

CC-211L

Object Oriented Programming

Laboratory 10

Inheritance - I

Version: 1.0.0

Release Date: 17-03-2023

Department of Information Technology

University of the Punjab

Lahore, Pakistan

Contents:

- Learning Objectives
- Required Resources
- General Instructions
- Background and Overview
 - Inheritance
 - Base Class
 - Derived Class
- Activities
 - Pre-Lab Activity
 - Base and Derived Classes
 - Task 01
 - Task 02
 - In-Lab Activity
 - Relationship between Base and Derived Classes
 - Explanation with an Example
 - Task 01
 - Task 02
 - Post-Lab Activity
 - Task 01
- Submissions
- Evaluations Metric
- References and Additional Material
- Lab Time and Activity Simulation Log

Learning Objectives:

- Base Class
- Derived Class
- Relationships between Base and Derived Class

Resources Required:

- Desktop Computer or Laptop
- Microsoft ® Visual Studio 2022

General Instructions:

- In this Lab, you are **NOT** allowed to discuss your solution with your colleagues, even not allowed to ask how is s/he doing, this may result in negative marking. You can **ONLY** discuss with your Teaching Assistants (TAs) or Lab Instructor.
- Your TAs will be available in the Lab for your help. Alternatively, you can send your queries via email to one of the followings.

Background and Overview:

Inheritance:

Inheritance is one of the core concepts of object-oriented programming (OOP) languages. It is a mechanism where you can derive a class from another class for a hierarchy of classes that share a set of attributes and methods.

Base Class:

A base class is an existing class from which the other classes are determined, and properties are inherited. It is also known as a superclass or parent class. In general, the class which acquires the base class can hold all its members and some further data as well.

Derived Class:

A derived class is a class that is constructed from a base class or an existing class. It tends to acquire all the methods and properties of a base class. It is also known as a subclass or child class.

Activities:

Pre-Lab Activities:

Base and Derived Classes:

A base class is a class in Object-Oriented Programming language, from which other classes are derived. The class which inherits the base class has all members of a base class as well as can also have some additional properties. The Base class members and member functions are inherited to Object of the derived class. A base class is also called parent class or superclass.

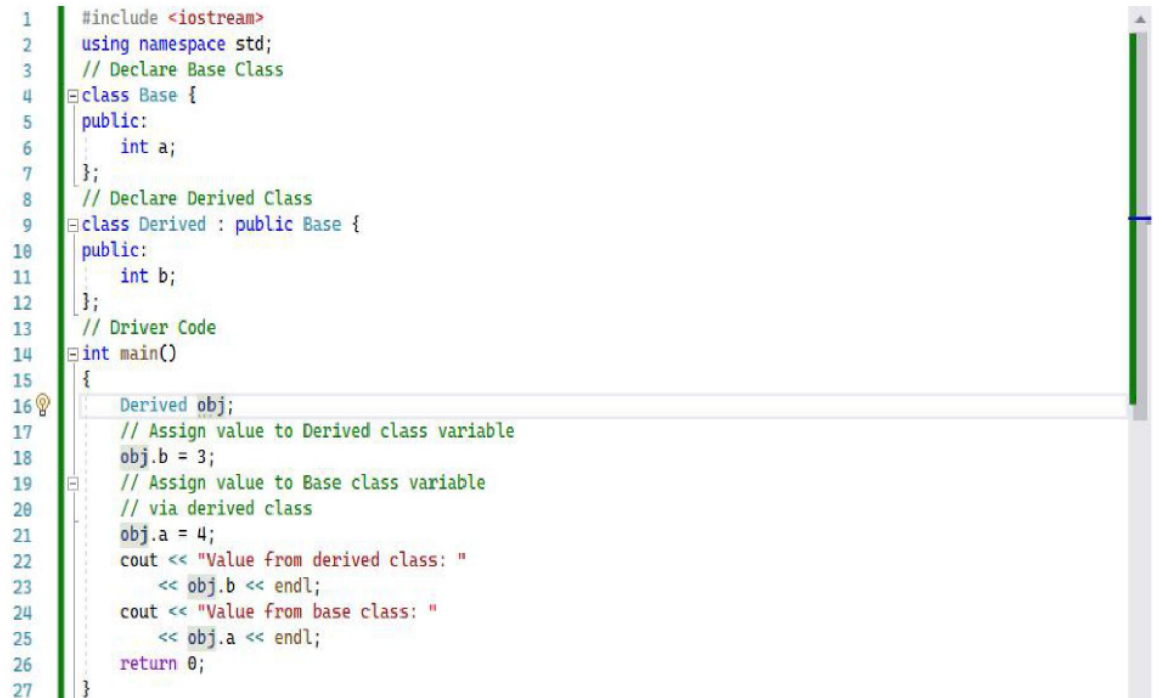
A class that is created from an existing class. The derived class inherits all members and member functions of a base class. The derived class can have more functionality with respect to the Base class and can easily access the Base class. A Derived class is also called a child class or subclass.

