# CS 115 - Introduction to Programming in Python

# Lab Guide 08

**Lab Objectives:** Inheritance

**Notes:**

1. Upload your solutions as **a single .zip file** to the Lab08 assignment for your section on Moodle. You must use the following naming convention: Lab08\_Surname\_FirstName.zip where Surname is your family name and FirstName is your first name.
2. Solutions sent through email will not be accepted.
3. You should only use functionality covered in CS115 in your solution.
4. Include a docstring for your functions.
5. Create a class, **BankAccount**, with the following data attributes and methods. Note all data attributes and class variables should be private.

**Data Attributes:**

* **account number**: stores the account number of the customer.
* **balance**: current balance of the customer

**Methods**:

* **\_\_init()\_\_**: initializes the balance to 0 and account number to the parameter
* **get\_account:** returns the **account number**.
* **deposit:** Adds the specified amount to the account balance
* **withdraw :** withdraws the specified amount from the account balance if there is enough money and returns True, otherwise prints a message and returns False.
* **transfer:** transfers the given amount from this account to the given other amount if there is enough money.
* **\_\_repr()\_\_**: returns a string representation of a BankAccount object formatted as shown in the sample run.

1. Create a subclass, **SavingsAccount**, by extending the superclass **BankAccount,** with the following data attributes and methods. Note all data attributes should be private.

**Data Attributes:**

* **rate**: interest rate

**Methods**:

* **\_\_init()\_\_:** 
  + Takes the following parameters: account number, initial balance and the interest rate as parameters.
  + Initialize the BankAccount data using the super class \_\_init\_\_ method.
  + Initialize interest rate to the parameter values.
* **add\_interest:** calculates the interest and deposits the interest to the balance.

1. Write a script **BankApp** with the following functions:

* **readCustomers():** takes a string filename as a parameter and returns a list of **SavingsAccounts** objects.
* **findCustomerIndex():** takes a list of **SavingsAccounts** objects and an account number and returns the index of that account.
* **makeOperations():** takes a string filename and the list of **SavingsAccounts** objects as parameters and performs the appropriate operations on the specified object in the list.
* **The script** should do the following:
  + Creates a list of **SavingsAccounts** objects, **accounts,** by sending. ‘customer.txt’ to **readCustomers** function.
  + Call makeOperations function by sending account.txt to make the specified operations on the specified accounts in **accounts.** Call **findCustomerIndex** function where necessary.
  + Display **accounts** list.

‘customer.txt’ contains: ‘account.txt’ contains:

1456;W;500

8720;D;120

3452;W;200

1456;W;700

8720;W;300

8720;W;420

1456;T;100;8720

1456;D;150

3452;W;240

1456;D;320

3452;W;125

8720;D;740

3452;W;700

1456;D;500

8720;W;200

8720;T;200;3452

1456,1000,15

8720,500,12.5

3452,2000,18

account number

D – Deposit

W – Withdraw

T – Transfer

amount of money

transferred account number

**Sample Run:**

There is not enough money!

700 TL cannot be withdrawn from account 1456

There is not enough money!

420 TL cannot be withdrawn from account 8720

[

Account Number: 1456

Balance: 1370 TL

,

Account Number: 8720

Balance: 760 TL

,

Account Number: 3452

Balance: 935 TL

]