Object-oriented design Part 2

CMSC 104 Section 2
November 25, 2025

Administrative Notes

Schedule for the rest of the semester (see Blackboard announcement)

- Today: Object-oriented design part 2; Classwork 9 (classes & dictionaries)
- December 2 (next Tuesday): Recursion; Classwork 10
- December 4 (next Thursday): Quiz 4
- December 9: review for Final
- December 11: Final Exam, 6-8 pm

Administrative Notes (p. 2)

I will post a “sample quiz 4” and a separate “answer key” for the sample quiz on GitHub sometime over the Thanksgiving break

- We don’t really have time to go over the sample quiz in class on December 2
    - But we’ll discuss any questions people have about it
- Recursion (the December 2 lecture topic) ***WILL NOT*** be on Quiz 4 but it ***WILL*** be on the Final

There will also be a ‘sample final’ posted on or around December 4

- We will go over that in class on December 9

We will try to get final grades posted during the week of December 15

Object-oriented programming

This lecture will focus on Classwork 9

First, a demo of what your program should produce

Now a look at the starter code

```
class BankAccount:
   """
   Start with a class definition
   """

   def __init__(self, number, name, balance):
       """
       We'll give you the init method. As noted, each account has an account number
as the key
       and its values are: the account number, the name of the account owner, and the
current balance
       You don't have to repeat the number in the values since it's already the key,
but I find it
       easier to do so.

       :param number: The account number; generated by whatever process you use to
generate your account numbers
       :param name: The name of the account owner, which will be typed in by the user
       :param balance: The initial balance of the account, which will be typed in by
the user
       """
       self.number = number
       self.name = name
       self.balance = balance
```

This is the class title and the __init__ method

The other necessary methods in the class definition

```
def deposit(self, number, amount):
   """
   You write this one. Your code should add the amount of the deposit to the current balance
   You should then print out a message that the deposit was successful, and show the new balance
   :param number:
   :param amount:
   :return:
   """

def withdraw(self, number, amount):
   if amount < self.balance:
       """
       If there's enough money in the account, go ahead and subtract the amount of the withdrawal from
       the account balance. Print out a message that the withdrawal was successful, and show the new balance
       """
       return True
   else:
       print("You don't have enough money!") #an error message saying that the withdrawal was refused due to insufficient funds
       return False

def view_account(self):
   return self.balance
```

You need a function to display the menu

```
def display_menu():
"""
You need a function to display the menu of choices to the user.
I have this function also get the user's choice, and return it to the main program

"""
```

You have to write this function!

Now a function to process the choice

```
def process_choice(choice,next_num):
"""
Now you need a function to process the choice - that is, to create a new account if the user entered 1;
to view the account balance if the user entered 2; to deposit money if the user entered 3; to withdraw
money if the user entered 4; to transfer money if the user entered 5. You don't need to worry about the user
choosing
6 in this function; that's handled directly in the main program.

This function will likely consist of a series of "if choice == " statements.

For a transfer, it might be easier to first call the method to withdraw funds from one account.
Then, only if that withdrawal is successful, you can call the method to deposit funds to the other account

"""
```

We’ll give you the main program

```
if __name__ == "__main__":
   next_num = 1000 # this is arbritrary. You can start at any number you want
   choice = 0 # we need an initial value. We could just as easily have prompted for it.
   bank_customers = {} # we start with an empty dictionary, which means we have no
customers
   while choice != 6:
       choice = display_menu()
       next_num = next_num + 1
       process_choice(choice, next_num)


   # the user has picked 6; time to print a farewell message
   print("Bye!")
```