Syntax for Derive Class:

```
class BaseClass{
    // members....
    // member function
}

class DerivedClass : public BaseClass{
    // members....
    // member function
}
```

Example:



```
1  #include <iostream>
2  using namespace std;
3  // Declare Base Class
4  class Base {
5  public:
6      int a;
7  };
8  // Declare Derived Class
9  class Derived : public Base {
10 public:
11     int b;
12 };
13 // Driver Code
14 int main()
15 {
16     Derived obj;
17     // Assign value to Derived class variable
18     obj.b = 3;
19     // Assign value to Base class variable
20     // via derived class
21     obj.a = 4;
22     cout << "Value from derived class: "
23          << obj.b << endl;
24     cout << "Value from base class: "
25          << obj.a << endl;
26     return 0;
27 }
```

Fig. 01 (Base and Derived Classes)

Output:A screenshot of the Microsoft Visual Studio Debug Console window. The window has a title bar with the text "Microsoft Visual Studio Debug Console" and standard window control buttons (minimize, maximize, close). The console area is black with white text. It displays two lines of output: "Value from derived class: 3" and "Value from base class: 4".

```
Microsoft Visual Studio Debug Console
Value from derived class: 3
Value from base class: 4
```

Fig. 02 (Base and Derived Classes)

Task 01: Vehicles**[Estimated time 15 minutes / 10 marks]**

- Create a base class Vehicles with a data member: maxSpeed, number of wheels, number of doors, number of seats, price, fuel type, model, color. Create setters getters for this class and a display function.
- Create derived classes Truck with Data members: load capacity.
- Create derived classes Car with Data members: useType (family, personal, Commercial, racing).
- Create derived classes MotorCycle with Data members: engineOption (petroleum or electric).
- Create an instance of each derived class in Main function, set their attributes and call the display functions (for Parent class attributes, use Parent class display function).

Task 02: Shape**[Estimated time 15 minutes / 10 marks]**

Create a parent class Shape with attributes length, width.

Create setters, getters and a display function.

Create derived class Rectangle, create setters getters, create a function in it to display length, width and the area of the rectangle.

Create derived class Square, create setters getters, create a function in it to display length, width and the area of the rectangle.

Task 03: Animals

Create two classes named Mammals and Marine Animals. Now, create a function in each of these classes which prints "I am mammal", "I am a marine animal" respectively. Think of 3 attributes for each class. Now you will have to create derived classes for the following Animals. Create a display function in these classes and a function to make sound of that animal.

