# Objectives

After completing this lab, you will be able to:

* Implement a base class and derived classes in C++
* Understand the working of base class and derived class constructors/destructors.
* Understand the working and purpose of private and protected attributes of a base class.
* Use base class member functions in a derived class.

# Inheritance

# Important Note:

For this lab write complete code in one cpp file. You do not need to make separate header and cpp files for each class.

# Exercise 1:

Consider the following hierarchy as it exists in a university:

* There are two types of persons in the university i.e. Student and Faculty
* Every Person has some basic information that is common to all persons i.e. the first\_name and last\_name stored as private attributes and age which is a protected attribute.
* A student can in turn be either an Undergraduate or a Graduate student, every student has a cgpa.
* An undergraduate student has a fyp\_name as his private attribute.
* A graduate student has a thesis topic as his private attribute.
* A faculty member has private attributes about the number of courses he is currently teaching, i.e. his course\_count and a three digit telephone extension number.

Draw Class Diagram (on paper) to represent the hierarchy of classes that you see in the description above. Include all the attributes in your diagram.

# Exercise 2:

Implement the entire hierarchy of the Class diagram you created in Exercise 1 i.e. define all the classes along with their attributes and their inheritance. Every class should be defined in a separate header file named according to the class name.

# Exercise 3:

Add appropriate constructors and destructors to all the classes created in Exercise 2. For example the constructor for the Person class should take three inputs (for **first\_name**, **last\_name** and **age**). The student constructor should take four inputs, three for its parent class (i.e. Person) and one float value to be assigned to the **cgpa** attribute.

This is accomplished in the following manner:

Person ::Person (string fname, string lname, int age)

{

...

cout << ”Person() called”;

}

Student::Student (string fname, string lname,int age,float cgpa): Person(fname,lname,age)

{

...

cout << ”Student() called”;

}

This syntax can be generalized to any parent and child constructor accordingly. Following this syntax, define and implement constructors and destructors for all the classes. Also, Notice that you have to add a print statement in every constructor which announces that the constructor has been called.

Also add a print statement to every destructor which announces that the destructor has been called. For example, the destructor for Person should look like:

~Person()

{

cout << ”~Person() called”;

}

# Exercise 4:

In .h files and .cpp file, add getters and setters function headings and implementation (resp) for all attributes in all the classes that you have defined.

# Exercise 5:

Create a C++ source file called **Lab\_Inheritance.cpp**. This file contains the **main()** function. In this main function create an undergraduate student “Ted Thompson” with cgpa 3.91 who is 22 years of age and a faculty member “Richard Karp” who is 45 years of age and who is teaching 2 courses this semester and his extension number is 420. Build and execute the code, copy the output and paste inside /\* comments \*/ in your **Lab\_Inheritance.cpp** file.

It should be pasted like this:

/\*

Output for Exercise 5:

~Person() called

...

...

\*/

# Exercise 6:

You should notice that the **age** attribute in a Person is protected, while the **first\_name** and **last\_name** attributes are private. What could be the reason for this?

|  |
| --- |
|  |

# Exercise 7: Overriding parent functions

Add a member function **void print()** in the Person class. This method should print the name and age of the person.

**Sample output:** “Ted Thompson is 22 years old”

Override function print in Student class such that it prints whatever parent (Person) printing and should also print cgpa.

**Sample output:** “Ted Thompson is 22 years old, his cgpa is 3.91”

This should be done by calling the person’s print in student’s print function and not by coping code from person’s print in student’s print

Also add a member function **void print()** in the **Faculty** class. This function should print the name, age, number of courses and extension number of the faculty member.

**Sample output:** “Faculty Member name: Richard Karp, Age: 45, Number of courses: 2, Ext. 420”

Use the following **main()** in the **Lab\_Inheritance.cpp** file. Build and execute the program and paste the output inside comments in the file **Lab\_Inheritance.cpp**.

void main()

{

Student s("Ted","Thompson",22,3.91);

Faculty f("Richard","Karp",45,2,420);

//here the number of courses is 2

//and the extension number is 420

Undergraduate u ("Ted","Thompson",22,3.91,"The Event Locator");

Graduate g ("Arnold","Gates",25,3.01,"Distributed Algorithms");

s.print ();

f.print ();

u.print ();

g.print ();

}

}

Output

|  |
| --- |
|  |

Which class’ print function was called for following objects in main?

|  |  |
| --- | --- |
| Object | Class (who’s print was called) |
| s |  |
| f |  |
| u |  |
| g |  |

# Exercise 8:

Now add **void print()** in Graduate and undergraduate classes.

Their outputs should look like as follows:

**Sample output for void Graduate::print ()**

“Ted Thompson is a graduate student, his cgpa is 3.91 and his thesis topic is Distributed Algorithms”

**Sample output for void Undergraduate::print ()**

“Ted Thompson is an undergraduate student, his cgpa is 3.91 and his final year project is titled The Even Locator”

Now run the main again and fill the table.

Which class’ print function was called for following objects in main?

|  |  |
| --- | --- |
| Object | Class (who’s print was called) |
| s |  |
| f |  |
| u |  |
| g |  |

# Question: private public protected inheritance and private public and protected data members of parent.

Try to use the following implementation of print in student.

void Student::print ()

{

cout << first\_name << ” ” << last\_name

<< ”is ” << age <<” years old, his cgpa is ” << cgpa;

}

Now call the **print()** function for the student created in **main()** in the last exercise. Build the code, you will get an error. Paste the error in the following box.

|  |
| --- |
|  |

Why did you get this error?

|  |
| --- |
|  |