C# Assignment on Polymorphism

Program.cs at beginning and main programs at end

PROGRAMS:

```
using System;
using System.Collections.Generic;
using System.Dynamic;
using System.Linq;
using System.Runtime.Remoting.Contexts;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp5
{
    //Assignment 1: To demonstrate Polymorphism and its Advantages
    //Create a C# program demonstrating polymorphism by using a base class Shape and derived classes Circle and Rectangle.Each derived class should implement
    //a method Draw(). Show how polymorphism helps in calling the correct method based on the object type.

    //class Shape
    //{
    //    public virtual void Draw()
    //    {
    //        Console.WriteLine("Drawing a shape");
    //    }
    //}

    //class Circle : Shape
    //{
    //    public override void Draw()
    //    {
    //        Console.WriteLine("Drawing a circle");
    //    }
    //}

    //class Rectangle : Shape
    //{
    //    public override void Draw()
    //    {
```

```
    //        Console.WriteLine("Drawing a rectangle");
    //    }
    //}

//-------------------------------------------------------------------------------------------------------------------------------------------

    //Assignment 2. Method Overloading and its uses
    //Create a C# program to show method overloading by implementing a Multiply method with different parameter types, numbers and order.

    //public class Calculator
    //{
    //    public int Multiply(int a, int b)
    //    {
    //        return a * b;
    //    }

    //    public double Multiply(double a, double b)
    //    {
    //        return a * b;
    //    }

    //    public int Multiply(int a, int b, int c)
    //    {
    //        return a * b * c;

    //    }
    //}

//-------------------------------------------------------------------------------------------------------------------------------------------

    //Assignment 3. Method Overriding
    //Write a program demonstrating method overriding by creating a base class Vehicle and a derived class Car that overrides the Drive() method.

    //class Vehicle
    //{
    //    public virtual void Drive()
    //    {
    //        Console.WriteLine("Vehicle is driving.");
```

```
    //    }
    //}

    //class Car : Vehicle
    //{
    //    public override void Drive()
    //    {
    //        Console.WriteLine("Car is driving on the road.");
    //    }
    //}

//-------------------------------------------------------------------------------------------------------------------------
-----------------


    //Assignment 5. Polymorphism with Static Data and Methods.
    //Create a C# program that demonstrates polymorphism using a base class Employee and
derived classes Manager and Developer.Include a static field
    //to keep track of the total number of employees and a static method to display the total
count.
    //Use method overriding to demonstrate polymorphism, while also explaining the need for
static members in this context.

    //public class Employee
    //{
    //    public static int TotalEmployees = 0;

    //    public string Name { get; set; }
    //    public int ID { get; set; }

    //    public Employee(string name, int id)
    //    {
    //        Name = name;
    //        ID = id;
    //        TotalEmployees++;
    //    }

    //    public virtual void Work()
    //    {
    //        Console.WriteLine($"{Name} is working as an employee.");
    //    }

    //    public static void DisplayTotalEmployees()
    //    {
```

```
    //        Console.WriteLine($"Total Employees: {TotalEmployees}");
    //    }
    //}

    //public class Manager : Employee
    //{
    //    public Manager(string name, int id) : base(name, id) { }

    //    public override void Work()
    //    {
    //        Console.WriteLine($"{Name} is managing the team.");
    //    }
    //}

    //public class Developer : Employee
    //{
    //    public Developer(string name, int id) : base(name, id) { }

    //    public override void Work()
    //    {
    //        Console.WriteLine($"{Name} is developing software.");
    //    }
    //}

//-------------------------------------------------------------------------------------------------------------------------
----------------


    //Assignment 6. Polymorphism with Arrays as Properties in a Class .Create a C# program
demonstrating polymorphism using a base class Employee and derived
    //classes Manager and Developer.In this program, each employee should have a collection of
tasks (stored in an array) assigned to them. Use arrays as
    //properties in the class to handle this data, and demonstrate how polymorphism and arrays
work together in the solution.


    //public class Employee
    //{
    //    public string Name { get; set; }
    //    public string[] Tasks { get; set; }

    //    public Employee(string name, int taskCount)
    //    {
    //        Name = name;
```

```
    //        Tasks = new string[taskCount];
    //    }

    //    public virtual void AssignTask(string task)
    //    {
    //        Console.WriteLine($"{Name} assigned task: {task}");
    //    }
    //}

    //public class Manager : Employee
    //{
    //    public Manager(string name, int taskCount) : base(name, taskCount)
    //    {
    //    }

    //    public override void AssignTask(string task)
    //    {
    //        base.AssignTask(task);
    //        Console.WriteLine($"{Name} also delegated the task.");
    //    }
    //}

    //public class Developer : Employee
    //{
    //    public Developer(string name, int taskCount) : base(name, taskCount)
    //    {
    //    }

    //    public override void AssignTask(string task)
    //    {
    //        base.AssignTask(task);
    //        Console.WriteLine($"{Name} started working on the task.");
    //    }
    //}

//-------------------------------------------------------------------------------------------------------------------------
---------------


    //Assignment 7: Understanding Early Binding and Late Binding in C#
    // Create a C# program that demonstrates early binding (compile-time polymorphism) using
method overloading and late binding(runtime polymorphism)
    //using method overriding.This will help illustrate the differences between the two concepts in
the context of polymorphism.
```

```
    //class Shape
    //{
    //    public virtual void Draw()
    //    {
    //        Console.WriteLine("Drawing a shape");
    //    }
    //}

    //class Circle : Shape
    //{
    //    public override void Draw()
    //    {
    //        Console.WriteLine("Drawing a circle");
    //    }
    //}

    //class Square : Shape
    //{
    //    public override void Draw()
    //    {
    //        Console.WriteLine("Drawing a square");
    //    }
    //}

//-------------------------------------------------------------------------------------------------------------------------
---------------


    //Assignment 8. Achieving Runtime Polymorphism with Abstract Classes and Interfaces in C#
    //Create a C# program that demonstrates how runtime polymorphism is achieved using
abstract classes and interfaces.Define an abstract
    //class Shape and an interface IShape, implementing these in derived classes to showcase
polymorphism.

    //abstract class Shape
    //{
    //    public abstract void Draw();
    //}

    //// Interface
    //interface IShape
    //{
```

```
    //    void Draw();
    //}

    //// Derived class implementing abstract class and interface
    //class Circle : Shape, IShape
    //{
    //    public override void Draw()
    //    {
    //        Console.WriteLine("Drawing a circle");
    //    }
    //}

    //// Derived class implementing abstract class and interface
    //class Square : Shape, IShape
    //{
    //    public override void Draw()
    //    {
    //        Console.WriteLine("Drawing a square");
    //    }
    //}

//-------------------------------------------------------------------------------------------------------------------------
---------------


    //Assignment 9. Demonstrating the Need for Multiple Inheritance of Interfaces
    //Create a C# program that demonstrates the concept of multiple inheritance through
interfaces. The program will define two interfaces,
    //IMovable and IDrawable, and implement them in a class Car that showcases how a class
can inherit from multiple interfaces.

    //interface IMovable
    //{
    //    void Move();
    //}

    //interface IDrawable
    //{
    //    void Draw();
    //}

    //class Car : IMovable, IDrawable
    //{
    //    public void Move()
```

```
    //    {
    //        Console.WriteLine("Car is moving");
    //    }

    //    public void Draw()
    //    {
    //        Console.WriteLine("Car is drawn");
    //    }
    //}

//-------------------------------------------------------------------------------------------------------------------------------
---------------


    //Assignment 10. Polymorphism in C# with Readonly Property
    //Create a C# program that demonstrates polymorphism with a readonly property. Define a
base class and derived classes where each class
    //provides specific behavior for a method, while using a readonly property to ensure that
certain values cannot be modified after initialization.

