C# Assignments on Inheritance

Lab 1. Single Inheritance

In Single Inheritance, a derived class inherits from a single base class.

Problem:

Create a base class Person with properties like Name and Age. Derive a class Student from Person and add a property Grade. Create objects and display the data.

Lab 2. Multilevel Inheritance

In Multilevel Inheritance, a class is derived from another derived class.

Problem:

Create a base class Animal with a method Eat(). Derive a class Dog that inherits Animal and add a method Bark(). Further derive a class Puppy from Dog and add a method Weep(). Show the behavior.

Lab 3. Multiple Inheritance (via Interfaces)

C# does not support multiple inheritance directly, but it can be achieved using interfaces.

Problem:

Create two interfaces IPrintable and IScannable with respective methods Print() and Scan(). Implement both interfaces in a class PrinterScanner.

Lab 4. Hierarchical Inheritance

In Hierarchical Inheritance, multiple derived classes inherit from a single base class.

Problem:

Create a base class Shape with a method Draw(). Create two derived classes Circle and Rectangle, both inheriting from Shape, and override the Draw() method to show specific behavior.

Lab 5. Hybrid Inheritance (Using Interfaces)

Hybrid inheritance combines multiple types of inheritance. Since C# doesn't support direct multiple inheritance, hybrid inheritance is implemented using interfaces.

Problem:

Create two interfaces IMovable and IRechargeable. Create a class Vehicle implementing IMovable and derive ElectricCar from Vehicle implementing both interfaces.

Lab 6. Overriding Methods in Inheritance

Demonstrate method overriding where a base class method is overridden in the derived class.

Problem:

Create a class Employee with a method Work(). Derive a class Manager that overrides the Work() method to show a different implementation.

Lab 7. Abstract Classes

Create an abstract class and demonstrate inheritance with abstract methods.

Problem:

Create an abstract class Vehicle with an abstract method Drive(). Create two derived classes Car and Bike that implement the Drive() method.

Lab 8. Sealed Classes

Create a class that cannot be inherited using the sealed keyword.

Problem:

Create a sealed class MathOperations with a method Add(). Show that it cannot be inherited.

Lab 9. Constructor Chaining

Demonstrate how constructors are called in a class hierarchy.

Problem:

Create a base class Person with a parameterized constructor. Create a derived class Employee that calls the base class constructor.

Lab 10. Interface Inheritance

Demonstrate that one interface can inherit from another.

Problem:

Create an interface IDriveable with a method Drive(). Create another interface IRaceable that inherits from IDriveable and adds a method Race().

Lab 11. IS-A Relationship (Inheritance)

Problem:

Create a base class Animal with properties like Name and methods like Eat(). Create a derived class Dog that inherits from Animal and adds its own method Bark(). Show how the IS-A relationship works.

Lab 12.HAS-A Relationship (Composition)

Problem:

Create a class Engine with properties like HorsePower. Create a class Car that contains an instance of Engine and shows the HAS-A relationship. Demonstrate how the Car can use its Engine to show engine-related details.

Lab 13. Calling Base Class Method Using base

Problem:

Create a base class Person with a method DisplayInfo(). Derive a class Employee that overrides DisplayInfo() but still calls the base class's DisplayInfo() using base.

Lab 14. Accessing Base Class Variable Using base

Problem:

Create a base class Person with a property Name. In the derived class Student, hide the Name property using the new keyword and use base to access the base class's Name property.

Lab 15. Calling Base Class Constructor Using base

This assignment demonstrates how to use the base keyword to call the base class constructor from the derived class constructor.

Problem:

Create a base class Vehicle with a constructor that accepts brand. Derive a class Car that passes values to the base class constructor using base.