CS224 Object Oriented Programming and Design Methodologies Lab Manual

Lab 08 - Rule of 3, Inheritence

Department of Computer Science

Dhanani School of Science and Engineering

Habib University

Fall 2025

Contents

1 Introduction

1.1 Guidelines

1. Use of AI is strictly prohibited. This is not limited to the use of AI tools for code generation, debugging, or any form of assistance. If detected, it will result in immediate failure of the lab, student will be awarded 0 marks, reported to the academic integrity board and appropriate disciplinary action will be taken.
2. Absence in lab regardless of the submission status will result in 0 marks.
3. All assignments and lab work must be submitted by the specified deadline on Canvas. Late submissions will not be accepted.

1.2 Objectives

Following are the lab objectives of this lab:

- Understand and apply the Rule of Three in C++.
- To know how to inherit classes in C++.
- To practice different examples of making class hierarchies in C++.

1.3 Directory Structure

Labs will have following directory structure:

```
lab
├── manual
│   └── lab.pdf
├── src
│   ├── *.hpp
│   └── *.cpp
├── tests (optional)
│   └── *.cpp
└── makefile (optional)
```

`manual` will contain the lab manual pdf. `src` will contain the source code files, and `tests` (if present) will contain the test files. `makefile` (if present) will contain the makefile for testing and running.

1.4 Rule fo Three

In C++, the Rule of Three states that if a class defines any one of the following:

- A destructor,
- A copy constructor, or
- A copy assignment operator,

then it should probably explicitly define all three.

1.5 Inheritence

Inheritance in C++ allows one class (called a derived class) to acquire the properties and behaviors (data members and member functions) of another class (called a base class). It's a core concept of Object-Oriented Programming (OOP) that promotes code reusability, extensibility, and hierarchical relationships.

Let's look at an example:

```cpp
#include <iostream>
using namespace std;
class Animal {
public:
    virtual void sound() { cout << "Some sound" << endl; }
};

class Dog : public Animal {
public:
    void sound() override { cout << "Woof!" << endl; }
};

class Cat : public Animal {
public:
    void sound() override { cout << "Meow!" << endl; }
};


int main() {
    Animal* a1 = new Dog();
    Animal* a2 = new Cat();
    a1->sound(); // Woof!
    a2->sound(); // Meow!
}
```

Listing 1.1: `inheritence_recap.cpp`

Inheritance allows:

- Code Reusability. You don't have to rewrite code for common functionality. The derived class inherits existing functionality and can add or modify behavior.
- Extensibility. You can extend an existing class by adding new features instead of rewriting everything.
- Polymorphism (Dynamic Behavior). Inheritance allows runtime polymorphism using virtual functions. It enables different classes to be treated as one type (the base class) but behave differently.

2 Exercises

2.1 Rule of Three in LinkedList [20 Points]

This question will help you understand and apply the Rule of Three in C++. You will design a LinkedList class that correctly manages dynamically allocated memory using:

- A Destructor
- A Copy Constructor
- A Copy Assignment Operator

Implement a singly linked list that:

- Stores integer values.
- Allows insertion at the end.
- Supports deep copying between list objects.
- Properly cleans up memory when the object is destroyed.

You are provided with a skeleton code in source folder. Complete all parts marked with // TODO.

This should be the expected output:

```
Original List (list1): 10 20 30
Copied List (list2): 10 20 30
Modified list2: 10 20 30 40
Original list1 (should be unchanged): 10 20 30
Assigned List (list3 = list1): 10 20 30
```

2.2 Publishing Company [40 Points]

Imagine a publishing company that markets both book and audio cassette versions of its works. Create a class `publication` that stores the title (`string`) and price (`float`) of a publication.
From this class derive two classes:

- `book`, which adds a `page_count` (`int`).
- `tape`, which adds a `playing_time` in minutes (`float`).

Each of these three classes should have a function `getdata()` to get the data from the input stream and `putdata()` function to display its data to the output stream.

Sample Case 1

```
>>> The Fellowship of the Ring
>>> 5000
>>> 432
>>> Guardians of the Galaxy
>>> 500
>>> 60
Publication title: The Fellowship of the Ring
Publication price: 5000
Book page count: 432
Publication title: Guardians of the Galaxy
Publication price: 500
Tape's playing time: 60
```

Go through the skeleton code of q2 and look at main function to understand which functions to complete.

2.3 Payment Class [40 Points]

You are required to implement following classes:

2.3.1 Payment Class

Define a class named `Payment` that contains:

- a private member variable of type double that stores the amount of the payment and appropriate accessor and mutator methods.
- a method named `paymentDetails()` that outputs an English sentence that describes the amount of the payment.

2.3.2 Cash Payment Class

Define a class named `CashPayment` that is derived from `Payment`.

- This class should redefine the `paymentDetails()` method to indicate that the payment is in cash. Include appropriate constructor(s).

2.3.3 Credit Card Payment Class

Define a class named `CreditCardPayment` that is derived from `Payment`.

- This class should contain member variables for the name on the card, expiration date, and credit card number. Include appropriate constructor(s).
- Redefine the `paymentDetails()` method to include all credit card information in the printout.

Go through the skeleton code of q3 and look at main function to understand which functions to complete.

This is the expected output of main function given in skeleton code.

```
Details of Cash #1...
Amount of cash payment: 75.25

Details of Cash #2...
Amount of cash payment: 36.95

Details of Credit Card #1...
Amount of credit card payment: 95.15
Name on the credit card: Smith
Expiration date: 12/21/2009
Credit card number: 321654987

Details of Credit Card #2...
Amount of credit card payment: 45.75
Name on the credit card: James
Expiration date: 10/30/2008
Credit card number: 963852741
```