Direct

A library to simplify the Android Wi-Fi Direct API

Direct is a library that provides a simplified interface to wrap around the Wi-Fi Direct API. In essence, this interface acts as a facade around the Wi-Fi Direct API by hiding its implementation details. Despite support of Wi-Fi Direct being released on Android 4.0 in October 2011, the Wi-Fi Direct API continues to be very complex and difficult to understand. Many developers steer clear of Wi-Fi Direct due to its complex nature and confusing documentation. The interface attempts to reduce these deterrents from preventing the use of Wi-Fi Direct in android development.

Wi-Fi Direct has peer-to-peer functionality, hence initially being called Wi-Fi Peer-to-Peer; however, this library is designed around a client server architecture instead. In order to establish connections and send objects amongst those connections, Wi-Fi Direct provides an abundance of details that are of little use to the developer. These details must be abstracted in order to enable the developer to use Wi-Fi Direct without needing to know the underlying functionality; thus, reducing the both the technical load on the developer and bugs, while also standardizing the process. This library aims to remove all redundant steps, and free the developer of the specific details of the API.

```
public abstract class DirectBroadcastReceiver extends
BroadcastReceiver
```

The `DirectBroadcastReceiver` class extends BroadcastReceiver, which receives and handles broadcasts intents. The BroadcastReceiver requires the extending class to implement onReceive(); however, this method generally promotes the If-Then-Else code smell, so the `DirectBroadcastReceiver` splits the onReceive() method into several more intuitive methods namely:

- `stateChanged(boolean wifiEnabled)`
  - This method is called when the `WIFI_P2P_STATE_CHANGED_ACTION` intent has been broadcasted.
- `discoveryChanged(boolean discoveryEnabled)`
  - This method is called when the `WIFI_P2P_DISCOVERY_CHANGED_ACTION` intent has been broadcasted.
- `peersChanged()`
  - This method is called when the `WIFI_P2P_PEERS_CHANGED_ACTION` intent has been broadcasted.
- `connectionChanged(WifiP2pInfo p2pInfo, NetworkInfo info, WifiP2pGroup p2pGroup)`
  - This method is called when the `WIFI_P2P_CONNECTION_CHANGED_ACTION` intent has been broadcasted.
- `thisDeviceChanged(WifiP2pDevice thisDevice)`
  - This method is called when the `WIFI_P2P_THIS_DEVICE_CHANGED_ACTION` intent has been broadcasted.

```
public abstract class Direct
```

The Direct abstract class contains common behaviour and variables that are inherited by both the `Host` and `Client` classes. In particular this class creates an instance of IntentFilter with listens for the following intent actions:

- `WIFI_P2P_STATE_CHANGED_ACTION`
  - This action indicates whether Wi-Fi P2P is enabled or disabled.
- `WIFI_P2P_DISCOVERY_CHANGED_ACTION`
  - This action indicates whether the peer discovery has either been started or stopped.
- `WIFI_P2P_PEERS_CHANGED_ACTION`

        - This action indicates that the available peer list has changed. The peer list will be changed when peers are lost, found, or updated. This will be exclusively used by the host to unregister clients who have been lost.
    - `WIFI_P2P_CONNECTION_CHANGED_ACTION`
        - This action indicates that the Wi-Fi P2P connectivity has changed. This will be used to get a handle on the:
            - WifiP2pInfo
                - This class represents Wi-Fi P2P group connection information. This class contains the field groupFormed which indicates whether a Wi-Fi P2P group has been successfully formed. This class also contains the field groupOwnerAddress which may be used to retrieve the host IP address, which is necessary for the client to register with the host.
            - NetworkInfo
                - This class represents the current network connection. This class is used to call the method isConnected() to determine whether the current device has established a connection and is able to perform data transactions.
            - WifiP2pGroup
                - This class represents the current Wi-Fi P2P group. This group consists of the group owner and one or more clients. In particular, this class will be used call the method getOwner() in to retrieve the host WifiP2pDevice.
    - `WIFI_P2P_THIS_DEVICE_CHANGED_ACTION`
        - This action indicates that the Wi-Fi P2P device has changed. This will be used to get a handle on the current WifiP2pDevice.

These actions are received by the abstract class `DirectBroadcastReceiver` which is extended by an anonymous class within both the `Host` and `Client` constructors and registered with the application context. The reason why these classes are anonymously extended is that they require access to private members of both the `Host` and `Client` classes.