    //class Shape
    //{
    //    public readonly string Name;

    //    public Shape(string name)
    //    {
    //        Name = name;
    //    }

    //    public virtual void Draw()
    //    {
    //        Console.WriteLine($"Drawing a {Name}");
    //    }
    //}

    //class Circle : Shape
    //{
    //    public Circle(string name) : base(name)
    //    {
    //    }

    //    public override void Draw()
    //    {
    //        Console.WriteLine($"Drawing a circle named {Name}");
```

```
    //    }
    //}

    //class Square : Shape
    //{
    //    public Square(string name) : base(name)
    //    {
    //    }

    //    public override void Draw()
    //    {
    //        Console.WriteLine($"Drawing a square named {Name}");
    //    }
    //}


}
```

MAIN PROGRAMS:

```
using System;
using System.Collections.Generic;
using System.Dynamic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp5
{
    internal class Program
    {
        static void Main(string[] args)

        //Assignment 1: To demonstrate Polymorphism and its Advantages

        //{
        //        Shape shape1= new Circle();
        //        Shape shape2 = new Rectangle();
        //        Shape shape3 = new Shape();
```

```
        //        shape1.Draw();
        //        shape2.Draw();
        //        shape3.Draw();

        //}

//-------------------------------------------------------------------------------------------------------------------------------
-----------------

        //Assignment 2. Method Overloading and its uses
        //{
        //    Calculator calculator = new Calculator();

        //    int result1 = calculator.Multiply(2, 3);
        //    double result2 = calculator.Multiply(2.5, 3.7);
        //    int result3 = calculator.Multiply(2, 3, 4);

        //    Console.WriteLine("Result 1: " + result1);
        //    Console.WriteLine("Result 2: " + result2);
        //    Console.WriteLine("Result 3: " + result3);
        //}

//-------------------------------------------------------------------------------------------------------------------------------
-----------------


        //Assignment 3. Method Overriding
        //{
        //    Vehicle vehicle = new Vehicle();
        //    Car car = new Car();

        //    vehicle.Drive();
        //    car.Drive();

        //    // Polymorphism in action
        //    Vehicle vVehicle = new Car(); // Polymorphic assignment
        //    vVehicle.Drive(); // Calls the overridden Drive() method from the Car class
        //}

//-------------------------------------------------------------------------------------------------------------------------------
-----------------


        //Assignment 5. Polymorphism with Static Data and Methods.
```

```
        //{
        //    Employee employee1 = new Employee("Alice", 1);
        //    Manager manager1 = new Manager("Bob", 2);
        //    Developer developer1 = new Developer("Charlie", 3);

        //    employee1.Work();
        //    manager1.Work();
        //    developer1.Work();

        //    Employee.DisplayTotalEmployees();
        //}

//-------------------------------------------------------------------------------------------------------------------------
----------

        //Assignment 6. Polymorphism with Arrays as Properties in a Class
        //{
        //    Employee manager = new Manager("John Doe", 5);
        //    Employee developer = new Developer("Jane Smith", 3);

        //    manager.AssignTask("Lead project meeting");
        //    developer.AssignTask("Write code for module A");

        //    Console.WriteLine();

        //    manager.Tasks[0] = "Review code";
        //    developer.Tasks[0] = "Debug application";

        //    Console.WriteLine($"{manager.Name}'s tasks:");
        //    foreach (string task in manager.Tasks)
        //    {
        //        Console.WriteLine("." + task);
        //    }

        //    Console.WriteLine();

        //    Console.WriteLine($"{developer.Name}'s tasks:");
        //    foreach (string task in developer.Tasks)
        //    {
        //        Console.WriteLine(". " + task);
        //    }
        //}
```

```
//-------------------------------------------------------------------------------------------------------------------------------
---------------


        //Assignment 7: Understanding Early Binding and Late Binding in C#

        //{
        //    // Early binding (method overloading)
        //    int result = Add(2, 3);
        //    Console.WriteLine("Result of adding 2 and 3: " + result);

        //    double doubleResult = Add(2.5, 3.7);
        //    Console.WriteLine("Result of adding 2.5 and 3.7: " + doubleResult);

        //    // Late binding (method overriding)
        //    Shape shape1 = new Circle();
        //    Shape shape2 = new Square();

        //    shape1.Draw();
        //    shape2.Draw();
        //}

        //static int Add(int a, int b)
        //{
        //    return a + b;
        //}

        //static double Add(double a, double b)
        //{
        //    return a + b;
        //}

//-------------------------------------------------------------------------------------------------------------------------------
---------------

        //Assignment 8. Achieving Runtime Polymorphism with Abstract Classes and Interfaces in
C#
        //{
        //    Shape shape1 = new Circle();
        //    Shape shape2 = new Square();

        //    shape1.Draw();
        //    shape2.Draw();
```

```
        //    // Using interface
        //    IShape shape3 = new Circle();
        //    IShape shape4 = new Square();

        //    shape3.Draw();
        //    shape4.Draw();
        //}

//-------------------------------------------------------------------------------------------------------------------------
----------------


        //Assignment 9. Demonstrating the Need for Multiple Inheritance of Interfaces
        //{
        //    Car car = new Car();
        //    car.Move();
        //    car.Draw();
        //}

//-------------------------------------------------------------------------------------------------------------------------
----------------


        //Assignment 10. Polymorphism in C# with Readonly Property
        //{
        //    Shape shape1 = new Circle(" Red Circle");
        //    Shape shape2 = new Square("Blue Square");

        //    shape1.Draw();
        //    shape2.Draw();


        //    shape1.Name = "Green Circle"; // causes compile-time error
        //}

    }
}
```