Object Oriented Paradigm

Lab 09

Topic(s): Inheritance

IMPORTANT INSTRUCTIONS:

Please keep in mind the following points while coding. Violating any of these will result in credit deduction.

- There should be no memory leakage in your class. There should be no danglingpointers.
- Make functions, objects, variables as constant wherever possible.
- Create Default, Parameterized and Copy Constructor whether mentioned ornot.
- Create Setters and Getters for all attributes.
- Follow the appropriate naming conventions as explained in class.
- Submit your files following the submission format explained in class.

Question No. 01

You have a class Person which has the following attributes as its private member variables.

- name (char*)
- age (int)

Write a parameterized constructor to initialize the values.

You have an Employee class which is publicly inherited from the Person class. It has the following attributes:

- salary (double)
- employeeId (int)

You have another class BaseballPlayer publically inherited from the Person class. BaseballPlayer has the following attributes:

- battingAverage (double)
- totalRuns (int)

Make an object of BaseballPlayer class in main and initialize values of name, age, battingAverage and totalRuns using base initialization list. Now, make an Employee class object in main and initialize values of name, age, employeeId and salary using base initialization list. Count the total number of Employees and Baseball players and display the counts.

Question No. 02

Create a class BankAccount and two additional classes (each derived from BankAccount) named SavingsAccount and CheckingAccount. The member attributes and functions for each class are given below.

BankAccount:

- title
- accountNumber
- balance
- dateOfOpening
- deposit()
- withdraw()

SavingsAccount:

- interestRate
- calculateInterest()

CheckingAccount:

- fee (charged per transaction)

SavingsAccount class should redefine member functions withdraw and deposit so that they subtract the interest amount from the account balance whenever either transaction is performed successfully.

CheckingAccount class should redefine member functions withdraw and deposit so that they subtract the fee amount from the account balance whenever either transaction is performed successfully.

You will then test the operations of each class in your driver function to simulate the transactions of both the checking account and the savings account. Maintain a count of the total number of Savings and Checking accounts and display the counts.