**National University of Computer and Emerging Sciences**



**Lab Manual 09**

**Object Oriented Programming**

|  |  |
| --- | --- |
| Course Instructor | Miss Abeeda |
| Lab Instructor (s) | Siddiqua Nayyer  Dilawar Shabbir |
| Section | A |
| Semester | Spring 2021 |

21st May, 2021

Department of Computer Science

FAST-NU, Lahore, Pakistan

**Objectives:**

* Understand and implement inheritance among various classes.
* Understand and implement function overloading and overriding

**TASK 1:**

**Function Overloading and Function Overriding:**

|  |  |
| --- | --- |
|  | Overriding means, giving a different definition of an existing function with same parameters, and overloading means adding a different definition of an existing function with different parameters.  Example:  #include <iostream>  class base{  public:  virtual void show(){std::cout<<"I am base";} //this needs to be virtual to be overridden in derived class  void show(int x){std::cout<<"\nI am overloaded";} //this is overloaded function of the previous one  };  class derived:public base{  public:  void show(){std::cout<<"I am derived";} //the base version of this function is being overridden  };  int main(){  base\* b;  derived d;  b=&d;  b->show(); //this will call the derived version  b->show(6); // this will call the base overloaded function  }  **Output:**  I am derived  I am overloaded |

**TASK 2:**

**a)** Define the class **bankAccount** to store a bank customer’s account number and balance. Suppose that account number is of type int, and balance is of type double. Your class should, at least, provide the following operations: set the account number, retrieve the account number, retrieve the balance, deposit and withdraw money, and print account information. Add appropriate constructors.

**b)** Every bank offers a checking account. Derive the class **checkingAccount** from the class bankAccount (designed in part (a)). This class **inherits members** to store the account number and the balance from the base class. A customer with a checking account typically receives interest, maintains a minimum balance, and pays service charges if the balance falls below the minimum balance. Add member variables to store this additional information. In addition to the operations inherited from the base class, this class should provide the following operations: **set interest rate, retrieve interest rate, set minimum balance, retrieve minimum balance, set service charges, retrieve service charges, post interest,** verify if the balance is less than the minimum balance, withdraw (override the method of the base class), and print account information. Add appropriate constructors.

**c)** Every bank offers a savings account. Derive the class **savingsAccount** from the class bankAccount (designed in part (a)). This class inherits members to store the account number and the balance from the base class. A customer with a savings account typically receives interest, makes deposits, and withdraws money. In addition to the operations inherited from the base class, this class should provide the following operations: set interest rate, retrieve interest rate, post interest, withdraw (override the method of the base class), and print account information. Add appropriate constructors.

Write main() to test your classes.

**TASK 3:**

In this task, you will design various classes and write a program to computerize the billing system of a hospital. Consider the following information to implement this task:

1. A **doctor** has a first name, last name, date of birth and a specialty e.g. cardiologist, ophthalmologist etc.
2. A **patient** has a patient id, first name, last name, date of birth, complete information of the doctor, hospital admit date and discharge date.
3. The system will generate a **bill** which will have the complete information of the patient and the doctor. The bill would contain all the hospital charges like the doctor fee, room charges, pharmacy charges, and nursing charges.
4. Identify all classes (hint: total number of classes are 3), their respective data members and composition relationships.
5. Add appropriate constructors and member functions for all classes to initialize, access, and manipulate their respective data members. (You can reuse the **Date** class from previous task for data members which store date values)
6. Write a program to test your classes and print the bill for a patient. The bill should contain all the patient details like patient id, patient name, admit date, discharge date, doctor name and specialty, physician fee, room charges, pharmacy charges, and nursing charges.

**Note:**

* Deallocate all dynamically allocated memory.
* Create **separate .h and .cpp** files for all classes
* Follow all the code indentation, naming conventions and code commenting guidelines.