# Description

The purpose of this lab is to check your understanding of the content in lecture 6a.

We will focus on the material from chapter 11 of the book. The objectives are:

* To create classes by inheriting from existing classes.
* The notions of base classes and derived classes and the relationships between them.
* The protected member access specifier.
* The use of constructors and destructors in inheritance hierarchies.
* The order in which constructors and destructors are called in inheritance hierarchies.
* The differences between public, protected, and private inheritance.
* To use inheritance to customize existing software.
* To create classes by inheriting from existing classes.

# Part 1: Questions

Complete the assignment in the module *Topic 6a Lab: Object-oriented Programming*, "Lab 6a Questions", in Canvas.

# Part 2: Activity

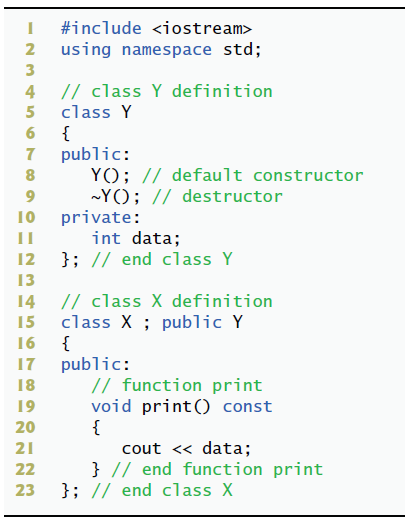
This part of the lab should be submitted using the Lab 6a Activity assignment in the *Topic 6a Lab: Object-oriented Programming* module.

**You should complete the non-programming parts of this lab using this document.** **It is possible that some of the images and answer-boxes below might move or need to be resized while using them. Do your best to make your final document neat and organized.**

## Problem 1: Correct the Code

For each of the given program segments, determine if there is an error in the code. If there is an error, specify whether it is a logic or compilation error, circle the error in the program, and write the corrected code in the space provided after each problem. If the code does not contain an error, write “no error.” [Note: It is possible that a program segment may contain multiple errors.]

1. Class X inherits from class Y.



**Answer: The issue is that data is a private member in class Y, therefore class X cannot have access to the data member normally. To rectify this we can create a function to access data in class Y which class X inherits.**

**class Y {**

**public:**

**Y();**

**~Y();**

**int getData() const;**

**private:**

**int data;**

**};**

**class X : public Y {**

**public:**

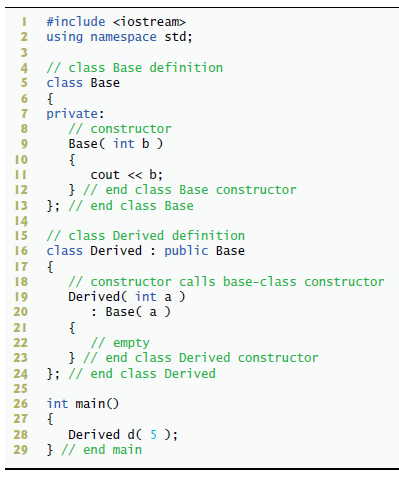
**void print() const {**

**std::cout << getData()**

**}**

**};**

1. The following code should construct a Derived object.



**Answer: The issue is that the Derived class cannot access the Base class’s constructor because it is in private, therefore it cannot access the constructor through normal means. We should set the constructor as public so that the Derived class may have access to it. We should also set the Derived constructor to public as well, because by default classes will have private members unless otherwise stated.**

**class Base {**

**public:**

**Base(int b) {**

**std::cout << b;**

**}**

**};**

**class Derived : public Base {**

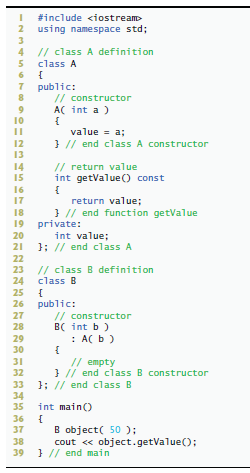
**public:**

**Derived(int a) : Base(a) {**

**}**

**};**

1. The following code creates an object of type B. Class B inherits from class A.



**Answer: The issue with this code is that class B never actually inherits class A because we never specify it to do so. We need to add a colon following class B’s declaration and follow it with public A so that it inherits the members of class A**

