Assignment 15

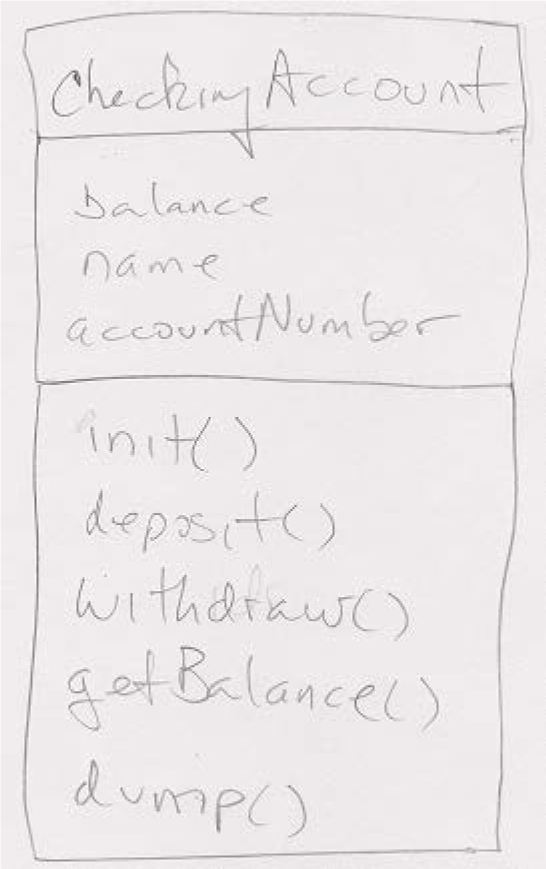
Methods - Classes and Objects

Assignment 15 is meant to demonstrate your understanding of using Python classes and objects. Primarily, I want you to see how to design a very simple class, instantiate an object of that class and invoke some methods on that object.

For this assignment I want you to create the following class:

```
Class CheckingAccount

Attributes

        balance

        name

        account

        number


Methods

        constructor()

                Initialize the object

        deposit()

                Make a deposit

        withdraw()

                Make a withdraw

        getBalance()

                Get the balance

        dump()

                Dump (Print) all information about the Account
```

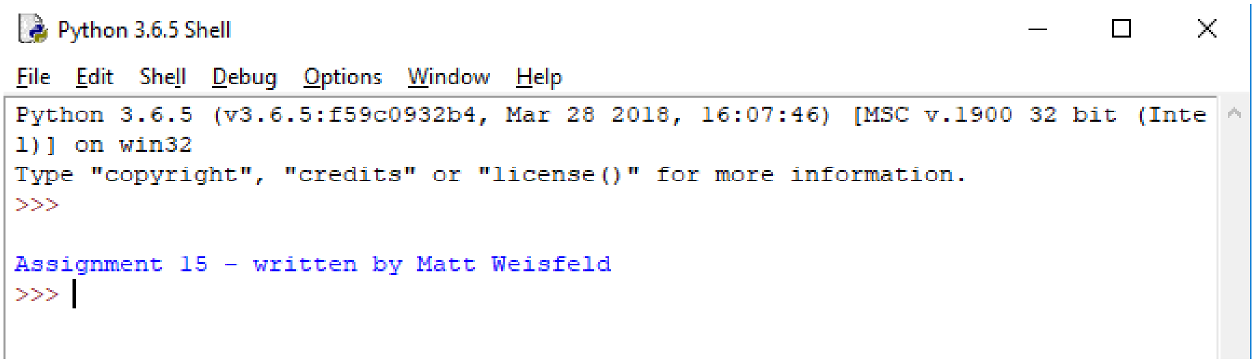
The class is represented by the following UML diagram.



Thus, take the following code and get it running in your environment:

```
class AccountP(object):
    def __init__(self, name, account_number, initial_amount):
        self._name = name
        self._no = account_number
        self._balance = initial_amount

    def deposit(self, amount):
        self._balance += amount

    def withdraw(self, amount):
        self._balance -= amount

    def get_balance(self):
        return self._balance

    def dump(self):
        print ("Dump Account Info")
        s = '%s, %s, balance: %s\n' % (self._name, self._no, self._balance)
        print (s)

# Main
print ("Assignment 15 - written by Matt Weisfeld")
```

When executed, this code will look something like this (it will simply print your name):



```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Inte
l)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 

Assignment 15 - written by Matt Weisfeld
>>> 
```

Now the interesting part – I want you to use this class. After you create an account, post some transactions to it.

