LAB MANUAL

OBJECT ORIENTED PROGRAMMING

[MC 307]

[3rd YEAR, B.Tech (MCE)]

Department of Applied Mathematics Delhi Technological University (Formerly Delhi College of Engineering) Bawana Road, Delhi-110042

COURSE DESCRIPTION

Subject Code	**:**	MC 307
Course Title	**:**	Object Oriented Programming
Contact Hours	:	L- 3 T- 0 P- 2
Examination Duration (Hrs.)	**:**	Theory: 3 Practical: 00
Relative Weightage	**:**	CWS: 15 PRS: 25 MTE: 20 ETE: 40 PRE: 00
Credits	:	4
Semester	**:**	V
Subject Area	**:**	DEC
Pre-requisite	:	Programming Fundamentals

Software to be used: C++ and Java programming languages

Experiment 1

Aim: Create a class named “Time” in C++ with three data members namely hour, minute and seconds. Create two member functions of the class namely “setTime” that sets the hour, minute and seconds for any object; and “print” function that prints the hour, minute, and seconds value for the object. Test the class by creating its two objects in main.

Objective : The aim is to make students learn how to create classes and their objects in an object oriented program. Students also learn how the member function can be defined within a class and how an object can call the member function of its class from the main function with an dot operator.

Software/ Resources used : C++ programming language

Procedure/ Methodology :

1. Create a class “Time” with data members and member functions as written in the aim of the practical.
2. Create two objects of the class “Time” inside the main function using the syntax: “Time obj1, obj2”.
3. Set the data members for each of the object with the help of dot operator, i.e., call the member function, say obj1.setTime, and set the data members for both the objects.
4. Call the print function from the object(s) defined inside the main function with the help of dot operator to print its hour, minute and seconds.

Conclusion: With this practical, students can learn the concept of classes, objects, and how objects can set their data members and member functions of its class from the main function in an object oriented environment such as C++.

Experiment 2

Aim: Write a program in C++ demonstrating the use of constructor (with no arguments), constructor (with parameterized arguments) and destructor. Create objects of that class in main and test those objects with these constructors and destructor function.

Objective : The aim is to make students learn how to create constructors and destructor functions inside a class. Students also learn how any object of the class allocates its memory and de-allocates its memory after calling the constructor and destructor function respectively.

Software/ Resources used : C++ programming language

Procedure/ Methodology :

1. Create any class in C++ with some data members, member functions and a constructor (with parameters and without parameters) and a destructor function.
2. Call the constructor function to set the data members for the object from the main. Use the dot operator to call the constructor.
3. Call any member function to verify the values for the data members set by the constructor.
4. Call the destructor to de-allocate the memory reserved by the object of the class. Use the dot operator to call the destructor.

Conclusion: With this practical, students can learn the concept of constructors, destructors, allocation of memory with constructors and de-allocation of memory with destructors. Students also understand the difference between default constructor and parameterized constructor.

Experiment 3

Aim: Write a C++ program illustrating the operator overloading of binary operators (addition and subtraction), and unary operators (prefix and postfix increment).

Objective : The aim is to make students learn how operator overloading works in object oriented programming. Students also learn two methods of operator overloading: one with friend function, and other without friend function.

Software/ Resources used : C++ programming language

Procedure/ Methodology :

1. Create any class in C++ with some data members, member functions and some operator functions that are to be overloaded.
2. Create one friend function of the class. Use this friend function to overload binary and unary operators.
3. Create objects of that class inside main function. Call the operator functions (friend functions and non-friend member functions) with the dot operator from the objects of that class.
4. Observe the behavior of the overloaded functions and analyze how those operators can be customized to work with objects of any class.

Conclusion: With this practical, students can learn the concept of operator overloading, i.e., how operators can be customized to work with objects of any class. Students also learn the concept of friend functions and how friend functions can help in implementing operator overloading.

Experiment 4

Aim: Implement inheritance in C++ program. Create a base class named Person (with data members first_name, and last_name). Derive a class named Employee from base class Person. The Employee class should include a data member named employee_id. Further a class named Manager should be derived from the class Employee and the Manager class should include two data members named employee_designation and employee_salary. Create at least three objects of class Manager, specify the values of all their attributes, and determine which manager has the highest salary.

Objective : The aim is to make students learn how inheritance works in C++. Students also learn how to create child classes from the parent class, relationship between the child and parent class and how objects of child class interact with parent class.

Software/ Resources used : C++ programming language

Procedure/ Methodology :

1. Create any class in C++ with some data members and member functions.
2. Create a child class from the above defined parent class using public inheritance, i.e., use public keyword while inheriting the parent class.
3. For the child class, inherit the public members from the parent class, and create some extra data members specific to the child class which are not present in the parent class.
4. Create objects of both parent and child classes inside the main function.
5. For both types of objects, call their corresponding member functions from the main function with the dot operator.

Conclusion: With this practical, students can learn the concept of inheritance in C++, i.e., how a child class can be created from the parent class and relationship between objects of both these classes. It also forms the basis of polymorphism in object oriented programming.

Experiment 5

Aim: Write a program in C++ demonstrating class templates.

Objective : The aim is to make students learn the concept of templates in object oriented programming, i.e. with general class templates, customized classes of integers, floats, or any other data type can be created at run-time.

Software/ Resources used : C++ programming language

Procedure/ Methodology :

1. Create a general class template <T> in C++ with some data members and member functions of type <T>.
2. Customize the class template <T> with integer, create an integer class at run-time, and test the member functions of that class with integer objects from the main function.
3. Customize the class template <T> with floats, create a float class at run-time, and test the member functions of that class with float objects from the main function.
4. Customize the class template <T> with character data type, create a character type class at run-time, and test the member functions of that class with character objects from the main function.

