PROGRAMMING IN C#

Module 8: Abstract classes and Interface
Module 9: Properties and Indexers

Lab Guide for Lab4

Session Objectives

In this session, you will be practicing with

- Abstract classes and Interface.
- Properties and Indexers

Part 1 – Getting started (30 minutes)

1. Abstract class and properties demo

This application create 3 class:
`GeometricObject` *abstract class has*
+ Fields: color (String), weight (double)
+ Properties: PColor, PWeight (read, write)
+ Abstract method: `findArea(),findPerimeter`
Circle *class and class Program in Main method creates some objects and uses its fields and methods.*

Step 1: Open Visual Studio
Step 2: Select the menu File->New->Project to create console based project named 'GeometricObjectDemo'
Step 3: Create ***`GeometricObject.cs`*** *abstract class and write code in it*

```
public abstract class GeometricObject
{
      protected string color;
      protected double weight;
      // Default construct
      protected GeometricObject(){
         color = "white";
         weight = 1.0;
      }
      // Construct a geometric object
      protected GeometricObject(string color, double weight)
      {
         this.color = color;
         this.weight = weight;
      }
      //properties
      public string PColor
      {
           get{return color;}
           set{color = value;}
      }
      public double PWeight
      {
```

```
            get{return weight;}
            set{weight = value;}
        }
        // Abstract method
        public abstract double findArea();
        // Abstract method
        public abstract double findPerimeter();
}
```

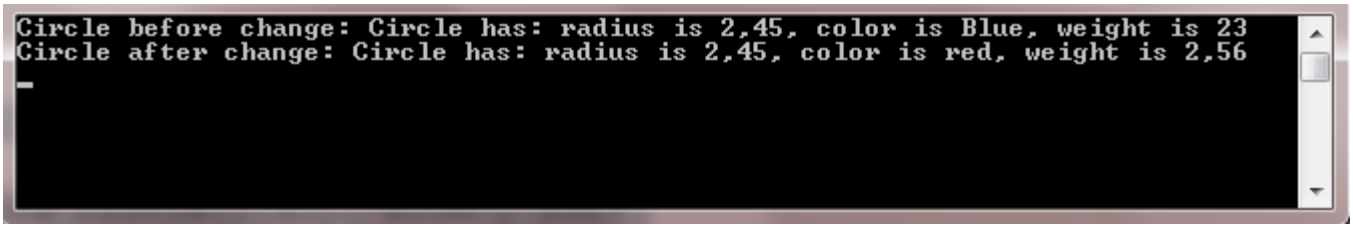
*Step 4: Create new file is **Circle.cs** and write code in it*

```
//Circle class extends GeometricObject class
public class Circle : GeometricObject {
    private double radius;
    public Circle(double x) {
        this.radius = x;
    }

    public Circle(double x, string c, double w) : base (c, w) {
        this.radius = x;
    }

    public override String ToString(){
        return "Circle has: radius is " + radius + ", color is " + PColor + ",
weight is " + PWeight;
    }

    public override double findArea(){return Math.PI*radius*radius;}

    public override double findPerimeter(){return 2 * Math.PI * radius;}

}
```

*Step 5: Create new file is **Program.cs** and write code in it*

```
public class Program{
        static void Main(string[] args){
            Circle c1 = new Circle(2.45, "Blue", 23);
            Console.WriteLine("Circle before change: " + c1.ToString());

            //using properties
            c1.PColor = "red";
            c1.PWeight = 2.56;

            Console.WriteLine("Circle after change: " + c1.ToString());
            Console.ReadLine();
        }
}
```

Step 6: Select menu File -> Save to save the file
Step 7: Select Build -> Build 'GeometricObjectDemo' option to build the project
Step 8: Select Debug -> Start without Debugging to execute the program
The output of the program as following



2. Indexers demo

*Step 1: Add a console based project '**IndexersDemo**' to the solution*

*Step 2: Right click on project **IndexersDemo** -> set as Startup project*

*Step 3: Rename the class file 'Program.cs' to '**IndexersDemo**.cs'*

*Step 4: Replace the code in '**IndexersDemo**.cs' with the given code*

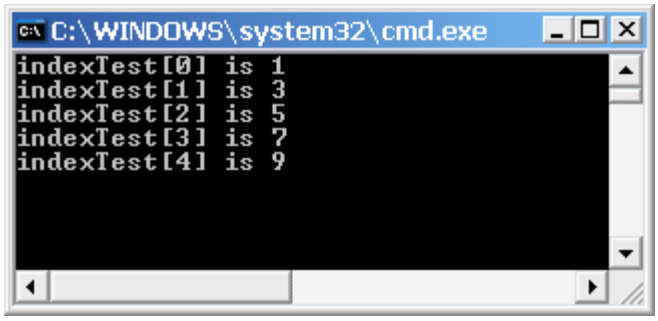
```
using System;
class IndexerExample
{
    public int[] intList = new int[10];
    public int this[int index]
    {
        get{return intList[index];}

        set{intList[index] = value;}
    }
}
class IndexerDemo
{
    static void Main()
    {
        int i, j = 0;
        IndexerExample indexTest = new IndexerExample();
        for (i = 1; i < 10; i += 2)
        {
            indexTest[j] = i;
            j++;
        }
        for (i = 0; i < 5; i++)
            Console.WriteLine("indexTest[{0}] is {1}", i, indexTest[i]);
        Console.ReadLine();
    }
}
```

Step 5: Select menu File -> Save to save the file

*Step 6: Select Build -> Build '**IndexersDemo**' option to build the project*

Step 7: Select Debug -> Start without Debugging to execute the program

The output of program as following



Part 2 – Workshops (30 minutes)

- Quickly look at Module 8 and 9 of workshops for reviewing basic steps for using Abstract class, Interface, Properties and Indexers types.
- Try to compile, run and observe the output of sample code provided for related workshops. Discuss with your class-mate and your instructor if needed.

Part 3 – Lab Assignment (60 minutes)

*Do the assignment for Module 8 carefully. Discuss with your class-mates and your instructor if needed. See **ACTCSharp_Module8_Assignment.pdf** file.*

Part 4 – Do it yourself

Implement a class named **Person** and two subclasses of Person named **Student** and **Employee**. Make **Faculty** and **Staff** subclasses of Employee. A Person has a name, phone number and email address. A student has a program to which he/she enrolled (Business, Computer Science...) . An Employee has a department, salary and the date hired. A faculty member has office hours and a rank. A staff member has a title. You are required to:

1. Override the **ToString**() to display the class name and the person's name and email address.
2. Provide properties in each class to read and write it's fields
3. Define a **CalculateBonus** and **CalculateVacation** as abstract methods in Employee class and implement them in Faculty and Staff as follows
    - Faculty get 1000 + 0.05 x Salary and Staff get 0.06 x Salary
    - Faculty get 5 weeks if they are employed more than 3 years and additional one week if he/she is "Senior Lecturer". Otherwise 4 weeks. Staff get 4 weeks for 5 year service. Otherwise get 3 weeks