For this assignment:

1. In the main, create an account called JohnSmith with the following initializations:
   ('John Smith', '19371554951', 20000)

2. Deposit $1000

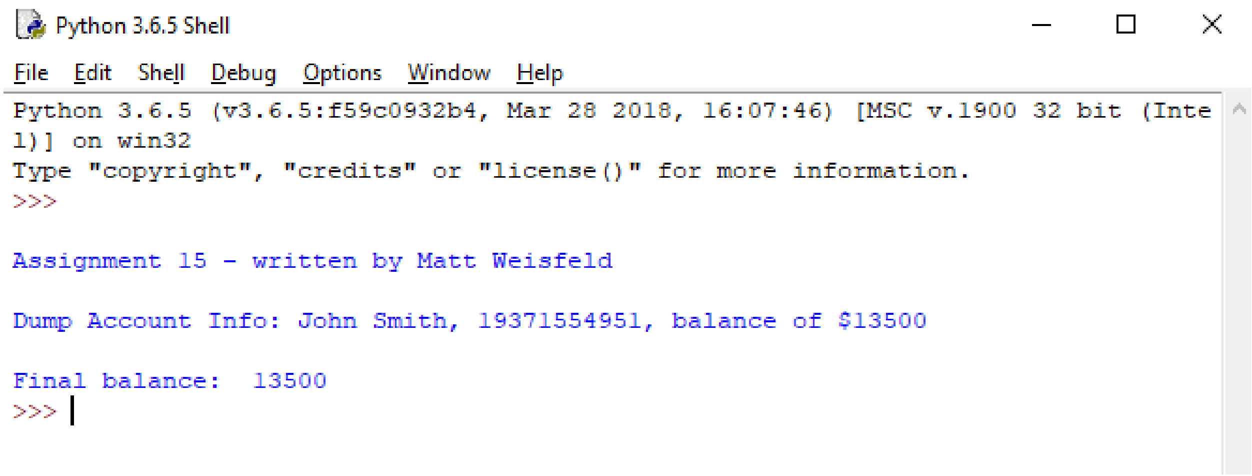
3. Withdraw $4000

4. Withdraw $3500

5. Print the balance

Test 01

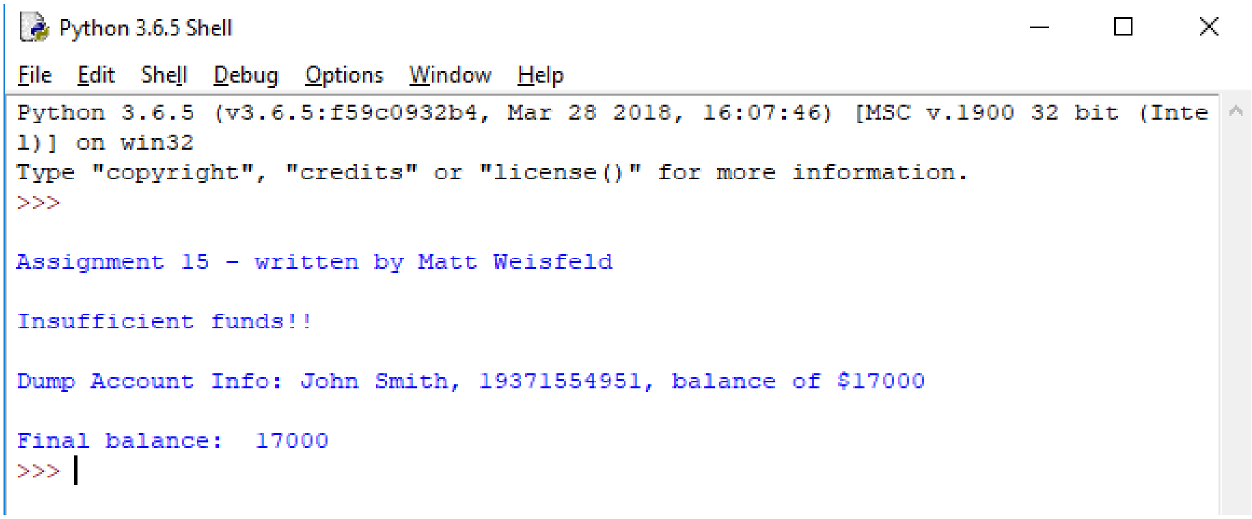
When I execute your code the first time the output **MUST** look exactly like this (part of the assignment is to figure out the formatting). I will use the exact entries listed above for my test. So make sure you final result is 13500.



```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Inte
l)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 

Assignment 15 - written by Matt Weisfeld

Dump Account Info: John Smith, 19371554951, balance of $13500

Final balance:  13500
>>> 
```

Test 02

Finally, since you don’t want someone to take out more money than they have in their account, do some simple range checking when attempting to withdraw money. So, if I try to take out $50,000, then an ‘Insufficient Funds’ message like this must appear.



```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Inte
l)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 

Assignment 15 - written by Matt Weisfeld

Insufficient funds!!

Dump Account Info: John Smith, 19371554951, balance of $17000

Final balance:  17000
>>> 
```

Submit the text file '*A15-your lastname.py*' via the Blackboard assignment portal.

I will copy it, paste it into my IDE and then test it.