* Internal storage

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 |