Clayton Turner
CSIS 603 - Design Patterns

<center>Builder Pattern Assignment</center>

Class Overview: http://developer.android.com/reference/android/net/VpnService.Builder.html
Code:
https://android.googlesource.com/platform/frameworks/base/+/27f592d/core/java/android/net/Vp
nService.java (The class is too long to copy over - it would be 5+ pages)

The builder pattern being observed here is the Builder class within the VpnService class. Though
explicitly named Builder, this class would not cause confusion anywhere as it is declared within
VpnService so its scope restricts it from causing any confusion. A builder is often seen as an
incremental build of set/build methods. Here, the builder has methods such as:

```
        public Builder setSession(String session) {
                mConfig.session = session;
                return this;
        }
```

This allows the builder to set one of the variables which will be inevitably passed to the
VpnService constructor. At the time of calling setSession, the session variable only exists inside
the Builder. The constructor here is not a normal Java constructor or the cookie-cutter builder
constructor as VpnService is an Android service (it extends Service). This construction takes
place through the VpnService's getService() method:

```
        private static IConnectivityManager getService() {
                return IConnectivityManager.Stub.asInterface(
                        ServiceManager.getService(Context.CONNECTIVITY_SERVICE));
        }
```

The construction call for the Builder is as follows:

```
        public ParcelFileDescriptor establish() {
                mConfig.address = mAddresses;
                mConfig.routes = mRoutes;
                try{
                        return getService().establishVpn(mConfig);
                } catch(RemoteException e) {
                        throw new IllegalStateException(e);
                }
        }
```

Address and routes are singled out in this establish call because they have a variable number of
entries and had separate variables maintaining them (mAddresses and mRoutes) rather than the
mConfig variable itself holding the data. For example, the maximum transmission unit of the

VPN interface is a variable held in the mConfig and is just set when and if setMtu(int mtu) is called on the Builder. The getService().establishVpn(mConfig) does the construction of the VpnService as per its name.