Apart from creating the IntentFilter, the Direct class initializes the Channel which is the channel that connects the given application to the Wi-Fi P2P framework. This class will also use the respective Application to retrieve the application context in order to create a Handler in order for asynchronous methods to post a Runnable to the main thread.

```
public class Host extends Direct
```

```
public class Client extends Direct
```

This class is responsible for discovering services, and connecting to said services.

```
public Client(Application application, String service, String instance)
```

This constructor will create an instance of an anonymous class inheriting from DirectBroadcastReceiver and register said instance with the application context. The constructor will finally set the DnsSdServiceResponseListener and DnsSdTxtRecordListener to be reused with each service request, more on this will be covered in the explanation of the `startDiscovery` method.

```
public void startDiscovery(final ResultCallback callback)
```

This method will create a new service request instance and add said service request to the Wi-Fi P2P framework. This service request is required to initiate service discovery. If successful in adding the new service request, this method will then initiate service discovery. Service discovery is a process that involves scanning for requested services for the purpose of establishing a connection to a peer that supports an available service.

The service discovery notifies the library through the use of both a DnsSdServiceResponseListener and DnsSdTxtRecordListener. For the purpose of this library, only the DnsSdTxtRecordListener is used. This is because the DnsSdTxtRecordListener retrieves the `Map<String, String> record` which contains two entries that are useful to the client. These two entries are `SERVICE_NAME_TAG` and `REGISTRAR_PORT_TAG`. The `SERVICE_NAME_TAG` entry used to filter the discovered services, this entry will contain the unique identifier of the application. The `REGISTRAR_PORT_TAG` will contain the host registrar port, which in combination with the IP address of the host will be used to both register and unregister.

The services discovered which contain the proper `SERVICE_NAME_TAG` will be stored in `Map<WifiP2pDevice, Integer> nearbyHostDevices`, which is a map which the key is the respective host `WifiP2pDevice` and value is the respective registrar port stored in the `REGISTRAR_PORT_TAG`.

```
public void stopDiscovery(final ResultCallback callback)
```

This method will remove the service request created in `public void startDiscovery(final ResultCallback callback)` if said service request is not null. `Map<WifiP2pDevice, Integer> nearbyHostDevices` will be cleared and all peer discovery will be ceased.

```
public void connect(WifiP2pDevice hostDevice, ObjectCallback
dataCallback, ResultCallback callback)
```

This method will attempt to connect to the given `hostDevice` based on a `WifiP2pConfig` consisting of the MAC address from said host device. This method will only attempt to establish this connection if the given host device is contained within `Map<WifiP2pDevice, Integer> nearbyHostDevices`, this is to prevent the client from connecting to host devices that are not running with the proper unique identifier.

This method will not actually establish the connection but rather sends a connection request to the Wi-Fi P2P framework. In the event of a successful connection between the client and host device, a WIFI_P2P_CONNECTION_CHANGED_ACTION intent will be broadcasted and notify the client of a change in connectivity.

When a successful connection is broadcasted, the client will first create an instance of ObjectReceiver, which is essentially a ServerSocket listening for objects to be received from the host. At this point, the client has access to the host device IP address through the WifiP2pGroup. Given the host IP address and registrar port, the client then connects to the HostRegistrar through the ClientRegistrar and sends a Handshake object consisting of the client MAC address and port the client ObjectReceiver is listening on. This is to notify the host of the established connection and to provide the host with means of sending objects to the client ObjectReceiver.

The host will then send a Handshake object in return consisting of the MAC address and port of the host ObjectReceiver is listening on. This is to provide the client of the established connection and to provide the host with means of sending objects to the client ObjectReceiver.

```
public void disconnect(ResultCallback callback)
```

This method will begin by unregistering the client from the host, this is done by sending an Adieu object to the host through the ClientRegistrar to the HostRegistrar. This will notify the host that the client is disconnecting to prevent the host form continuing to send objects to the client ObjectReceiver. This is required as the Wi-Fi P2P framework has no reliable functionality to detect client disconnects. The registrar is required to unregister before disconnecting, for when the client leaves the P2P group the ClientRegistrar will no longer be permitted to connect to the HostRegistrar's ServerSocket.

After unregistering, this method will then remove the current P2P group, and through reflection, this method will attempt to delete the persistent P2P group, as P2P groups are by default persisted in the Wi-Fi P2P framework. By reflection I mean that the method to within the Wi-Fi P2P framework to delete these persistent groups is not visible and must be accessed through reflection. This will effectively disconnect the client from the host.

As usual, this method will not actually remove the P2P group but rather sends a request to the Wi-Fi P2P framework. In the event of a successful disconnect between the client and host device, a WIFI_P2P_CONNECTION_CHANGED_ACTION intent will be broadcasted and notify the client of a change in connectivity. When a successful disconnect is broadcasted, all host information will be cleared and the client ObjectReceiver will be stopped.

```
public void send(Serializable object, ResultCallback callback)
```

With the host IP address and the host ObjectReceiver port obtained from the registration handshake, this method will effectively send the host the respective `object` through ObjectTransmitter. The `object` is required to implement Serializable as communication between the ObjectTransmitter and the ObjectReceiver make use of ObjectInputStream and ObjectOutputStream.