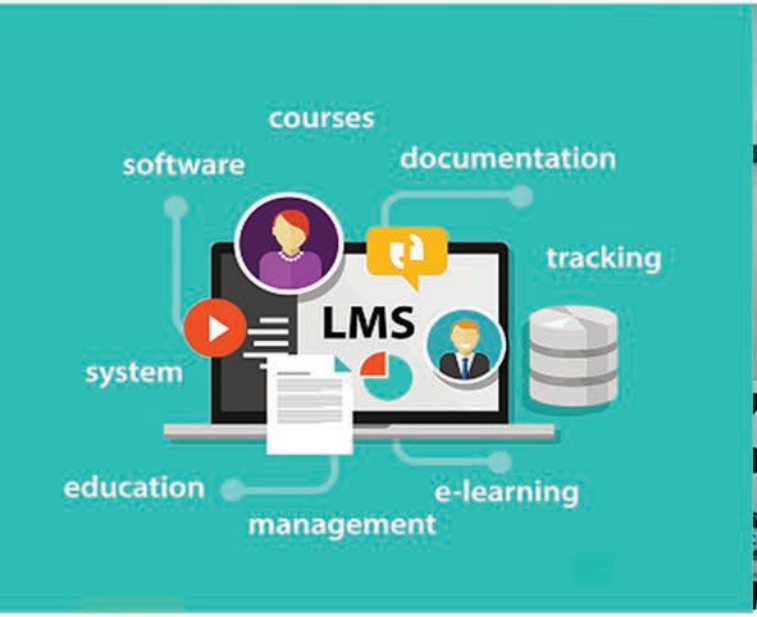
Andhra Pradesh State Skill Development Corporation



ANDROID APPLICATION DEVELOPMENT

INTERNAL STORAGE

Android - Internal Storage

Android Internal storage is the storage of the private data on the device memory. By default, saving and loading files to the internal storage are private to the application and other applications will not have access to these files. When the user uninstalls the applications the internal stored files associated with the application are also removed. However, note that some users root their Android phones, gaining superuser access. These users will be able to read and write whatever files they wish.

Android provides many kinds of storage for applications to store their data. These storage places are shared preferences, internal and external storage, SQLite storage, and storage via network connection.

In this chapter we are going to look at the internal storage. Internal storage is the storage of the private data on the device memory.

By default these files are private and are accessed by only your application and get deleted, when user delete your application.

Writing file

In order to use internal storage to write some data in the file, call the openFileOutput() method with the name of the file and the mode. The mode could be private , public e.t.c. Its syntax is given below.

```
FileOutputStream fOut = openFileOutput("file name here",MODE_WORLD_READABLE);
```

The method openFileOutput() returns an instance of FileOutputStream. So you receive it in the object of FileInputStream. After that you can call write method to write data on the file. Its syntax is given below

```
String str = "data";
fOut.write(str.getBytes());
fOut.close();
```

Reading file

In order to read from the file you just created , call the openFileInput() method with the name of the file. It returns an instance of FileInputStream. Its syntax is given below

```
FileInputStream fin = openFileInput(file);
```

After that, you can call read method to read one character at a time from the file and then you can print it. Its syntax is given below

```
int c;
String temp="";
while( (c = fin.read()) != -1){
   temp = temp + Character.toString((char)c);
}
```

```
//string temp contains all the data of the file.
fin.close();
```

Apart from the the methods of write and close, there are other methods provided by the FileOutputStream class for better writing files. These methods are listed below

Sr.No	Method & description
1	**FileOutputStream(File file, boolean append)** This method constructs a new FileOutputStream that writes to file.
2	**getChannel()** This method returns a write-only FileChannel that shares its position with this stream
3	**getFD()** This method returns the underlying file descriptor
4	**write(byte[] buffer, int byteOffset, int byteCount)** This method Writes count bytes from the byte array buffer starting at position offset to this stream

Apart from the the methods of read and close, there are other methods provided by the FileInputStream class for better reading files. These methods are listed below

Sr.No	Method & description
1	**available()** This method returns an estimated number of bytes that can be read or skipped without blocking for more input
2	**getChannel()** This method returns a read-only FileChannel that shares its position with this stream
3	**getFD()** This method returns the underlying file descriptor
4	**read(byte[] buffer, int byteOffset, int byteCount)** This method reads at most length bytes from this stream and stores them in the byte array b starting at offset

Example

Here is an example demonstrating the use of internal storage to store and read files. It creates a basic storage application that allows you to read and write from internal storage.

Following is the modified content of the xml res/layout/activity_main.xml.

