Practical No. 6

Title: Android program using Shared Preferences, Internal and External Storage

Aim: Create an application to demonstrate Shared Preferences, Internal and External Storage

Introduction

Android provides many ways of storing data of an application. One of this way is called Shared Preferences. Shared Preferences allow you to save and retrieve data in the form of key,value pair.

In order to use shared preferences, you have to call a method getSharedPreferences() that returns a SharedPreference instance pointing to the file that contains the values of preferences.

SharedPreferences sharedpreferences = getSharedPreferences(MyPREFERENCES, Context.MODE_PRIVATE);

The first parameter is the key and the second parameter is the MODE. Apart from private there are other modes available that are listed below –

Sr.No	Mode & description
1	**MODE_APPEND** This will append the new preferences with the already existing preferences
2	**MODE_ENABLE_WRITE_AHEAD_LOGGING** Database open flag. When it is set , it would enable write ahead logging by default
3	**MODE_MULTI_PROCESS**

	This method will check for modification of preferences even if the sharedpreference instance has already been loaded
4	**MODE_PRIVATE** By setting this mode, the file can only be accessed using calling application
5	**MODE_WORLD_READABLE** This mode allow other application to read the preferences
6	**MODE_WORLD_WRITEABLE** This mode allow other application to write the preferences

You can save something in the sharedpreferences by using SharedPreferences.Editor class. You will call the edit method of SharedPreference instance and will receive it in an editor object. Its syntax is –

```
Editor editor = sharedpreferences.edit();
editor.putString("key", "value");
editor.commit();
```

Apart from the putString method , there are methods available in the editor class that allows manipulation of data inside shared preferences. They are listed as follows –

Sr. NO	Mode & description
1	**apply()** It is an abstract method. It will commit your changes back from editor to the sharedPreference object you are calling
2	**clear()** It will remove all values from the editor

3	**remove(String key)** It will remove the value whose key has been passed as a parameter
4	**putLong(String key, long value)** It will save a long value in a preference editor
5	**putInt(String key, int value)** It will save a integer value in a preference editor
6	**putFloat(String key, float value)** It will save a float value in a preference editor

Exercise - Create android application to demonstrate Shared Preferences

Implementation:
Program:

MainActivity.java

```
package com.example.database;

import androidx.appcompat.app.AppCompatActivity;

import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

    private TextView t1;
    private EditText e1;
```

```
    private Button b1;

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

        t1=findViewById(R.id.tv);
        e1=findViewById(R.id.editText);
        b1=findViewById(R.id.button);

        b1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String val=e1.getText().toString();
                SharedPreferences
sp1=getSharedPreferences("myPref",MODE_PRIVATE);
                SharedPreferences.Editor ed=sp1.edit();
                ed.putString("name",val);
                ed.apply();
                t1.setText(val);

            }
        });

        SharedPreferences sp1=getSharedPreferences("myPref",MODE_PRIVATE);
        String editval=sp1.getString("name","No value");
        t1.setText(editval);


    }
}
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
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">

    <TextView
        android:id="@+id/tv"
        android:layout_width="181dp"
```

```
        android:layout_height="54dp"
        android:text="No pref"
        android:textAlignment="center"
        android:textSize="24sp"
        app:layout_constraintBottom_toTopOf="@+id/editText"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.649" />

    <EditText
        android:id="@+id/editText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="88dp"
        android:ems="10"
        android:inputType="textPersonName"
        android:text="Name"
        app:layout_constraintBottom_toTopOf="@+id/button"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.497"
        app:layout_constraintStart_toStartOf="parent" />

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="268dp"
        android:text="Click"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.498"
        app:layout_constraintStart_toStartOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Output: