Abstract Class & Interface

Abstract Class

We need to calculate area for some figures

Square

Rectangle

Triangle

Circle

// We created different classes for each figure

// Without Inheritance

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassDemo

{

class Square

{

int side, area;

public void GetDimensions()

{

Console.WriteLine("Enter Side");

side = int.Parse(Console.ReadLine());

}

public void CalculateArea()

{

area = side \* side;

}

public void DisplayArea()

{

Console.WriteLine("Area is " + area);

}

}

// RECTANGLE

class Rectangle

{

int l, w, area;

public void GetDimensions()

{

Console.WriteLine("Enter Length");

l = int.Parse(Console.ReadLine());

Console.WriteLine("Enter Width");

w = int.Parse(Console.ReadLine());

}

public void CalculateArea()

{

area = l \* w;

}

public void DisplayArea()

{

Console.WriteLine("Area is " + area);

}

}

// TRIANGLE

class Triangle

{

int b, h, area;

public void GetDimensions()

{

Console.WriteLine("Enter Base");

b = int.Parse(Console.ReadLine());

Console.WriteLine("Enter Height");

h = int.Parse(Console.ReadLine());

}

public void CalculateArea()

{

area = b \* h;

}

public void DisplayArea()

{

Console.WriteLine("Area is " + area);

}

}

class CalculateArea

{

}

}

When we see this example, we observed that many statements are same. We can remove that duplicacy (Inheritance)

While defining the base class, We realized that we cannot define or write statements in all the methods. WHY> because those statements are different in every ,method.

Such methods which do not contain any statements is Known as abstract method

Any class which contains even a single abstract method is known as Abstract Class

abstract class Figure

{

int dim1, dim2, area;

public abstract void GetDimensions();

public abstract void CalcuateArea();

public void DisplayArea()

{

Console.WriteLine("Area is " + area);

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassDemo

{

abstract class Figure

{

protected int dim1, dim2, area;

public abstract void GetDimensions();

public abstract void CalcuateArea();

public void DisplayArea()

{

Console.WriteLine("Area is " + area);

}

}

class Sqaure : Figure

{

public override void CalcuateArea()

{

area = dim1 \* dim1;

}

public override void GetDimensions()

{

Console.WriteLine("Enter Side");

dim1 = int.Parse(Console.ReadLine());

}

}

class Rectangle : Figure

{

public override void GetDimensions()

{

Console.WriteLine("Enter Length");

dim1 = int.Parse(Console.ReadLine());

Console.WriteLine("Enter Width");

dim2 = int.Parse(Console.ReadLine());

}

public override void CalcuateArea()

{

area = dim1 \* dim2;

}

}

class CalculateAreaWithAbstactClass

{

static void Main()

{

Console.WriteLine("Square");

Sqaure sqaure = new Sqaure();

sqaure.GetDimensions();

sqaure.CalcuateArea();

sqaure.DisplayArea();

Console.WriteLine("Rectnagle");

Rectangle rectangle = new Rectangle();

rectangle.GetDimensions();

rectangle.CalcuateArea();

rectangle.DisplayArea();

}

}

}

We cannot make object of Abstract Class , why? Because the class is not complete

**Interface : It’s a syntactical contract**

**We can only declare methods , properties, indexes . Variables cannot be declared**

**In interface none of the methods can be defined whereas in Abstract class, some of the methods are defined and dome of the methods are not defined.**

**By default , all the methods have public access specifier.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassDemo1

{

interface IFigure

{

void GetDimensions();

void CalcuateArea();

void DisplayArea();

}

class Triange : IFigure

{

int b, h, area;

public void CalcuateArea()

{

area = (int).5 \* b \* h;

}

public void DisplayArea()

{

Console.WriteLine("Area is " + area);

}

public void GetDimensions()

{

Console.WriteLine("Enter Base ");

b = int.Parse(Console.ReadLine());

Console.WriteLine("Enter Height ");

h = int.Parse(Console.ReadLine());

}

}

class Sqaure : IFigure

{

int s, area;

public void CalcuateArea()

{

area = s \* s;

}

public void GetDimensions()

{

Console.WriteLine("Enter Side");

s = int.Parse(Console.ReadLine());

}

public void DisplayArea()

{

Console.WriteLine("Area is " + area);

}

}

class Rectangle : IFigure

{

int w, l, area;

public void DisplayArea()

{

Console.WriteLine("Area is " + area);

}

public void GetDimensions()

{

Console.WriteLine("Enter Length");

l = int.Parse(Console.ReadLine());

Console.WriteLine("Enter Width");

w = int.Parse(Console.ReadLine());

}

public void CalcuateArea()

{

area = l \* w;

}

}

class InterfaceDemo

{

static void Main()

{

Console.WriteLine("Square");

Sqaure sqaure = new Sqaure();

sqaure.GetDimensions();

sqaure.CalcuateArea();

sqaure.DisplayArea();

Console.WriteLine("Rectnagle");

Rectangle rectangle = new Rectangle();

rectangle.GetDimensions();

rectangle.CalcuateArea();

rectangle.DisplayArea();

}

}

}