Work on this lab with a new partner: someone with whom you have not yet worked.

Write the code for the lab below and each partner must upload the zipped src folder to the learning hub (learn.bcit.ca 🡪 Activities 🡪 Assignments 🡪 Lab2a) before the start of lesson 3.

Include your full name and your partner’s full name at the top of each file, using a Java Javadoc   
/\*\* @author \*/ tag (for example: /\*\* @author Paul Mills and Jason Harrison \*/).

Below are two Python classes. They define properties and methods for bank accounts and banks. Another file has a main function which creates some bank account objects and a bank object and calls their methods. Create the corresponding Java bank account class, bank class, and main class (which contains the main method) that does the same things. Comment every class, constructor, and method. Use proper naming conventions, indentation, etc… USE HASHMAPS AND SETS IN YOUR BANK CLASS.

**bank\_account.py**

class BankAccount:

def \_\_init\_\_(self, balance\_cdn, account\_number, member\_last\_name):

self.balance\_cdn = balance\_cdn

self.account\_number = account\_number

self.member\_last\_name = member\_last\_name

def withdraw(self, amount\_cdn):

self.balance\_cdn -= amount\_cdn

def deposit(self, amount\_cdn):

self.balance\_cdn += amount\_cdn

def transfer(self, amount\_cdn, recipient\_account):

recipient\_account.deposit(amount\_cdn)

self.withdraw(amount\_cdn)

**bank.py**

class Bank:

def \_\_init\_\_(self, bank\_name):

self.name = bank\_name

self.accounts = {} # A DICTIONARY OF KEY->VALUE PAIRS (String account number is the key; BankAccount is the value)

def add\_account(self, account):

self.accounts[account.account\_number] = account

def get\_account(self, account\_number):

return self.accounts[account\_number]

def remove\_account(self, account\_number):

self.accounts.remove(account\_number)

def get\_number\_of\_accounts(self):

return len(self.accounts)

def get\_total\_accounts\_balance(self):

total\_cdn = 0.0

for key, value in self.accounts.items():

total\_cdn += value.balance\_cdn

return total\_cdn

def deposit\_to(self, amount\_cdn, account\_num):

self.accounts[account\_num].deposit(amount\_cdn)

def print\_all\_customer\_data(self):

for key, value in self.accounts.items():

print("Customer %s has $%.2f in account #%s" % (value.member\_last\_name.title(), value.balance\_cdn, value.account\_number))

print("Total bank balance in all accounts for %s is %.2f" % (self.name, self.get\_total\_accounts\_balance()))

**main.py**

from bank\_account import BankAccount

from bank import Bank

def main():

bank = Bank("Bank of Kanada")

account1 = BankAccount(100.0, "abc111", "woods")

account2 = BankAccount(200.0, "def222", "gates")

account3 = BankAccount(300.0, "ghi333", "bezos")

account4 = BankAccount(400.0, "jkl444", "zuckerberg")

bank.add\_account(account1)

bank.add\_account(account2)

bank.add\_account(account3)

bank.add\_account(account4)

bank.deposit\_to(22.22, "def222")

bank.get\_account("abc111").transfer(5.00, bank.get\_account("ghi333"))

bank.print\_all\_customer\_data()

if \_\_name\_\_ == "\_\_main\_\_":

main()

**output**

Customer Woods has $95.00 in account #abc111

Customer Gates has $222.22 in account #def222

Customer Bezos has $305.00 in account #ghi333

Customer Zuckerberg has $400.00 in account #jkl444

Total bank balance in all accounts for Bank of Kanada is 1022.22