Lion

- Dolphin
- Whale (Whale is both marine and mammal so consider)
- Cow
- Shark

Task 04: Concept of Inheritance

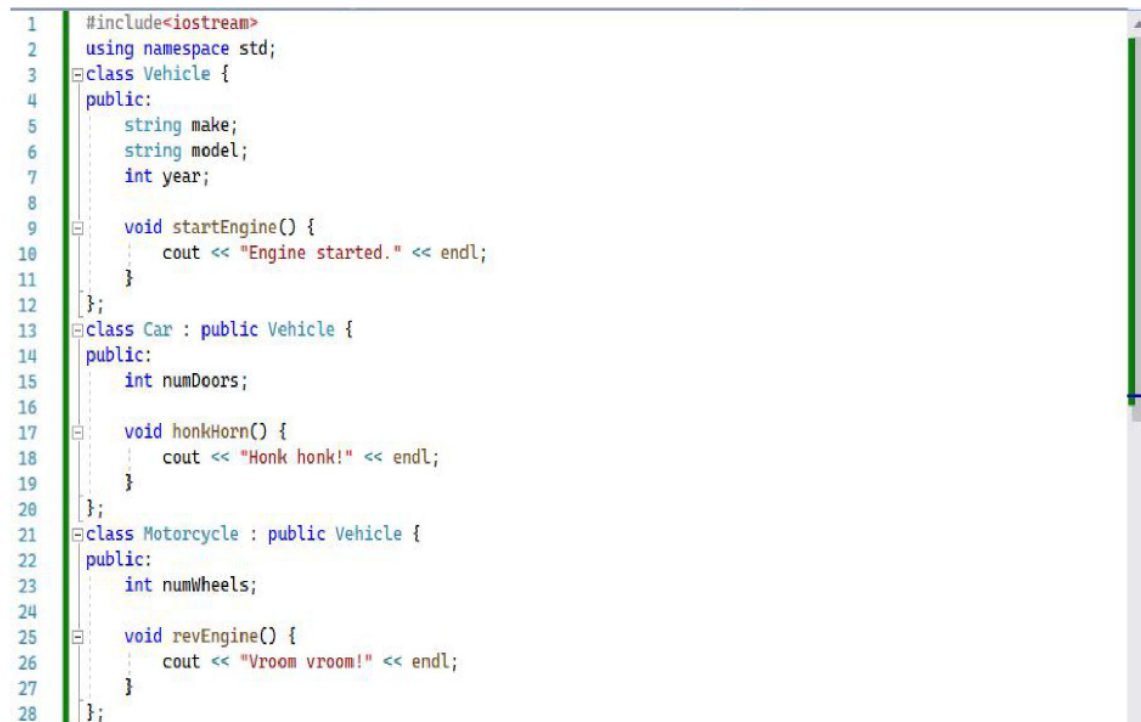
Explain Concept of Inheritance in your own words (just 1 paragraph).

In-Lab Activities:

Relationships between Based and Derived Classes:

The relationship between base and derived classes is established through inheritance, where a derived class can inherit data members and member functions from its base class. Here's an example to illustrate this concept:

Example:



```
1  #include<iostream>
2  using namespace std;
3  class Vehicle {
4  public:
5      string make;
6      string model;
7      int year;
8
9      void startEngine() {
10         cout << "Engine started." << endl;
11     }
12 };
13 class Car : public Vehicle {
14 public:
15     int numDoors;
16
17     void honkHorn() {
18         cout << "Honk honk!" << endl;
19     }
20 };
21 class Motorcycle : public Vehicle {
22 public:
23     int numWheels;
24
25     void revEngine() {
26         cout << "Vroom vroom!" << endl;
27     }
28 };
```

Fig. 03 (Inheritance)

Explanation:

In the above example, 'Car' and 'Motorcycle' are both derived classes of Vehicle. They inherit the data members 'make', 'model', and 'year', as well as the member function 'startEngine()'. This means that 'Car' and 'Motorcycle' can use these members as if they were defined in their own class.

For instance, a 'Car' object can have a 'make', 'model', 'year', and 'numDoors' data member. It can also use the 'startEngine()' member function to start the engine. Additionally, it can use the 'honkHorn()' member function that is unique to the 'Car' class.

