StacksOnStacks coding standards

Purpose

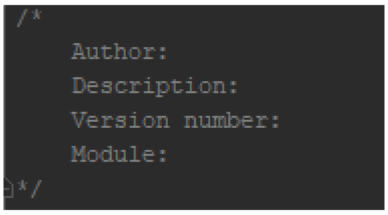
This document states the coding standards that will be followed by the team in order to achieve improved coding maintainability, quality ,usability and adaptability.

File Names

All the Android application files should have an extension of ".java", ".png" and ".xml". All the Arduino files should have an extension of ".ino". Furthermore, all the documentation should be in PDF format that has been created using LaTex. The names of all the files should follow an upper case camel-casing.

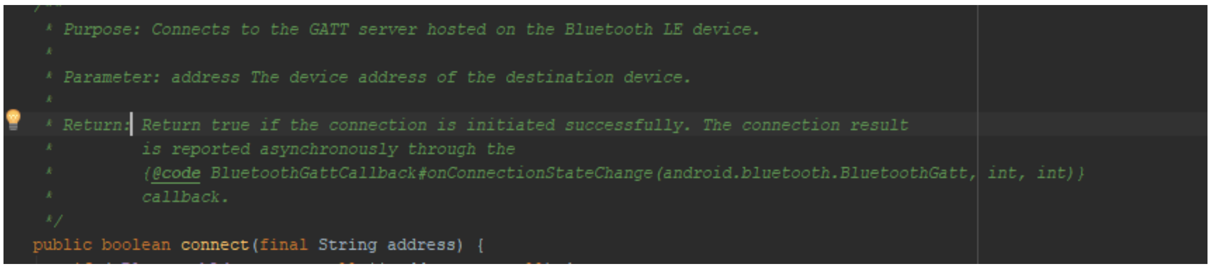
Comments

Comments in Arduino and Android(Java) are made by either a '//' or a '/*' followed by a '*/'. Comments will be used to provide more information to the reader about the classes and methods. Every file will have a file header with the version number, author, purpose, description and the module it belongs to. Below is a screenshot of how a class' comments will provide more information of the class.



```
/*
    Author:
    Description:
    Version number:
    Module:
*/
```

Besides the non-helper function, the functions of the classes will have the purpose of the function, parameter it takes in and what it returns. Below is a screenshot of how a main function's comments will provide more information of the function.



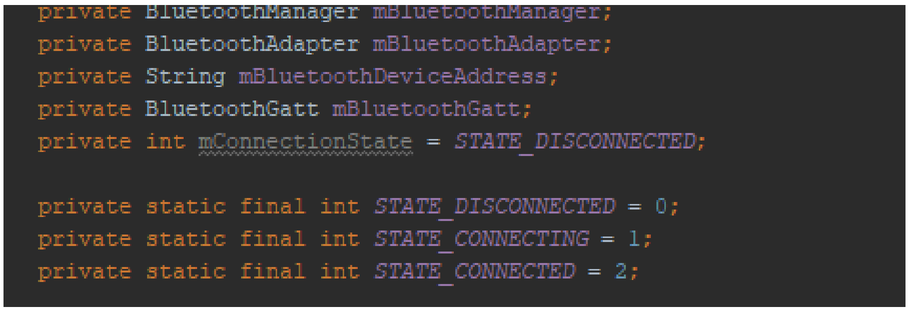
```
 * Purpose: Connects to the GATT server hosted on the Bluetooth LE device.
 *
 * Parameter: address The device address of the destination device.
 *
 * Return: Return true if the connection is initiated successfully. The connection result
 *         is reported asynchronously through the
 *         {@code BluetoothGattCallback#onConnectionStateChange(android.bluetooth.BluetoothGatt, int, int)}
 *         callback.
 */
public boolean connect(final String address) {
```

Lines and Indentation

Use 4 spaces and no tabs when creating functions. Packages, imports and classes will not use spaces or tabs. Ever line of code will have a maximum of 120 characters.

Naming conventions for variables, classes and functions

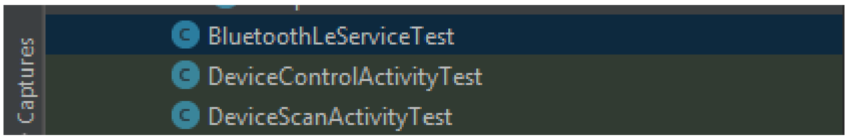
Upper-case camel casing will be used for naming classes, according to appropriate names. The static final variables will follow an Upper-case with underscores naming conventions and the other variables will follow a lower-case camel casing. Image below is the screenshot of an example of how the variables will be named:



```
private BluetoothManager mBluetoothManager;
private BluetoothAdapter mBluetoothAdapter;
private String mBluetoothDeviceAddress;
private BluetoothGatt mBluetoothGatt;
private int mConnectionState = STATE_DISCONNECTED;

private static final int STATE_DISCONNECTED = 0;
private static final int STATE_CONNECTING = 1;
private static final int STATE_CONNECTED = 2;
```

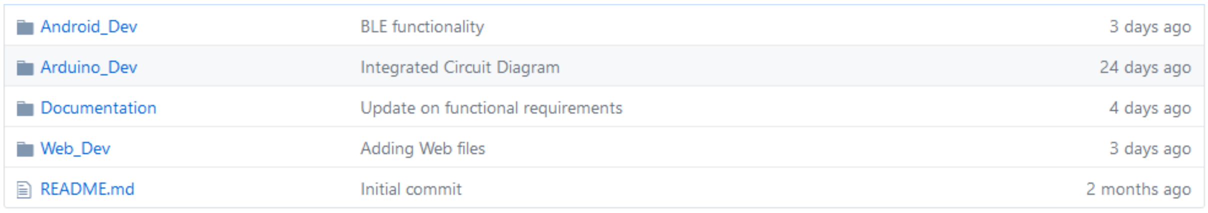
The variables, classes and objects names should be consistent throughout the classes, meaningless variable names will not be used, the names of the variables cannot contain space, control characters or special letters. Testing class names will use the upper-case camel casing method of the class to be tested followed by the keyword "Test". Image below is a screenshot of the testing classes of the Java files.



BluetoothLeServiceTest
DeviceControlActivityTest
DeviceScanActivityTest

Naming conventions for folders

For an improved organization, names of the folders will consist of an underscore to separate two words and every word will start with an uppercase letter. Every folder will contain the necessary files of that specific module or documentation. Image below is a screenshot of the gitHub master branch with all the folder names that are related to the module names of the project. This same folder structure exists on every team member's laptop.



Android_Dev	BLE functionality	3 days ago
Arduino_Dev	Integrated Circuit Diagram	24 days ago
Documentation	Update on functional requirements	4 days ago
Web_Dev	Adding Web files	3 days ago
README.md	Initial commit	2 months ago