Conclusion: With this practical, students can learn the concept of class templates, i.e., how general class templates <T> can be customized specifically to work with integers, floats or any other data type at run time.

Experiment 6

Aim: Write a program in C++ demonstrating rethrowing an exception.

Objective : The aim is to make students learn the concept of exception handling, and how the exceptions can be handled at various levels, i.e., handling the exceptions at first catch block, or rethrowing it and further catching it at next try - catch block.

Software/ Resources used : C++ programming language

Procedure/ Methodology :

1. Create any class with some data members and member functions.
2. Create a try-catch block with some parameters, say divide by zero or overflow condition.
3. Throw an exception from the try block and catch that exception with its corresponding catch block.
4. Then, throw a different exception from the try block, do not catch that exception in the first catch block, rather rethrow the same exception from its catch block so that it is handled by the next try – catch block, thus implementing the concept of rethrowing an exception.

Conclusion: With this practical, students can learn how to handle the exceptions at various levels such as first try catch block, or rethrowing that exception and handling it at next try catch block.

Experiment 7

Aim: Create an inheritance hierarchy (any inheritance example of your choice) in JAVA. Create one super class and three sub classes should inherit from that super class. Also highlight the concept of function overriding and function overloading in your program.

Objective : The aim is to make students learn the concept of super class and sub classes in Java programming. Students also learn the difference between function overloading and function overriding in object oriented programming.

Software/ Resources used : Java programming language

Procedure/ Methodology :

1. Create any class in Java with some data members and member functions.
2. Inherit three sub classes from the above defined super class with public inheritance.
3. Create two functions with same name in the super class and implement function overloading with difference in terms of number or type of arguments.
4. Create two functions with same name, one in super class and other in sub class, and implement function overriding by calling the super class member function and sub class member function from the sub class object defined within the main function.
5. Test member functions of super and sub classes by creating objects of these classes inside the main function.

Conclusion: With this practical, students can learn the concept of super class, sub classes, function overloading and function overriding in Java programming.

Experiment 8

Aim: Write a JAVA program explaining the working of constructors along with the super keyword.

Objective : The aim is to make students learn the concept of constructors in Java programming. Students also learn the concept of Super keyword in Java, i.e., how super class constructor and methods can be called from the sub class with Super keyword.

Software/ Resources used : Java programming language

Procedure/ Methodology :

1. Create any class in Java with some data members and member functions.
2. Inherit a sub class from the above defined super class with public inheritance.
3. Create constructor for both sub class and super class.
4. Call the constructor of the super class from the sub class constructor.
5. Now, apply the Super keyword and then call the constructor of the super class from the sub class constructor and observe the differences.
6. Test member functions of super and sub classes by creating objects of these classes inside the main function.

Conclusion: With this practical, students can learn the concept of super class, sub class, and analyze the differences between constructor calling without Super keyword and constructor calling with the Super keyword.

Experiment 9

Aim: Write a Java program consisting of abstract classes and abstract method(s). The first concrete sub class in the inheritance hierarchy must implement the abstract method(s). Create the objects of the concrete class and test your program.

Objective : The aim is to make students learn the concept of abstract classes and abstract methods in Java. Students also learn the difference between concrete class and abstract class in object oriented programming and how abstract methods of abstract classes can be overridden in non-abstract sub classes.

Software/ Resources used : Java programming language

Procedure/ Methodology :

1. Create any abstract class in Java with some data members and abstract member functions.
2. Inherit two sub classes from the above defined super class with public inheritance, one with abstract keyword and other without abstract keyword.
3. Override the abstract method of the super class in the non abstract sub class.
4. Create objects of non abstract classes and call their corresponding methods.
5. Try to create objects of abstract classes and analyze the difference between abstract classes and concrete classes.

Conclusion: With this practical, students can learn the concept of abstract classes, abstract and non-abstract methods, and difference between abstract and concrete classes in object oriented programming.

Experiment 10

Aim: Write a Java program explaining the working of static methods in a class.

Objective : The aim is to make students learn the concept of static and non-static methods and data members. Students learn how one copy of static data member is shared by all objects of a class as compared to non static members, which every object has its own copy of.

Software/ Resources used : Java programming language

Procedure/ Methodology :

1. Create any class in Java with some data members and member functions.
2. Create some members and methods as static and some as non-static.
3. Define the body of static method with static data members.
4. Call the static and non-static member functions from the objects of the class created inside the main function.
5. Change the value of static data member from different objects and observe how one copy of static data member is shared by all objects of the class.

Conclusion: With this practical, students can learn the concept of static and non-static data members and static vs non-static methods defined inside a class.

Experiment 11

Aim: Write a Java program implementing Swing Applet.

Objective : The aim is to make students learn the concept of applets, which run on the browser and downloaded on demand, and swings which are the collection of user interface components in Java programming.

Software/ Resources used : Java programming language

Procedure/ Methodology :

1. Create any class in Java that inherits Applet in built class and implements the interface of Action Listener.
2. Create some text fields, labels, buttons and JLabel in the class.
3. Set their corresponding names and labels for those text fields and buttons.
4. Set their foreground and background colors, if required.
5. Run the Java program and observe the output in Java enabled web browser.

Conclusion: With this practical, students can learn the concept of applets and swings in Java and how these are used to design various real life applications on the web browser.