Design Doc

Questions Going into Meeting:

* Error Handling
  + Do we do any revalidation on our end, or just send whatever the user inputs to Nessie (e.g. invalid lat, lng coordinates – do we handle or just make the request anyway and then pass the error on to the user)
  + Do we want to use/create our own Exceptions? Is this a model we think is useful?
* Method Design
  + Getters and Setters – is this good practice for python, or do we ditch?
  + Return objects – do we return just the object or the object and an error message?
* Instantiation of Objects
  + Do we accept a number of fields or a dictionary?
  + Or Both – this looks like the approach that Anthony took
* Directory Structure
  + How should the files be organized?
  + Should there be a master “client” class that can make any API call?

Running Design:

* Methods return objects (e.g. an Account object, an Atm object, etc.)
* Error Handling
  + The SDK Prevalidates User Input
  + If user input is invalid (e.g. latitude=4800), raise a ValidationException
  + If input seems valid but Nessie returns an error, raise an ApiException?
* Instantiation: Two Options
  + Instantiate field by field (e.g. Account(customer\_id, balance, account\_number=None, …): {…}
  + Instantiate with a dict (e.g. Account(dict): {…}
  + When instantiating with a dict, the user input must be prevalidated (e.g. ensure that they are providing all the required fields, see action items)
* Users have option of importing a specific client (e.g. AccountClient) or a master client which contains all possible API calls (e.g. just Client)
* Classes of Nessie objects (e.g. Account, Client, Atm, etc.) should be located in the model folder
* Don’t use Getters and Setters; rather, access the fields directly

Action Items:

* Reach out to Bori and see if there’s a list of optional and required parameters for API calls; there appears to be no documentation on this available through the website
* Reach out to Bori and see if he wants to do any sort of checkup on our code once we have a bit code more in the repo
* Research Exception handling in Python and see if this is the best idea; my feeling is that this is probably the simplest way to go
* Check if pip requires any sort of directory or file structure of projects before being listed
* ***Look into Unit Testing With Python (I forgot to bring this up at the meeting)***