**Data Persistence**

1. **Principal Data Storage Options in Android**

* Saving Key-Value Sets
  + Using a shared preferences file for storing small amounts of information in key-value pairs.
* Saving files
  + Saving basic files such as to store long sequences of data that are generally read in order.
* Saving data in SQL databases
  + Using a SQLite database to read and write structured data.

Your data storage options are the following:

**Shared Preferences**

Store private primitive data in key-value pairs.

**Internal Storage**

Store private data on the device memory, default place for app storage.

**External Storage**

Store public data on the shared external storage. Not always available as external storage can be removed.

**SQLite Databases**

Store structured data in a private database.

**Network Connection**

Store data on the web with your own network server.

Cache files will be first removed if there is a lack of space.

1. **SharedPreferences API**

The SharedPreferences class provides a general framework that allows the developer to save and retrieve persistent key-value pairs of primitive data types. It can be used to save any primitive data: Booleans, floats, ints, longs and strings. The data will persist across user sessions even if the application is killed.

**To get a SharedPreferences object for your application, use one of two methods:**

getSharedPreferences() - Use this if you need multiple preferences files identified by name, which you specify with the first parameter. This may occur if multiple users have different preferences.

getPreferences() - Use this if you need only one preferences file for your Activity. Because this will be the only preferences file for your Activity, you don't supply a name.

**To write values:**

Call edit() to get a SharedPreferences.Editor.

Add values with methods such as putBoolean() and putString().

Commit the new values with commit()

To read values, use SharedPreferences methods such as getBoolean() and getString().

You can save files directly on the device's internal storage. By default, files saved to the internal storage are private to your application and other applications cannot access them (nor can the user). When the user uninstalls your application, these files are removed.

**To create and write a private file to the internal storage:**

Call openFileOutput() with the name of the file and the operating mode. This returns a FileOutputStream.

Write to the file with write().

Close the stream with close().

**To read a file from internal storage:**

Call openFileInput() and pass it the name of the file to read. This returns a FileInputStream.

Read bytes from the file with read().

Then close the stream with close().