Similarly, a 'Motorcycle' object can have a 'make', 'model', 'year', and 'numWheels' data member. It can also use the 'startEngine()' member function to start the engine. Additionally, it can use the 'revEngine()' member function that is unique to the 'Motorcycle' class.

Overall, inheritance is a powerful feature in object-oriented programming that allows for code reuse and organization. It can help to reduce redundancy and improve the clarity of code by allowing classes to inherit properties and behavior from other classes.

Explanation with an Example:

Let's say we want to create a program for managing bank accounts. We can start by defining a base class called '**BankAccount**' that has the basic properties and behaviors of a bank account:

```

4 class BankAccount {
5 public:
6     BankAccount(double balance) : m_balance(balance) {}
7
8     double getBalance() const { return m_balance; }
9     void deposit(double amount) { m_balance += amount; }
10    void withdraw(double amount) { m_balance -= amount; }
11
12 private:
13     double m_balance;
14 };

```

Fig. 04 (Inheritance Explanation)

In this example, '**BankAccount**' has a '**balance**' data member, which represents the current balance of the account. It also has member functions to get the balance, deposit money, and withdraw money from the account.

Now let's say we want to create two types of bank accounts: a savings account and a checking account. We can create two derived classes, '**SavingsAccount**' and '**CheckingAccount**', that inherit from '**BankAccount**':

```

16 class SavingsAccount : public BankAccount {
17 public:
18     SavingsAccount(double balance, double interestRate)
19         : BankAccount(balance), m_interestRate(interestRate) {}
20
21     double calculateInterest() const {
22         return getBalance() * m_interestRate / 100.0;
23     }
24
25 private:
26     double m_interestRate;
27 };
28
29 class CheckingAccount : public BankAccount {
30 public:
31     CheckingAccount(double balance, double monthlyFee)
32         : BankAccount(balance), m_monthlyFee(monthlyFee) {}
33
34     void deductMonthlyFee() {
35         withdraw(m_monthlyFee);
36     }
37
38 private:
39     double m_monthlyFee;
40 };

```

Fig. 05 (Inheritance Explanation)

'**SavingsAccount**' and '**CheckingAccount**' both inherit the balance data member and the '**getBalance()**', '**deposit()**', and '**withdraw()**' member functions from '**BankAccount**'.

'**SavingsAccount**' adds a new data member called '**interestRate**', which represents the annual interest rate for the savings account. It also adds a member function called '**calculateInterest()**' that calculates the interest earned on the account based on the current balance and interest rate.

'**CheckingAccount**' adds a new data member called '**monthlyFee**', which represents the monthly fee for the checking account. It also adds a member function called '**deductMonthlyFee()**' that deducts the monthly fee from the account balance.

Now we can use these derived classes to create specific types of bank accounts:

```
42 int main() {  
43     SavingsAccount savings(1000.0, 1.5);  
44     CheckingAccount checking(500.0, 10.0);  
45  
46     savings.deposit(500.0);  
47     savings.withdraw(200.0);  
48  
49     checking.deposit(100.0);  
50     checking.deductMonthlyFee();  
51  
52     cout << "Savings balance: " << savings.getBalance() << endl;  
53     cout << "Checking balance: " << checking.getBalance() << endl;  
54  
55     return 0;  
56 }
```

Fig. 06 (Inheritance Explanation)

We create a '**SavingsAccount**' object with an initial balance of 1000.0 and an interest rate of 1.5%. We deposit 500.0 into the account and then withdraw 200.0. We also create a '**CheckingAccount**' object with an initial balance of 500.0 and a monthly fee of 10.0. We deposit 100.0 into the account and then deduct the monthly fee.

Finally, we output the current balances of the accounts to the console.

