Task 1 (5 points): Data types and control structures

Define an int variable called "age" and initialize it with your age. Define a string variable named "name" and initialize it with your name. Write a loop that iterates from 1 to 10, and inside the loop print a message containing your name and the index of the current loop.

Assignment 2 (5 points): Classes and methods

Create a class named "Person" with the following properties: private integer field "age". Public string property "Name" get and set with accessories. A constructor that takes two parameters to initialize age and name. Create a public method called "Introduction" that returns the message: "Hi, I'm [Name] and I'm [age] years old". Create an object of the "Person" class, assign it a name and age, and call the "Introduction" method.

Task 3 (5 points): Inheritance and method overriding

Create a class called "Shape" with the following: protected double property "Area." A constructor that initializes "Area" and assigns 0 to it. Create a child class called "Rectangle" that inherits from "Shape" with the following: two private double fields "\_length" and "\_width." and properties corresponding to their name. Overload (Method Overloading) the "CalculateArea()" method in the "Rectangle" class, which takes length and width parameters and calculates the area accordingly. Create a child class called "Circle" that also inherits from "Shape" with the following: private double field "\_radius and a property corresponding to the name." Override the "CalculateArea()" method in the "Circle" class with a version that takes a radius parameter and calculates the area accordingly. Create objects of the "Rectangle" and "Circle" classes with different dimensions (length, width or radius). Calculate and print the areas of both rectangles and circles using the overloaded "CalculateArea()" methods.

Task 4 (5 points): Abstract classes and interfaces

Create an abstract class "Vehicle" with the following members: abstract method "StartEngine()" without implementing it. Create a successor class "Car" of the "Vehicle" class that provides an implementation of "StartEngine()" returning "Car engine started". Create a class "Bicycle", which is also a child of "Vehicle", and implements "StartEngine()" and returns the result "Pedaling the bicycle". Create objects of the "Car" and "Bicycle" classes and call their "StartEngine()" method.

Task 5 (5 points): Generics and general Generic List

Create a Generic class "MyList<T>". It should contain the following methods: Method "Add", which adds an element of type T to the List. A "Get" method that returns an element from List based on the passed index. Create a "MyList" object for integers (int type) and add some items to the list. Use the "Get" method to retrieve and print an element from the list based on the given index.

using System; // Importing the System namespace for basic functionalities

using System.Collections.Generic; // Importing the Collections.Generic namespace for using generic collections like List

// Main class that contains the entry point of the program

class Program

{

// The main method where execution starts

static void Main()

{

// Task 1: Data types and control structures

int age = 25; // Define an integer variable 'age' and initialize it with your age

string name = "YourName"; // Define a string variable 'name' and initialize it with your name

// Loop from 1 to 10

for (int i = 1; i <= 10; i++)

{

// Print a message with the name and current loop index

Console.WriteLine($"Hello, I'm {name}. This is iteration number {i}.");

}

// Task 2: Classes and methods

// Create an instance of the Person class with age and name

Person person = new Person(25, "YourName");

// Call the Introduction method and print the result

Console.WriteLine(person.Introduction());

// Task 3: Inheritance and method overriding

// Create an instance of Rectangle and calculate its area

Rectangle rectangle = new Rectangle();

double rectArea = rectangle.CalculateArea(5, 10); // Calculate area using length 5 and width 10

Console.WriteLine($"Area of Rectangle: {rectArea}");

// Create an instance of Circle and calculate its area

Circle circle = new Circle();

double circleArea = circle.CalculateArea(7); // Calculate area using radius 7

Console.WriteLine($"Area of Circle: {circleArea}");

// Task 4: Abstract classes and interfaces

// Create an instance of Car and call its StartEngine method

Vehicle car = new Car();

Console.WriteLine(car.StartEngine()); // Output: Car engine started

// Create an instance of Bicycle and call its StartEngine method

Vehicle bicycle = new Bicycle();

Console.WriteLine(bicycle.StartEngine()); // Output: Pedaling the bicycle

// Task 5: Generics and general Generic List

// Create an instance of MyList for integers

MyList<int> myList = new MyList<int>();

myList.Add(1); // Add the integer 1 to the list

myList.Add(2); // Add the integer 2 to the list

myList.Add(3); // Add the integer 3 to the list

// Retrieve and print the second element (index 1) from the list

Console.WriteLine(myList.Get(1)); // Prints: 2

}

}

// Task 2: Classes and methods

public class Person

{

private int age; // Private field to store age

public string Name { get; set; } // Public property for accessing the name

// Constructor that takes age and name as parameters

public Person(int age, string name)

{

this.age = age; // Initialize the private age field

this.Name = name; // Initialize the public Name property

}

// Method that returns an introduction message

public string Introduction()

{

return $"Hi, I'm {Name} and I'm {age} years old."; // Return a formatted string

}

}

// Task 3: Inheritance and method overriding

public class Shape

{

protected double Area; // Protected field to store the area

// Constructor initializing Area to 0

public Shape()

{

Area = 0; // Set initial area to 0

}

}

// Rectangle class that inherits from Shape

public class Rectangle : Shape

{

private double \_length; // Private field for length

private double \_width; // Private field for width

// Property for Length

public double Length

{

get { return \_length; } // Getter for Length

set { \_length = value; } // Setter for Length

}

// Property for Width

public double Width

{

get { return \_width; } // Getter for Width

set { \_width = value; } // Setter for Width

}

// Method to calculate area based on length and width

public double CalculateArea(double length, double width)

{

\_length = length; // Assign length

\_width = width; // Assign width

Area = \_length \* \_width; // Calculate area

return Area; // Return calculated area

}

}

// Circle class that inherits from Shape

public class Circle : Shape

{

private double \_radius; // Private field for radius

// Property for Radius

public double Radius

{

get { return \_radius; } // Getter for Radius

set { \_radius = value; } // Setter for Radius

}

// Method to calculate area based on radius

public double CalculateArea(double radius)

{

\_radius = radius; // Assign radius

Area = Math.PI \* \_radius \* \_radius; // Calculate area using πr²

return Area; // Return calculated area

}

}

// Task 4: Abstract classes and interfaces

public abstract class Vehicle

{

public abstract string StartEngine(); // Abstract method to start the engine

}

// Car class that inherits from Vehicle

public class Car : Vehicle

{

public override string StartEngine()

{

return "Car engine started"; // Return a message indicating the car engine has started

}

}

// Bicycle class that also inherits from Vehicle

public class Bicycle : Vehicle

{

public override string StartEngine()

{

return "Pedaling the bicycle"; // Return a message indicating pedaling the bicycle

}

}

// Task 5: Generics and general Generic List

public class MyList<T>

{

private List<T> \_list = new List<T>(); // Private field to store the list of type T

// Method to add an item to the list

public void Add(T item)

{

\_list.Add(item); // Add the item to the list

}

// Method to get an item from the list by index

public T Get(int index)

{

return \_list[index]; // Return the item at the specified index

}

}