National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department Of Electrical Engineering and Computer Sciences

Instructor: Mehreen Tahir **Date:** 20th November 2023

Lab Engineer: Mehwish Kiran **Time:** 10:00am – 12:50pm

CS 212: Object Oriented Programming

Lab 08: Inheritance and Composition

Information	Description
Name:	Irfa Farooq
CMS ID:	412564
Class:	BEE-14
Section:	D

Task 1: Use concepts of Inheritance and Composition to implement the tasks at hand.

NoteBook.h:

```
#include <iostream>

class NoteBook {
private:
    int manufacturerID;
    std::string manufacturerName;
public:
    NoteBook();
    void set_ID(int);
    int get_ID();
    void set_name(std::string);
    std::string get_name();
    ~NoteBook();
};
```

ENoteBook.h:

```
#include <iostream>

class ENoteBook:protected NoteBook {
private:
    int size;
public:
    ENoteBook();
    void set_size(int);
    int get_size();
    void Display(int, std::string);
    ~ENoteBook();
};
```

PaperNoteBook.h:

```
#include <iostream>

class PaperNoteBook:protected NoteBook {
private:
    int TotalPages;
public:
    PaperNoteBook();
    void set_pages(int);
    int get_pages();
    void Display(int, std::string);
    ~PaperNoteBook();
};
```

NoteBook_Implementation.cpp:

```
#include <iostream>
#include "NoteBook.h"

NoteBook::NoteBook() {
    manufacturerID = 0;
    manufacturerName = "NULL";
    std::cout << "Note Book Constructed." << std::endl;
}
void NoteBook::set_ID(int x) {
    manufacturerID = x;
}
int NoteBook::get_ID() {
    return manufacturerID;
}
void NoteBook::set_name(std::string n) {
    manufacturerName = n;
}
std::string NoteBook::get_name() {
    return manufacturerName;
}
NoteBook::~NoteBook(){
    std::cout << "Note Book Destructed." << std::endl;
}
```

ENoteBook_Implementation.cpp:

```
#include <iostream>
#include "NoteBook.h"
#include "ENoteBook.h"

ENoteBook::ENoteBook() {
    std::cout << "E Note Book constructed." << std::endl;
}
void ENoteBook::set_size(int s) {
    size = s;
}
int ENoteBook::get_size() {
    return size;
}
void ENoteBook::Display(int x, std::string y){
    set_ID(x);
    std::cout << "Manufacturing ID: " << get_ID() << std::endl;
    set_name(y);
    std::cout << "Manufacturer name: " << get_name() << std::endl;
    std::cout << "Size of E Note Book: " << get_size() << std::endl;
}
ENoteBook::~ENoteBook(){
    std::cout << "E Note Book Destructed." << std::endl;
}
```

PaperNoteBook_Implementation.cpp:

```
#include <iostream>
#include "NoteBook.h"
#include "PaperNoteBook.h"

PaperNoteBook::PaperNoteBook() {
    std::cout << "Paper Note Book Constructed." << std::endl;
}
void PaperNoteBook::set_pages(int p) {
    TotalPages = p;
}
int PaperNoteBook::get_pages() {
    return TotalPages;
}
void PaperNoteBook::Display(int x, std::string y) {
    set_ID(x);
    std::cout << "Manufacturing ID: " << get_ID() << std::endl;
    set_name(y);
    std::cout << "Manufacturer name: " << get_name() << std::endl;
    std::cout << "Total pages in Paper Note Book: " << get_pages() <<
std::endl;
}
PaperNoteBook::~PaperNoteBook() {
    std::cout << "Paper Note Book Destructed." << std::endl;
}
```