Output:



```
Microsoft Visual Studio Debug Console  
Savings balance: 1300  
Checking balance: 590
```

Fig. 07 (Inheritance Explanation)

Task 01: Library System**[Estimated time 90 minutes / 70 marks]**

Create a program that simulates a library system using classes to represent books, thesis, and magazines. It should also allow checkouts as well with these items.

First create a class Person which will have properties:

- Cnic
- Name
- Address
- gender

Your program should have a base class called Item that includes the following data members and member functions:

- **string title:** the title of the item
- **string author:** the author of the item
- **int numPages:** the number of pages in the book
- **isBorrowed (bool)**
- **void setTitle(string title):** a member function that sets the item's title
- **string getTitle():** a member function that returns the item's title
- **void setAuthor(string author):** a member function that sets the item's author
- **string getAuthor():** a member function that returns the item's author
- **void setId(int id):** a member function that sets the item's ID
- **int getId():** a member function that returns the item's ID
- **void setNumPages(int numPages):** a member function that sets the number of pages
- **int getNumPages():** a member function that returns the number of pages
- **Void Display();**

Your program should also have a derived class called Book that inherits from the Item class and includes the following additional data members and member functions:

- **string isbn:** the ISBN of the book (*must be unique*)
- **void setISBN(string isbn):** a member function that sets the book's ISBN
- **string getISBN():** a member function that returns the book's ISBN
- **Person Author:** Author of the book.

Methods:

- Default, Parameterized and copy constructors.
- Setters/getters
- Display Function

Your program should also have a derived class called Thesis that inherits from the Item class and includes the following additional data members and member functions:

- **Int id:** the id of thesis (*must be unique*)
- **void setId(int id):**
- **int getId();**
- **Person writer:** Author of the Thesis.

Methods:

- Default, Parameterized and copy constructors.
- Setters/getters
- Display Function

Create a class **Student** which will inherit from person class.

- Additional properties:
- Roll number
- Degree
- Department

Create the required methods for this class.

Create class **BorrowItems**:

Properties:

Book/ Thesis name (string)

Borrow date (string)

Return date (string)

lendingTo (student)

Create a function to lend a book or thesis to a student on a specified date. Store that student's information in lendTo object.

Create a display function as well.

Create a class **Library** with properties:

- Library name
- Person admin
- Books (vector)
- Thesis (vector)
- Borrowed (BorrowItems vector>

Methods:

- Create all three constructors constructors and a Destructor which displays a message.
- Create functions to add a book, thesis to library by specifying required details.
- There should be functions to add, delete, update and display them respectively.
- Add function to add a borrow event in Borrow vector so that a student can borrow a book or thesis for specified date.
- Create a function to show all the content of Borrowed vector so that we can see all the books taken by various students.
- Create a display function to display whole data of the Library.
- Create functions to return a book from a student. Remove that book borrow event from borrowed vector and set isborrowed variable to false.
- Create a function to describe how many books and Thesis are present and borrowed in total.

Task 02 Shapes:**[Estimated time 30 minutes / 30 marks]**

- Create a class named **Circle** with a data member named **radius**
- Implement a display function in the Circle class that prints "This is a circle".
- Derive two other classes named Cylinder and Sphere from the Circle class.
- Implement display functions in both derived classes that print "Cylinder is a circle" and "Sphere is a circle" respectively.
- Also, add an area function to the Circle class that prints "This is circle area" and calculates the area of the circle using the formula $\pi * \text{radius}^2$.
- Implement area functions in the Cylinder and Sphere classes that calculate the surface area of the cylinder and the surface area of the sphere respectively, using the formulae
- $(2 * \pi * \text{radius} * (\text{radius} + \text{height}))$ for the cylinder and $(4 * \pi * \text{radius}^2)$ for the sphere.
- Keep the data member radius private in all classes and access through getters and setters. Create default constructor, parameterized constructor for all classes. Declare members variables as public for now. Create objects for each class then call all functions on each object.

