University of Central Punjab 

**Faculty of Information Technology**

# Object Oriented Programming (C8)

|  |  |
| --- | --- |
| **Lab 11** |  |
| **Name:** M. Tayyeb Shahzad Butt | |

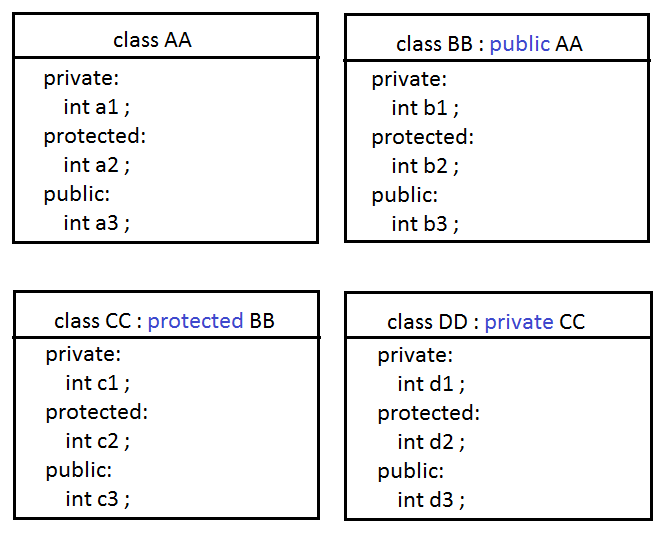
**Instructions:**

* Indent your code.
* Comment your code.
* Use meaningful variable names.
* Plan your code carefully on a piece of paper before you implement it.

**If the submissions are NOT on the portal, we will not mark them**

**Task 1:**

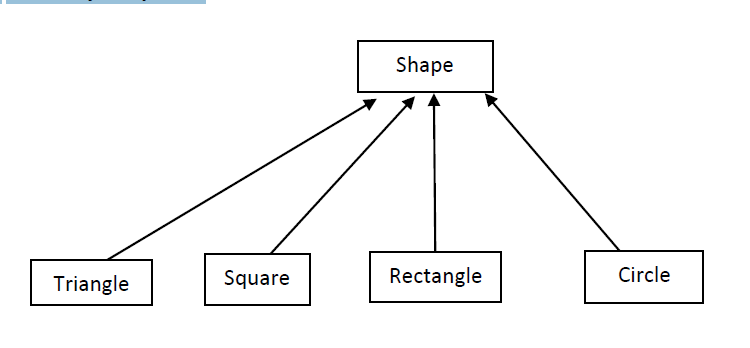
Keeping in mind the given inheritance class hierarchy, answer the questions asked below:

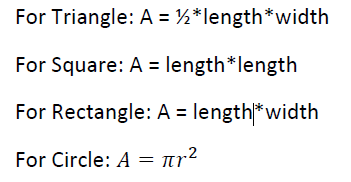


|  |  |
| --- | --- |
|  | **Access**  *(private, protected, public, inaccessible)* |
| Access of a1 inside class BB | *Inaccessible* |
| Access of a1 inside class CC | *Inaccessible* |
| Access of a1 inside class DD | *Inaccessible* |
| Access of a2 inside class BB | *Protected* |
| Access of a2 inside class CC | *Protected* |
| Access of a2 inside class DD | *Private* |
| Access of a3 inside class BB | *Public* |
| Access of a3 inside class CC | *Protected* |
| Access of a3 inside class DD | *Private* |
| Access of b1 inside class CC | *Inaccessible* |
| Access of b1 inside class DD | *Inaccessible* |
| Access of b2 inside class CC | *Protected* |
| Access of b2 inside class DD | *Private* |
| Access of b3 inside class CC | *Protected* |
| Access of b3 inside class DD | *Private* |
| Access of c1 inside class DD | *Inaccessible* |
| Access of c2 inside class DD | *Private* |
| Access of c3 inside class DD | *Private* |
| **Statement in main()** | **Access**  *(accessible OR inaccessible)* |
| objectOfAA.a1 ; | *inaccessible* |
| objectOfAA.a2 ; | *inaccessible* |
| objectOfAA.a3 ; | *Accessible* |
|  |  |
| objectOfBB.a1 ; | *inaccessible* |
| objectOfBB.a2 ; | *inaccessible* |
| objectOfBB.a3 ; | *Accessible* |
| objectOfBB.b1 ; | *inaccessible* |
| objectOfBB.b2 ; | *inaccessible* |
| objectOfBB.b3 ; | *Accessible* |
|  |  |
| objectOfCC.a1 ; | *inaccessible* |
| objectOfCC.a2 ; | *inaccessible* |
| objectOfCC.a3 ; | *inaccessible* |
| objectOfCC.b1 ; | *inaccessible* |
| objectOfCC.b2 ; | *inaccessible* |
| objectOfCC.b3 ; | *inaccessible* |
| objectOfCC.c1 ; | *inaccessible* |
| objectOfCC.c2 ; | *inaccessible* |
| objectOfCC.c3 ; | *Accessible* |
|  |  |
| objectOfDD.a1 ; | *inaccessible* |
| objectOfDD.a2 ; | *inaccessible* |
| objectOfDD.a3 ; | *inaccessible* |
| objectOfDD.b1 ; | *inaccessible* |
| objectOfDD.b2 ; | *inaccessible* |
| objectOfDD.b3 ; | *inaccessible* |
| objectOfDD.c1 ; | *inaccessible* |
| objectOfDD.c2 ; | *inaccessible* |
| objectOfDD.c3 ; | *inaccessible* |
| objectOfDD.d1 ; | *inaccessible* |
| objectOfDD.d2 ; | *inaccessible* |
| objectOfDD.d3 ; | *Accessible* |

**Task 2:**

Implement the following class hierarchy. Write a function ***area()*** to calculate the area of each object of any **Shape**.





* All the attributes MUST be pointers.
* Make a parameterized constructor with default arguments that take parameters required in each class and use the base initializer list to call the parameterized constructor with default arguments of Base class
* Make a copy constructor and use the base initializer list to call the copy constructor of Base class. (Remember you can call constructors of the base class through initializer lists; this rule applies to copy constructors as well).
* Further, write a display function as well.

Do submit the screenshot of the output with your code.

**Task 3:**

Create a base class **Card** with the following attributes

* Card number : private
* Owner name : protected
* Expiry date : public

Derive the following classes from **Card**, with mentioned additional attributes

• **Calling card** (public inheritance)

* Amount : private
* Company name : private
* PIN : private

• **ID card** (protected Inheritance)

* CNIC Number : private
* Age : private

• **Driving license card** (private Inheritance)

* Driving license type (heavy, light, bike) : private
* Issued in city : private

Your tasks:

1. In the derived classes, write the getters and setters of every member variable (including the derived variables). You are not allowed to make any member function in the base class. A main() is required to add up some a card of each type, and then to display their information. The object of the base class will not be instantiated.
2. According to the rules of inheritance, clearly specify (by adding comments in the derived classes) which of the members are inherited and clearly mention their access specifiers in the derived classes.
3. Draw the UML diagram of the above-mentioned scenario in any software of your choice.

**Task 4:**

Create a base class, called BankAccount, and two additional classes (each derived from BankAccount), called SavingsAccount and CheckingAccount.

BankAccount:

* + Title
  + AccountNumber
  + Balance
  + Deposit()
  + Withdraw()

SavingAccount:

* InterestRate
* CalculateInterest()

CheckingAccount:

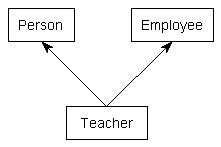
* fee charged per transaction

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

You will then test the operations of each class in function main() to simulate the transactions of both a checking account and a savings account.

**Task 5:**

Provide the C++ implementation of the following.



The Person class has name and age as its attributes. It has an overloaded constructor to initialize the values and appropriate accessor and mutator methods. The Employee class has the name of the employer and wage as its attributes with an overloaded constructor and appropriate accessor and mutator methods.

The Teacher class is inherited from the Person and Employee class with an attribute of Pay Scale of type integer. It has an overloaded constructor, appropriate accessor, mutator methods, and a display function to print the Name, Age, Name of Employer, Wage, and Pay Scale of the Teacher.