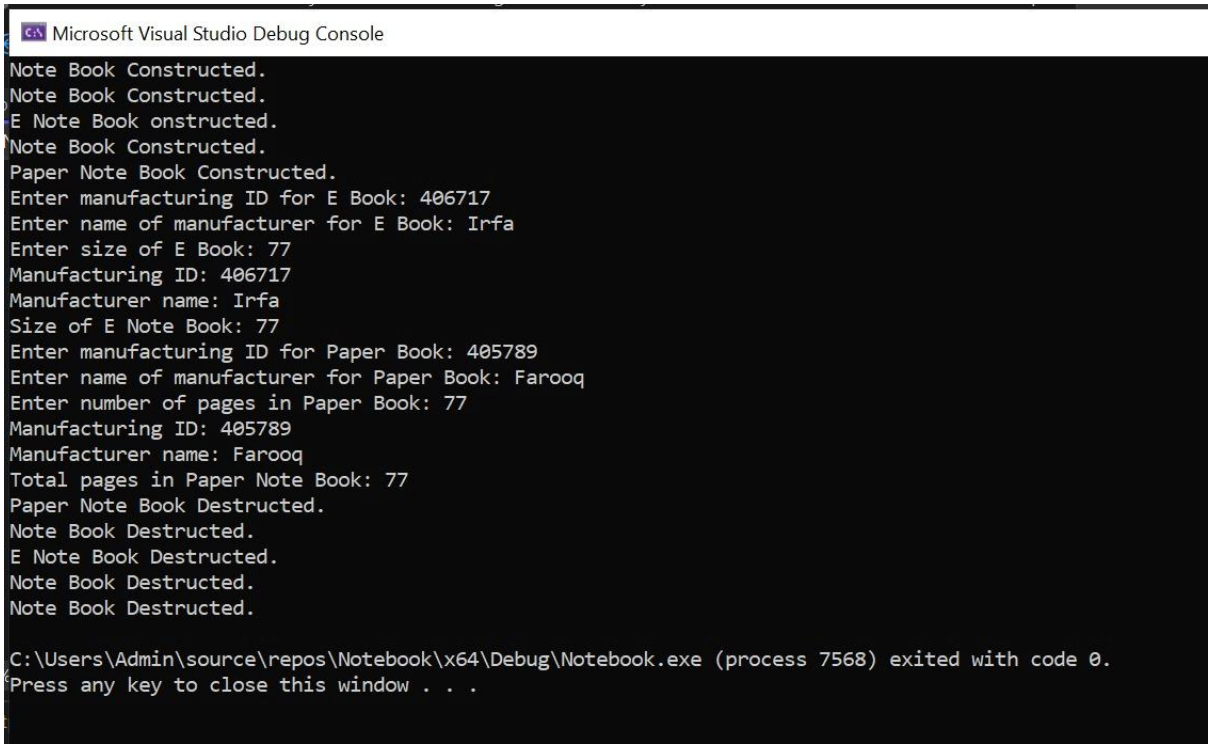
Main.cpp:

```
#include <iostream>
#include "NoteBook.h"
#include "ENoteBook.h"
#include "PaperNoteBook.h"

int main() {
    NoteBook Book;
    ENoteBook EBook;
    PaperNoteBook PBook;
    int ID, size;
    std::string name;
    std::cout << "Enter manufacturing ID for E Book: ";
    std::cin >> ID;
    std::cout << "Enter name of manufacturer for E Book: ";
    std::cin >> name;
    std::cout << "Enter size of E Book: ";
    std::cin >> size;
    EBook.set_size(size);
    EBook.Display(ID, name);
```

```
    int id, pages;
    std::string Name;
    std::cout << "Enter manufacturing ID for Paper Book: ";
    std::cin >> id;
    std::cout << "Enter name of manufacturer for Paper Book: ";
    std::cin >> Name;
    std::cout << "Enter number of pages in Paper Book: ";
    std::cin >> pages;
    PBook.set_pages(pages);
    PBook.Display(id, Name);
}
```

Output Screenshots



Task 2: Using Inheritance and Composition.

Course.h:

```
#include <iostream>

class course {
private:
    int courseNumber, courseHours;
public:
    void setCourse (int, int);
    int getCourseNumber();
    int getCourseHours();
};
```

Section.h:

```
#include <iostream>
#include "Course.h"

class section {
private:
    int secNumber, secCourse_num, secCourse_hrs;
    course c;
public:
    void setSection (int, int, int);
    int getSecNumber();
    int getCourse_num();
    int getCourse_hrs();
};
```

Course.cpp:

```
#include <iostream>
#include "course.h"

void course::setCourse(int x, int y) {
    courseNumber = x;
    courseHours = y;
}
int course::getCourseNumber(){
    return courseNumber;
}
int course::getCourseHours(){
    return courseHours;
}
```

Section.cpp:

```
#include <iostream>
#include "section.h"
#include "course.h"

void section::setSection (int x, int y, int z) {
    secCourse_num = x;
    secCourse_hrs = y;
    secNumber = z;
}
int section::getSecNumber(){
    return secNumber;
}
int section::getCourse_num(){
    return secCourse_num;
}
```

```
int section::getCourse_hrs(){
    return secCourse_hrs;
}
```

Main.cpp:

```
#include <iostream>
#include "course.h"
#include "section.h"

int main() {
    course C;
    section *S = new section [7];
    for (int i = 0; i < 7; i++){
        S[i].setSection(117, 3, i+1);
        std::cout << "Section Number: " << S[i].getSecNumber() << std::endl;
        std::cout << "Course Hours: " << S[i].getCourse_hrs() << std::endl;
        std::cout << "Course Number: " << S[i].getCourse_num() << std::endl;
    }
}
```

Output Screenshots

```
Output

/tmp/R90859iDXQ.o
Section Number: 1
Course Hours: 3
Course Number: 117
Section Number: 2
Course Hours: 3
Course Number: 117
Section Number: 3
Course Hours: 3
Course Number: 117
Section Number: 4
Course Hours: 3
Course Number: 117
Section Number: 5
Course Hours: 3
Course Number: 117
Section Number: 6
Course Hours: 3
Course Number: 117
Section Number: 7
Course Hours: 3
Course Number: 117
```

Conclusion:

In this lab, we were able to identify the order of object development in an inherited class. The base class constructors are called even when the child classes are called and are destroyed in the reverse order of their creation. Other than that, we were also able to understand the concepts of composition in inheritance.