Post-Lab Activities:**Task 01: Course Registration****[Estimated time 60 minutes / 40 marks]**

Create a program that simulates a university course registration system using classes to represent students, courses, and enrollments. Your program should have a base class called **'Person'** that includes the following data members and member functions:

- **string name:** the name of the person
- **int id:** a unique identifier for the person
- **void setName(string name):** a member function that sets the person's name
- **string getName():** a member function that returns the person's name
- **void setId(int id):** a member function that sets the person's ID
- **int getId():** a member function that returns the person's ID

Create a derived class **Teacher** with attributes like teacher id (unique), department, coursesInrolled (Number of courses the Teacher is teaching).

Your program should also have a derived class called **'Course'** that includes the following data members and member functions:

- **string name:** the name of the course
- **int id:** a unique identifier for the course
- **int capacity:** the maximum number of students that can enroll in the course.
- **Teacher:** object of the teacher who is teaching this course.
- **CreditHours**

Your program should also have a derived class called **'Student'** that inherits from the **'Person'** class and includes the following additional data members and member functions:

- **vector<Course> courses:** a vector of pointers to Course objects representing the courses the student is enrolled in
- **void addCourse(course object):** a member function that adds a Course object to the courses vector
- **void dropCourse(course id):** a member function that removes a Course object from the courses vector
- **Void displayAllCourses()**
- **Update a course info(course id)**

In main function you will have a vector of Courses offered by University. When university starts to offer a new course to students, it is added in this vector. If a student is to be enrolled in one of these courses, this course must be passed to course vector present in that student's object.

Your program should allow the user to create new students and courses, enroll students in courses, drop students from courses, and view the list of courses a student is enrolled in. Additionally, the program should enforce capacity limits for courses, display an error message if a student tries to enroll in a full course, and display an error message if a student tries to drop a course, they are not enrolled in.

Here are some specific requirements and guidelines for the program:

- When a new student or course is created, the user should be prompted for the appropriate information (name, ID, capacity, etc.).
- Student and course IDs should be assigned automatically and sequentially as they are created.

- The program should display a list of all students and courses, sorted by ID.
- The program should allow the user to enroll a student in a course by entering the student's ID and the course's ID.
- The program should display a list of all courses a student is enrolled in when the user enters the student's ID.
- The program should enforce capacity limits for courses, displaying an error message if a student tries to enroll in a full course.
- The program should display an error message if a student tries to drop a course they are not enrolled in.

Submissions:

- For In-Lab Activity:
 - Save the files on your PC.
 - TA's will evaluate the tasks offline.
- For Pre-Lab & Post-Lab Activity:
 - Submit the .cpp file on Google Classroom and name it to your roll no.

Evaluations Metric:

- All the lab tasks will be evaluated offline by TA's
- **Division of Pre-Lab marks:** [20 marks]
 - Task 01: Vehicle Speed [10 marks]
 - Task 02: Animal Classes [10 marks]
- **Division of In-Lab marks:** [70 marks]
 - Task 01: Library System [35 marks]
 - Task 02: Music Streaming [35 marks]
- **Division of Post-Lab marks:** [40 marks]
 - Task 01: Course Registration [40 marks]

References and Additional Material:

- **Inheritance**
<https://www.programiz.com/cpp-programming/inheritance>
- **Base and Derived Classes**
<https://learncplusplus.org/learn-c-inheritance-base-classes-and-derived-classes/>

Lab Time Activity Simulation Log:

- Slot – 01 – 00:00 – 00:15: Class Settlement
- Slot – 02 – 00:15 – 00:40: In-Lab Task
- Slot – 03 – 00:40 – 01:20: In-Lab Task
- Slot – 04 – 01:20 – 02:20: In-Lab Task
- Slot – 05 – 02:20 – 02:45: Evaluation of Lab Tasks
- Slot – 06 – 02:45 – 03:00: Discussion on Post-Lab Task