**class B : public A {**

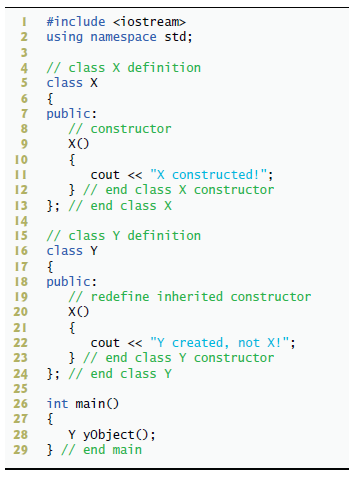
**public:**

**B(int b) : A(b) {**

**}**

**};**

1. The following code creates an object of type Y. Class Y inherits from class X.



**Answer: There are two issues with this code. Once again class Y does not actually inherit class X because the code did not specify so. Another issue is that derived classes do not inherit the base class’s constructors, however they may call upon them. The derived class thus needs its own constructor, but it may call the base class’s constructor.**

**class Y : public X {**

**public:**

**Y() : X() {**

**std::cout << “Y created, not X!”;**

**}**

**};**

### Problem 2: Programming

This lab was designed to reinforce programming concepts from chapter 11. You must create each of your classes in two files, a ".h" and a ".cpp" file. **Your classes should not include *iostream* because they do not print anything to the console nor do they handle user input.**

**Included with this assignment is the *account.h* file**.

#### Problem Description

Create an inheritance hierarchy that a bank might use to represent customers’ bank accounts. All customers at this bank can deposit (i.e., credit) money into their accounts and withdraw (i.e., debit) money from their accounts. More specific (derived) types of accounts also exist. Savings accounts, for instance, earn interest on the money they hold. Checking accounts, on the other hand, charge a fee per transaction (i.e., credit or debit).

Create an inheritance hierarchy containing base class Account and derived classes SavingsAccount and CheckingAccount that inherit from class Account.

* Base class Account should include one data member of type double to represent the account balance.
* The 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 0.0. If not, the constructor should throw an *invalid\_argument* exception 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 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 throw an *out\_of\_range* exception.
  + Member function getBalance should return the current balance.

Derived class SavingsAccount should inherit the functionality of an Account

* It should also include a data member of type double indicating the interest rate (percentage) assigned to the Account.
* SavingsAccount’s constructor should receive the initial balance, as well as an initial value for the SavingsAccount’s interest rate.
* SavingsAccount should provide a public member function calculateInterest that returns a double indicating the amount of interest earned by an account.
* Member function calculateInterest should determine this amount by multiplying the interest rate by the account balance. [Note: SavingsAccount should inherit member functions credit and debit as is without redefining them.]

Derived class CheckingAccount should inherit from base class Account

* It should include an additional data member of type double that represents the fee charged per transaction.
* CheckingAccount’s constructor should receive the initial balance, as well as a parameter indicating a fee amount.
* Class CheckingAccount should redefine member functions credit and debit so that they subtract the fee from the account balance whenever either transaction is performed successfully.
* CheckingAccount’s versions of these functions should invoke the base-class Account version to perform the updates to an account balance.
* CheckingAccount’s debit function should charge a fee only if money is actually withdrawn (i.e., the debit amount does not exceed the account balance). [Hint: Define Account’s debit function so that it returns a bool indicating whether money was withdrawn. Then use the return value to determine whether a fee should be charged.]

**All the header files that you define must include pre and post condition comments as demonstrated in the *account.h* file.**

After defining the classes in this hierarchy, **write a program called "bank.cpp"** that creates objects of each class and tests their member functions. Add interest to the SavingsAccount object by first invoking its calculateInterest function, then passing the returned interest amount to the object’s credit function. Test that, those functions which are supposed to throw exceptions, do throw exceptions.

# What to Submit for Lab 6a Activity

* This lab document completed.
* Header (.h) and source (.cpp) files for each class.
* bank.cpp file

A screenshot of the *bank.cpp* test program's output when you run it. Be sure to include the entire Visual Studio Code window. Do not include your desktop or anything else in the image. Do not take a picture with a camera or phone, use your computer to create a screenshot.