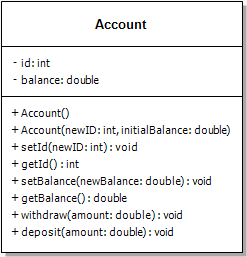
**OOP Sheet**

1. Create an employee class. The member data should comprise an int for storing the employee number and a float for storing the employee’s compensation. Member functions should allow the user to enter this data and display it. Write the employee class and a test program containing main() that allows the user to enter data for three employees and display it.
2. Create an Account class in C++ that implements this UML class diagram.



Create a main program which;

1. Requests the initial balances for two separate accounts
2. Creates two Account objects
3. Requests an amount to deposit in the first account
4. Requests an amount to withdraw from the first account
5. Displays the current balance of that account
6. Repeats steps 3-5 with the second account
7. This requires that both Account objects be active at the same time

Create Your TestProgram.cpp and Account.h files with the Account class and main program. ( separate files)

Sample Output (user input in **bold**)

Please enter the initial balance for account number C1 **567.25**

Please enter the initial balance for second account number C2 **1064.32**

How much will you deposit in account C1? **281.00**

How much will you withdraw from account C1? **193.67**

The current balance in account number C1 is 654.58

How much will you deposit in account C2? **338.34**

How much will you withdraw from account C2? **274.99**

The current balance in account C2 is 1127.67

1. Create a class called Employee that includes three pieces of information as data members—a first name (type string), a last name (type string) and a monthly salary (type int). Your class should have a constructor that initializes the three data members. Provide a set and a get function for each data member. If the monthly salary is not positive, set it to 0. Write a test program that demonstrates class Employee’s capabilities. Create two Employee objects and display each object’s yearly salary. Then give each Employee a 10 percent raise and display each Employee’s yearly salary again.
2. Create a class called Account that a bank might use to represent customers' bank accounts. Your class should include one data member of type int to represent the account balance. Your class should provide a constructor that receives an initial balance and uses it to initialize the data member. The constructor should validate the  
   initial balance to ensure that it is greater than or equal to 0. If not, the balance should be set to 0 and the constructor should display an error message, indicating that the initial balance was invalid. The class should provide three member functions. Member function credit should add an amount to the current balance. Member function debit should withdraw money from the Account and should ensure that the debit amount does not exceed the Account's balance. If it does, the balance should be left unchanged and the function should print a message indicating "Debit amount exceeded account balance." Member function getBalance should return the current balance. Create a program that creates two Account objects and tests the member  
   functions of class Account.
3. Create a class called Invoice that a hardware store may use to represent an invoice for an item sold at the store:

The class should have four (4) instance variables:

* part number : string
* part description : string (what is the item)
* quantity being purchased : int
* price per item : double

Create a constructor that initializes the four data members. Create an accessor and mutator for each data member as well.

Create a member function named getInvoiceAmount that calculates the invoice amount. (i.e., multiplies the quantity by the price per item), then returns the amount as an int value. If the quantity is not positive, it should be set to 0. If the price per item is not positive, it should be set to 0. Write a test program that demonstrates class Invoice’s capabilities.

1. Create a class called Student that includes a data member that holds a name for each object created from the class and a data member that represents the number of students. That is, the first object created will be 1, the second 2, and so on.

To do this, you’ll need another data member that records a count of how many objects have been created so far. (What keyword specifies this?) Then, as each object is created, its constructor increase the count member variable by one.

Then write a main() program that creates three objects and queries each one about their names and the count.