COMP4336/9336 Lab 9

Data Storage

Lab Objectives

- Learning different approaches to store data on the mobile devices both internal and external memories.
- Learn how to save a collection of data on the internal memory of the device.
- Learn how to check the availability of any external media that is connecting to the phone.
- Learn how to save a collection of data on an external storage e.g. SD card.

Preparation

- ***Background**:*
  Android provides several options for you to save persistent application data. The solution you choose depends on your specific needs, such as whether the data should be private to your application or accessible to other applications (and the user) and how much space your data requires. Read more:

  http://developer.android.com/guide/topics/data/data-storage.html

  Based on the Android documents, there are five data storage options:
  1. **Shared Preferences,** for store private primitive data in key-value pairs.
  2. **Internal Storage,** for store private data on the device memory.
  3. **External Storage,** for store public data on the shared external storage.
  4. **SQLite Databases**, for store structured data in a private database.
  5. **Network Connection**, for store data on the web with your own network server.

  We will learn about first three in this lab and the last two in the next lab.

- ***Some useful Android classes and methods*:**
  - The *SharedPreferences* class provides a general framework that allows you to save and retrieve persistent key-value pairs of primitive data types. You can use SharedPreferences to save any primitive data: booleans, floats, ints, longs, and strings. This data will persist across user sessions (even if your application is killed).
    Note1:You can use *SharedPreferences* to save "user preferences," such as what ringtone a user has chosen.

    Note2: you can use *getSharedPreferences()* and *getPreferences()* to get a *SharedPreferences* object for your application. 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()*

  More details:
  http://developer.android.com/guide/topics/data/data-storage.html

- In order to create and write into a private file within 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()*.

  Here is an example from Android Developers:

```
String          FILENAME          =          "hello_file";
String        string        =        "hello        world!";

FileOutputStream      fos      =      openFileOutput(FILENAME,
Context.MODE_PRIVATE);
fos.write(string.getBytes());
fos.close();
```

  To read a file:

  - 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()*.

- In order to read or write files on the external storage, first, your app must acquire these permissions *READ_EXTERNAL_STORAGE or WRITE_EXTERNAL_STORAGE* system permissions like:

```
<manifest                                                  ...>
    <uses-permission

android:name="android.permission.READ_EXTERNAL_STORAGE"         />
android:name="android.permission.WRITE_EXTERNAL_STORAGE"        />
                                                              ...
</manifest>
```

If you need to both read and write files, then you need to request only the *write* one.

* You can use this code to check the availability of the external media:

```
/* Checks if external storage is available for read and write */

public         boolean         isExternalStorageWritable()         {
         String    state    =    Environment.getExternalStorageState();
              if       (Environment.MEDIA_MOUNTED.equals(state))     {
                                                return          true;
                                                                    }
                                       return                  false;
}
```

** You can use following code to obtain the capacity of the external storage if it exists.

```
StatFs                    stat                    =                  new
StatFs(Environment.getExternalStorageDirectory().getPath());
long     bytesAvailable     =     (long)stat.getBlockSize()     *
(long)stat.getBlockCount();
long megAvailable = bytesAvailable / 1048576;
```

Lab Tasks

Task1: Change and save a string in Shared Preferences
Develop an android application which is able to save any arbitrarily string (via an *edit box*) to the *shared preference.* It should be able to retrieve the stored string and show it on the screen after you restart the phone.

Task2: Check the availability of external storages
Develop an Android application, which is able to check and report the availability of any external media. If any external storage is available, it has to report the capacity of the storage in Gigs

Task3: Create a file on the external memory
Extend the previous work to create a new application, which would be able to read all available access points (APs) on the lab's environment and save it as a file in the external memory of the phone. It should be able to show the previous records as a list view as well when you press a button.

Each entry should include:

1-Date and time (format : yyyy-MM-dd HH:mm:ss)
2-Location information including latitude and longitude (you can obtain either from network or GPS)
3-SSID
4-Signal strength

Optional task: You can add the ability of deleting a given record after you hold your finger for 5 seconds on the given entry.