  @Override
  protected void onDestroy() {
    super.onDestroy();
    if (mediaPlayer != null) {
        mediaPlayer.release();
        mediaPlayer = null;
    }
  }
}
```

**Alternative: Streaming Audio from URL**

For streaming audio from a URL instead of a local file:

Add music into MainActivity.java

```
mediaPlayer.setDataSource("https://www.example.com/sample.mp3");
mediaPlayer.prepareAsync();
mediaPlayer.setOnPreparedListener(mp -> mp.start());
```