Task 1 (5 points): Data types and control structures Define an int variable called "quantity" and initialize it (e.g., 7). Define a string variable named "itemName" and initialize it to some item name (e.g., "Widget"). Write a loop that iterates from 1 to 5 inclusive, and inside the loop print a message containing the product name and the index of the current loop. Assignment 2 (5 points): Classes and methods Create a class named "Employee" with the following properties: private integer field "\_employeeName". Public string property "Salary" with get and set accessories. A constructor that takes two parameters to initialize name and salary. Create a public method called "DisplayInfo" that returns the message: "Hi, I'm [\_employeeName] and I earn [Salary] per year". Create an object of class "Employee" with name and salary and call the "DisplayInfo" method. Task 3 (5 points): Inheritance and method overriding Create a class called "Item" with the following: protected double property "Price". A constructor that initializes "Price" and assigns 0 to it. Create a child class of "Item" called "Fruit", with two private fields "\_weightInKg" and "\_priceOfOneKg" and properties corresponding to their names. Overload (Method Overloading) the method "CalculateTotalPrice()" in the class "Fruit", which takes the parameters of weight and price of one kilogram ("\_priceOfOneKg") and calculates the final price accordingly. Create a child class called "Electronics" that also inherits from "Item" with the following: private int field "\_quantity" and private double field "\_oneUnitPrice" ” their respective property. Override the "CalculateTotalPrice()" method in the "Electronics" class with a version that takes a unit price and quantity parameter and calculates the final price accordingly. Create objects of the "Fruit" and "Electronics" classes, calculate and print the total prices of both the fruit and the electronic item using the overloaded "CalculateTotalPrice()" methods. Task 4 (5 points): Abstract classes and interfaces Create an abstract class "Bird" with the following members: the abstract method "MakeSound()" without implementing it. Create a successor class of "Bird" class "Sparrow" that provides an implementation of "MakeSound()" text: "Chirp." by returning Create a class "Pigeon", which is also a child of "Bird", and implements "MakeSound()" and returns the result "Coo". Create objects of the `Sparrow'' and Pigeon'' classes and call their MakeSound()'' method. Task 5 (5 points): Generics and general Generic List Create a Generic class "MyCollection<T>". It should contain the following methods: Method "AddItem", which adds an element of type T to the List. Method "GetItem" that returns an item from the List according to the passed index. Create a "MyCollection" object with string type Generic and add some items to the list. Use the "GetItem" method to retrieve and print the item from the list based on the given index. same with this

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 quantity = 7; // Define an integer variable 'quantity' and initialize it with 7

string itemName = "Widget"; // Define a string variable 'itemName' and initialize it with "Widget"

// Loop from 1 to 5 (inclusive)

for (int i = 1; i <= 5; i++)

{

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

Console.WriteLine($"Item: {itemName}, Iteration: {i}");

}

// Task 2: Classes and methods

// Create an instance of the Employee class with name and salary

Employee employee = new Employee("John Doe", 50000);

// Call the DisplayInfo method and print the result

Console.WriteLine(employee.DisplayInfo());

// Task 3: Inheritance and method overriding

// Create an instance of Fruit and calculate its total price

Fruit fruit = new Fruit();

double fruitTotalPrice = fruit.CalculateTotalPrice(2, 3.5); // 2 kg at $3.5 per kg

Console.WriteLine($"Total price of Fruit: ${fruitTotalPrice}");

// Create an instance of Electronics and calculate its total price

Electronics electronics = new Electronics();

double electronicsTotalPrice = electronics.CalculateTotalPrice(150.0, 2); // $150 per unit, quantity 2

Console.WriteLine($"Total price of Electronics: ${electronicsTotalPrice}");

// Task 4: Abstract classes and interfaces

// Create an instance of Sparrow and call its MakeSound method

Bird sparrow = new Sparrow();

