We are required to calculate Area of Square

using System;

using System.Collections.Generic;

using System.Text;

namespace AbstractClassInterfaceDemos

{

class Square

{

int side, area;

public void Get()

{

Console.WriteLine("Enter Side");

side = Convert.ToByte(Console.ReadLine());

}

public void CalcuateArea()

{

area = side \* side;

}

public void DisplayArea()

{

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

}

}

}

using System;

using System.Collections.Generic;

using System.Text;

namespace AbstractClassInterfaceDemos

{

class Rectangle

{

int length, width, area;

public void Get()

{

Console.WriteLine("Enter Length");

length = Convert.ToByte(Console.ReadLine());

Console.WriteLine("Enter Width");

width = Convert.ToByte(Console.ReadLine());

}

public void CalcuateArea()

{

area = length \* width;

}

public void DisplayArea()

{

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

}

}

}

using System;

using System.Collections.Generic;

using System.Text;

namespace AbstractClassInterfaceDemos

{

class Triangle

{

int Base, height, area;

public void Get()

{

Console.WriteLine("Enter Base");

Base = Convert.ToByte(Console.ReadLine());

Console.WriteLine("Enter height");

height = Convert.ToByte(Console.ReadLine());

}

public void CalcuateArea()

{

area = height \* height;

}

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 ConsoleApp2

{

abstract class Shape

{

protected int dimension1, dimension2, area;

abstract public void GetDetails();

abstract public void CalculateArea();

public void DisplayArea()

{

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

}

}

class Square : Shape

{

public override void GetDetails()

{

Console.WriteLine("ENter value of side");

dimension1 = Convert.ToByte(Console.ReadLine());

}

public override void CalculateArea()

{

area = dimension1 \* dimension1;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp2

{

class Rectangle : Shape

{

public override void GetDetails()

{

Console.WriteLine("ENter value of length");

dimension1 = Convert.ToByte(Console.ReadLine());

Console.WriteLine("ENter value of width");

dimension2 = Convert.ToByte(Console.ReadLine());

}

public override void CalculateArea()

{

area = dimension1 \* dimension2;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp2

{

class Triangle : Shape

{

public override void CalculateArea()

{

area = (int).5 \* dimension1 \* dimension2;

}

public override void GetDetails()

{

Console.WriteLine("ENter value of height");

dimension1 = Convert.ToByte(Console.ReadLine());

Console.WriteLine("ENter value of base");

dimension2 = Convert.ToByte(Console.ReadLine());

}

}

}

Abstract Class : A class in which some methods are defined and some are not defined(abstract methods)

The methods that are abstract need to be defined in Child class by using override keyword.

Purpose of Abstract Class : It is used to store common methods and it can be only inherited

It can not be instantiated as it’s is not complete

Interface : Syntactical contract in which we only declare methods, properties , indexers

It only contains methods / rules which has to be implemented by the classes which uses them