**CS150 Intro to CS**

**Worksheet 11: Classes**

**Problem 1**

1. Create two files:
   1. bankaccount.py
   2. lastnameWk11\_Prob1.py
2. Implement a class to represent a banking account. Your class should have the following attributes (instance variables) and methods

|  |
| --- |
| **BankAccount** |
| balance |
| \_\_init\_\_(self, balance=0)  getBalance(self)  setBalance(self, amount)  deposit(self, amount)  withdraw(self, amount)  \_\_str\_\_(self) |

* 1. \_\_init\_\_() method – will allow creation of a BankAccount object with a beginning balance
  2. getBalance() accessor method – will return the current balance
  3. setBalance() mutator method – will set the current balance
  4. deposit() method – amount to be deposited will be passed into this method parameter ‘amount’. The method should add the deposit amount to the balance
  5. withdraw() method – amount to be withdrawn will be passed into this method parameter ‘amount’. The method should determine if the balance has sufficient funds to allow for this withdrawal. If so, then subtract the withdrawal amount from the balance. If balance is less than withdrawal amount, print a message indicating insufficient funds.
  6. \_\_str\_\_() method – will determine how to display the balance. It should be formatted with a dollar sign and two decimal places. Example: **$500.00**

1. In the file (lastnameWk11\_Prob1.py) write a main function to execute.
2. Ask the user to enter a beginning balance
3. Create a BankAccount object using this beginning balance
4. User will enter amount pay to be deposited
   1. Show message indicating amount to be deposited
   2. Call method to add this amount to the balance
   3. Show current balance
5. User will enter amount to be withdrawn
   1. If sufficient funds, show message indicating amount to be withdrawn
   2. Call method to withdraw this amount from the balance
   3. Show current balance

**OUTPUT of Sample Run 1 (user input shown in bold)**

Enter beginning balance: **1000**

How much were you paid this week? **500**

$500.00 will be deposited.

Account balance: $1500.00

How much would you like to withdraw? **1200**

$1200.00 will be withdrawn.

Account Balance: $300.00

**OUTPUT of Sample Run 2 (user input shown in bold)**

Enter beginning balance: **1000**

How much were you paid this week? **500**

$500.00 will be deposited.

Account balance: $1500.00

How much would you like to withdraw? **2000**

Insufficient funds

Account Balance: $1500.00

**Problem 2**

1. Create two files:
   1. student.py
   2. lastnameWk11\_Prob2.py
2. Implement a class to represent a student. The class should have the following attributes (instance variables) and methods
   1. UML class diagram

|  |
| --- |
| **Student** |
| firstName  lastName  major  gpa |
| \_\_init\_\_(self, first=”unknown”, last=”unknown”, major=”unknown”, gpa = 0.0)  setFirstName(self, first)  setLastName(self, last)  setMajor(self, major)  setGpa(self, gpa)  getFirstName(self)  getLastName(self)  getMajor(self)  getGpa(self)  \_\_str\_\_(self) |

Note: the \_\_str\_\_() method should be a string formatted to print **one** student object formatted as shown in the sample output.

1. In the lastnameWk11\_Prob2 .py file, write functions to do the following:
   1. readStudents(file)
      1. This method will read student records from a file **studentData.txt,** create a Student object for each record, store the student objects in a list, and return the list of student objects
   2. printStudents(studentList)
      1. This method will print the student records in the list, one record per line formatted as sample output below
   3. findHighestGpa(studentList)
      1. This method will find the student in the list **studentList** with the highest gpa and return that Student object
   4. writeToFile(studentList)
      1. This method will write to a file **topStudents.txt** all the students with a gpa of 3.00 or higher. Output to the file should be one record per line in the format:

firstName lastName major gpa

* + 1. Print each student object to the screen that you are writing to the file. Output to the screen should use the \_\_str\_\_ () method to format

lastName, firstName major gpa

* 1. main ()
     1. call the readStudents() method to get information into the list of Student objects
     2. call the printStudents() method to print out the students
     3. call the findHighestGpa() method to print out the student with the highest gpa
     4. call the writeToFile() method to write to a file (‘topStudents.txt’) all students with a gpa of 3.00 or higher (one record per line)
     5. be sure to include blank lines as shown in the sample run below

**Output of Sample Run**

Smith, Amy CS 3.89

Brown, John MIS 2.99

White, Paul IDM 3.59

Black, Sue Biology 2.56

Jones, Tony Music 1.63

Henry, Allison Business 2.69

Doe, John Health 4.00

Prather, Paige History 2.57

Callaway, Colton Psychology 1.39

Thomas, Tyler PE 3.99

Miller, Mike Physics 3.75

Highest GPA:

Doe, John Health 4.00

Sending to file:

Smith, Amy CS 3.89

White, Paul IDM 3.59

Doe, John Health 4.00

Thomas, Tyler PE 3.99

Miller, Mike Physics 3.75