Experiment - 4

Name: Shaikh Tanveer **Rollno : 16DCO79**
Class : BE.CO **Batch : 03**

Aim : To develop an application that writes the data to SD card

#Theory

- Data and file storage overview

  Android provides several options for you to save your app data. The solution you choose depends on your specific needs, such as how much space your data requires, what kind of data you need to store, and whether the data should be private to your app or accessible to other apps and the user.
  - Internal file storage: Store app-private files on the device file system.
  - External file storage: Store files on the shared external file system. This is usually for shared user files, such as photos.
  - Shared preferences: Store private primitive data in key-value pairs.
  - Databases: Store structured data in a private database.

- Save files on device storage

  Android uses a file system that's similar to disk-based file systems on other platforms. This page describes how to work with the Android file system to read and write files with the File APIs.

  A File object works well for reading or writing large amounts of data in start-to-finish order without skipping around. For example, it's good for image files or anything exchanged over a network.

  The exact location of the where your files can be saved might vary across devices, so you should use the methods described on this page to access internal and external storage paths instead of using absolute file paths.

  To view files on a device, you can log the file location provided by methods such as File.getAbsolutePath(), and then browse the device files with Android Studio's Device File Explorer.

- Java.io.File Class

  The File class is Java’s representation of a file or directory path name. Because file and directory names have different formats on different platforms, a simple string is not adequate to name them. The File class contains several methods for working with the path name, deleting and renaming files, creating new directories, listing the contents of a directory, and determining several common attributes of files and directories
  - It is an abstract representation of file and directory pathnames.
  - A pathname, whether abstract or in string form can be either absolute or relative. The parent of an abstract pathname may be obtained by invoking the getParent() method of this class.
  - First of all, we should create the File class object by passing the filename or directory name to it. A file system may implement restrictions to certain operations on the

actual file-system object, such as reading, writing, and executing. These restrictions are collectively known as access permissions.

- Instances of the File class are immutable; that is, once created, the abstract pathname represented by a File object will never change.

➢ How to create File

A File object is created by passing in a String that represents the name of a file, or a String or another File object. For example,

```
File a = new File("/usr/local/bin/geeks");
```

defines an abstract file name for the geeks file in directory /usr/local/bin. This is an absolute abstract file name.

Constructors:

- File(File parent, String child) : Creates a new File instance from a parent abstract pathname and a child pathname string.
- File(String pathname) : Creates a new File instance by converting the given pathname string into an abstract pathname.
- File(String parent, String child) : Creates a new File instance from a parent pathname string and a child pathname string.
- File(URI uri) : Creates a new File instance by converting the given file: URI into an abstract pathname.

➢ Request App permission

Every Android app runs in a limited-access sandbox. If an app needs to use resources or information outside of its own sandbox, the app has to request the appropriate permission. You declare that your app needs a permission by listing the permission in the app manifest and then requesting that the user approve each permission at runtime (on Android 6.0 and higher). The Android framework provides similar methods as of Android 6.0 (API level 23), but using the support library makes it easier to provide compatibility with older versions of Android.

On all versions of Android, to declare that your app needs a permission, put a <uses-permission> element in your app manifest, as a child of the top-level <manifest> element. For example, an app that needs to access the write to storage would have this line in the manifest:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="com.example.snazzyapp">

    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <!-- other permissions go here -->

    <application ...>
        ...
    </application>
</manifest>
```

Program :-

MainActivity.java

```
package com.example.exp4;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import android.Manifest;
import android.content.pm.PackageManager;
import android.os.Build;
import android.os.Bundle;
import android.os.Environment;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
public class MainActivity extends AppCompatActivity {
    EditText et_name,et_content;
    Button b_save;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.M &&
checkSelfPermission(Manifest.permission.WRITE_EXTERNAL_STORAGE)
                != PackageManager.PERMISSION_GRANTED){
            requestPermissions(new String[]
{Manifest.permission.WRITE_EXTERNAL_STORAGE},1000);
        }
        et_name = (EditText) findViewById(R.id.editText);
        et_content = (EditText) findViewById(R.id.editText1);
        b_save = (Button) findViewById(R.id.button);
        b_save.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                String filename= et_name.getText().toString();
                String content= et_content.getText().toString();
                saveFile(filename,content);
            }
        });
    }
    private void saveFile(String filename,String content){
        String fileName = filename + ".txt";
        File file = new
File(Environment.getExternalStorageDirectory().getAbsolutePath(),fileName);
        try {
            FileOutputStream foa = new FileOutputStream(file);
            foa.write(content.getBytes());
            foa.close();
            Toast.makeText(this,"saved!!",Toast.LENGTH_SHORT).show();
        }catch (FileNotFoundException e)
        {
            e.printStackTrace();
            Toast.makeText(this,"File Not Found",Toast.LENGTH_SHORT).show();
        }catch (IOException e){
            e.printStackTrace();
            Toast.makeText(this,"Error Saving",Toast.LENGTH_SHORT).show();
        }
```

```
    }
    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {
        switch (requestCode){
            case 1000:
                if (grantResults[0] == PackageManager.PERMISSION_GRANTED){
                    Toast.makeText(this,"Permission
Granted",Toast.LENGTH_SHORT).show();
                }else{
                    Toast.makeText(this,"Permission Not
Granted",Toast.LENGTH_SHORT).show();
                    finish();
                }
        }
    }
}
```

activity_main.xml

```
<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" android:id="@+id/relativeLayout"
android:background="#b2bec3"
    >
<EditText
    android:id="@+id/editText"
    android:layout_width="300dp"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="155dp"
    android:ems="10"
    android:hint="File Name"
    android:padding="16dp"
    android:textColorHint="#ffffff"
    android:background="#bdc3c7"
    />
<EditText
    android:id="@+id/editText1"
    android:layout_width="300dp"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="250dp"
    android:ems="10"
    android:hint="Content"
    android:gravity="start|top"
    android:inputType="textMultiLine"
    android:padding="16dp"
    android:textColorHint="#ffffff"
    android:background="#bdc3c7"
    />
<Button
    android:id="@+id/button"
    android:layout_width="300dp"
```

```
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="480dp"
    android:text="save"
    android:paddingTop="15dp"
    android:paddingBottom="15dp"
    android:background="#dfe6e9"
    />
<TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="61dp"
    android:fontFamily="@font/billabong"
    android:text="Experiment NO 4"
    android:textSize="36sp" />
</RelativeLayout>
```

OUTPUT:-