Console.WriteLine(sparrow.MakeSound()); // Output: Chirp

// Create an instance of Pigeon and call its MakeSound method

Bird pigeon = new Pigeon();

Console.WriteLine(pigeon.MakeSound()); // Output: Coo

// Task 5: Generics and general Generic List

// Create an instance of MyCollection for strings

MyCollection<string> myCollection = new MyCollection<string>();

myCollection.AddItem("Apple"); // Add "Apple" to the collection

myCollection.AddItem("Banana"); // Add "Banana" to the collection

myCollection.AddItem("Cherry"); // Add "Cherry" to the collection

// Retrieve and print the second item (index 1) from the collection

Console.WriteLine(myCollection.GetItem(1)); // Prints: Banana

}

}

// Task 2: Classes and methods

public class Employee

{

private string \_employeeName; // Private field to store the employee's name

public double Salary { get; set; } // Public property for salary with get and set accessors

// Constructor that takes name and salary as parameters

public Employee(string employeeName, double salary)

{

\_employeeName = employeeName; // Initialize the private employee name field

Salary = salary; // Initialize the public Salary property

}

// Method that returns employee information

public string DisplayInfo()

{

return $"Hi, I'm {\_employeeName} and I earn {Salary} per year."; // Return a formatted string

}

}

// Task 3: Inheritance and method overriding

public class Item

{

protected double Price; // Protected field to store the price

// Constructor initializing Price to 0

public Item()

{

Price = 0; // Set initial price to 0

}

}

// Fruit class that inherits from Item

public class Fruit : Item

{

private double \_weightInKg; // Private field for weight in kg

private double \_priceOfOneKg; // Private field for price of one kg

// Property for WeightInKg

public double WeightInKg

{

get { return \_weightInKg; } // Getter for WeightInKg

set { \_weightInKg = value; } // Setter for WeightInKg

}

// Property for PriceOfOneKg

public double PriceOfOneKg

{

get { return \_priceOfOneKg; } // Getter for PriceOfOneKg

set { \_priceOfOneKg = value; } // Setter for PriceOfOneKg

}

// Overloaded method to calculate total price based on weight and price per kg

public double CalculateTotalPrice(double weight, double priceOfOneKg)

{

\_weightInKg = weight; // Assign weight

\_priceOfOneKg = priceOfOneKg; // Assign price of one kg

Price = \_weightInKg \* \_priceOfOneKg; // Calculate total price

return Price; // Return calculated price

}

}

// Electronics class that also inherits from Item

public class Electronics : Item

{

private int \_quantity; // Private field for quantity

private double \_oneUnitPrice; // Private field for unit price

// Property for Quantity

public int Quantity

{

get { return \_quantity; } // Getter for Quantity

set { \_quantity = value; } // Setter for Quantity

}

// Property for OneUnitPrice

public double OneUnitPrice

{

get { return \_oneUnitPrice; } // Getter for OneUnitPrice

set { \_oneUnitPrice = value; } // Setter for OneUnitPrice

}

// Overridden method to calculate total price based on unit price and quantity

public double CalculateTotalPrice(double unitPrice, int quantity)

{

\_oneUnitPrice = unitPrice; // Assign unit price

\_quantity = quantity; // Assign quantity

Price = \_oneUnitPrice \* \_quantity; // Calculate total price

return Price; // Return calculated price

}

}

// Task 4: Abstract classes and interfaces

public abstract class Bird

{

public abstract string MakeSound(); // Abstract method to make a sound

}

// Sparrow class that inherits from Bird

public class Sparrow : Bird

{

public override string MakeSound()

{

return "Chirp"; // Return the sound made by a Sparrow

}

}

// Pigeon class that also inherits from Bird

public class Pigeon : Bird

{

public override string MakeSound()

{

return "Coo"; // Return the sound made by a Pigeon

}

}

// Task 5: Generics and general Generic List

public class MyCollection<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 AddItem(T item)

{

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

}

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

public T GetItem(int index)

{

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

}

}