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   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:orientation="vertical"
   android:padding="10dp"
   tools:context=".MainActivity">

   <EditText
      android:id="@+id/editText"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:focusable="true"
      android:hint="Enter Name"
      android:textColorHighlight="#0721B5"
      android:textColorHint="#D103B9" />
   <EditText
```

```
    android:id="@+id/editText1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:focusable="true"
    android:hint="Enter Mail"
    android:textColorHighlight="#0721B5"
    android:textColorHint="#D103B9" />

<Button
    android:id="@+id/button"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Save Data" />

<Button
    android:id="@+id/button2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Read Data" />

<TextView
    android:id="@+id/textView2"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:text="Read"
    android:gravity="center"
    android:textColor="#2196F3"
    android:textSize="25dp" />

</LinearLayout>
```

Following is the content of the modified main activity file src/MainActivity.java.

MainActivity.java

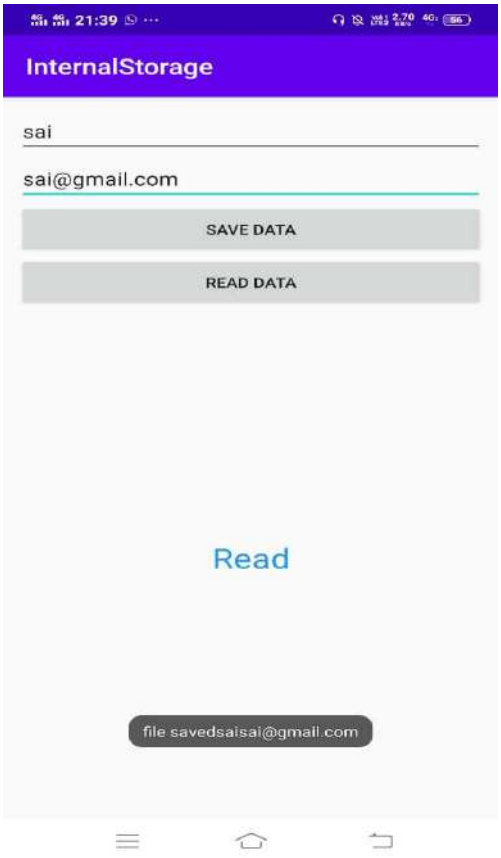
```
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;

public class MainActivity extends AppCompatActivity {

    Button b1,b2;
    TextView tv;
```

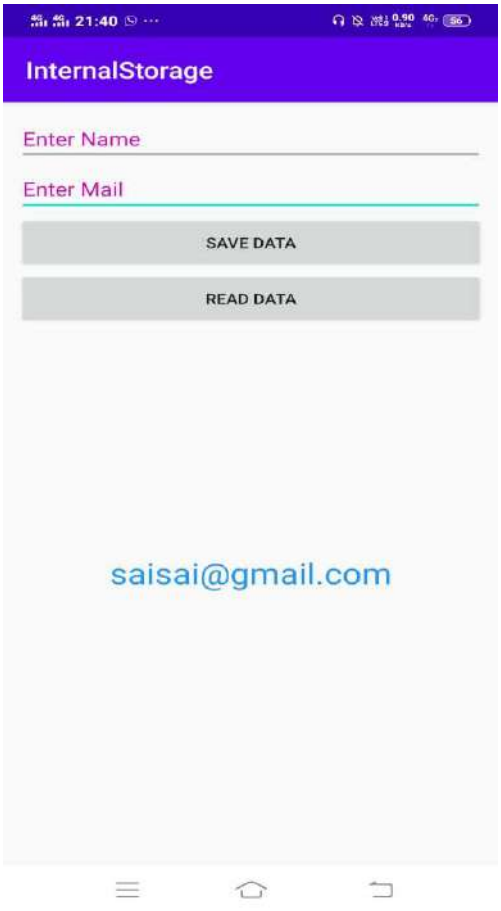
```
EditText et1,et2;

String data,data1;
static final int READ_BLOCK_SIZE = 100;
@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);

   b1 = findViewById(R.id.button);
   b2 = findViewById(R.id.button2);

   et1 = findViewById(R.id.editText);
   et2 = findViewById(R.id.editText1);
   tv = findViewById(R.id.textView2);

   b1.setOnClickListener(new View.OnClickListener() {

      @Override
      public void onClick(View v) {

         try {
            data = et1.getText().toString();
            data1 = et2.getText().toString();
            FileOutputStream fileout=openFileOutput("mytextfile.txt", MODE_PRIVATE);
            OutputStreamWriter outputWriter=new OutputStreamWriter(fileout);
            outputWriter.write(data);
            outputWriter.write(data1);
            outputWriter.close();
            Toast.makeText(MainActivity.this,"file saved"+data+data1,Toast.LENGTH_SHOR
T).show();
         }
         catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
         }
      }
   });

   b2.setOnClickListener(new View.OnClickListener() {

      @Override
      public void onClick(View v) {
         try {
            FileInputStream fileIn=openFileInput("mytextfile.txt");
            InputStreamReader InputRead= new InputStreamReader(fileIn);

            char[] inputBuffer= new char[READ_BLOCK_SIZE];
            String s="";
            int charRead;
```

```
            while ((charRead=InputRead.read(inputBuffer))>0) {
                // char to string conversion
                String readstring=String.copyValueOf(inputBuffer,0,charRead);
                s +=readstring;
            }
            InputRead.close();
            tv.setText(s);
            Toast.makeText(MainActivity.this,"file read",Toast.LENGTH_SHORT).show();
        }
        catch(Exception e){
        }
    }
  });
 }
}
```

Data Saving Image:

Data Read


21:39
InternalStorage
sai
sai@gmail.com
SAVE DATA
READ DATA
Read
file savedsaisai@gmail.com


21:40
InternalStorage
Enter Name
Enter Mail
SAVE DATA
READ DATA
saisai@gmail.com