**Assignment -8 Solution**

1. We have to calculate the percentage of marks obtained in three subjects (each out of 100) by student A and in four subjects (each out of 100) by student B. Create an abstract class 'Marks' with an abstract method 'getPercentage'. It is inherited by two other classes 'A' and 'B' each having a method with the same name which returns the percentage of the students. The constructor of student A takes the marks in three subjects as its parameters and the marks in four subjects as its parameters for student B. Create an object for eac of the two classes and print the percentage of marks for both the students.

**Solution :-**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace AssignmentAbstract

{

abstract class Marks

{

public abstract double getPercentage();

}

class A :Marks

{

public double s1, s2, s3;

public A(double s1, double s2, double s3)

{

this.s1 = s1;

this.s2 = s2;

this.s3 = s3;

}

public override double getPercentage()

{

return (((s1 + s2 + s3 ) / 300) \* 100);

}

}

class B : Marks

{

public double s1, s2, s3, s4;

public B(double s1, double s2, double s3, double s4)

{

this.s1 = s1;

this.s2 = s2;

this.s3 = s3;

this.s4 = s4;

}

public override double getPercentage()

{

return (((s1 + s2 + s3 + s4) / 400) \* 100);

}

}

class Program

{

static void Main(string[] args)

{

double per;

A a = new A(50, 60, 70);

per = a.getPercentage();

Console.WriteLine("The percentage of student A is: " + per);

B b = new B(50, 60, 70,80);

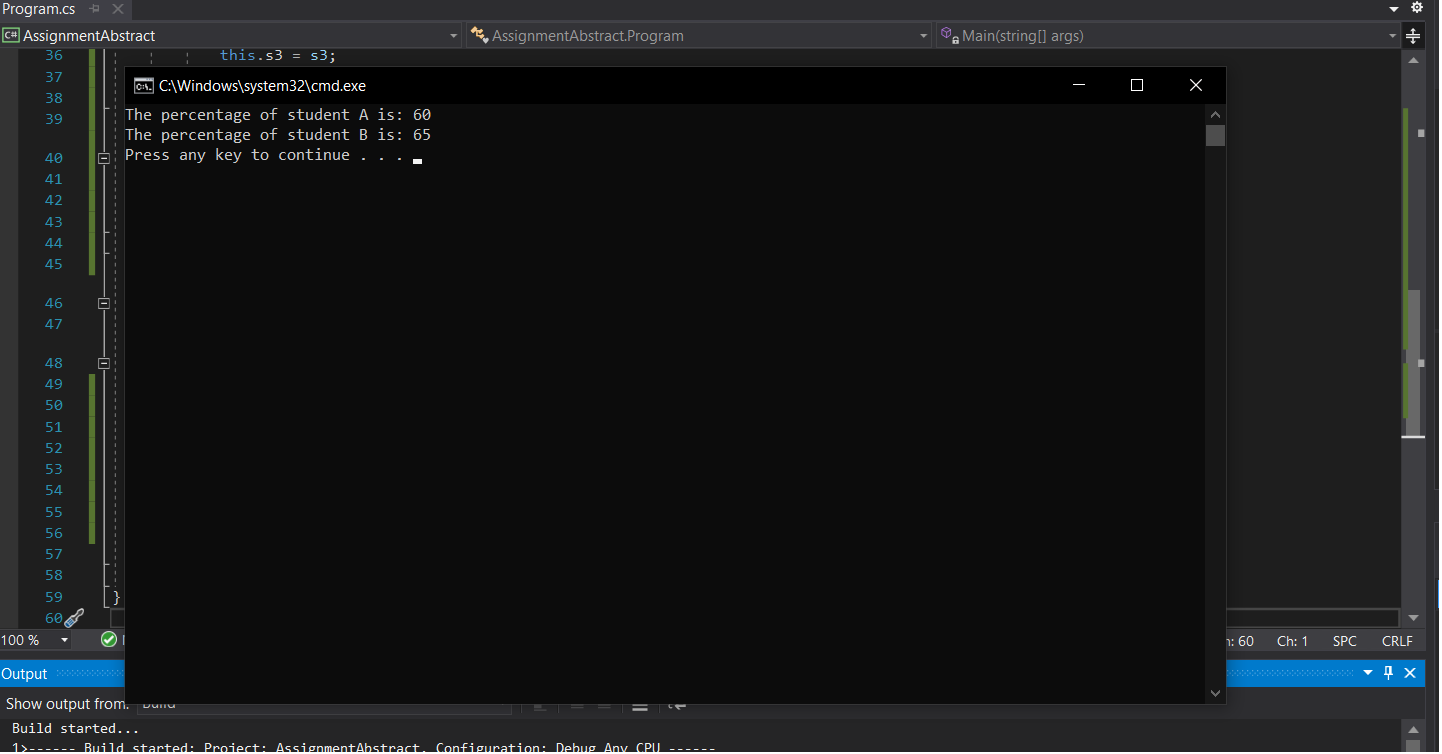
per = b.getPercentage();

Console.WriteLine("The percentage of student B is: " + per);

}

}

}



1. An abstract class has a construtor which prints "This is constructor of abstract class", an abstract method named 'a\_method' and a non-abstract method which prints "This is a normal method of abstract class". A class 'SubClass' inherits the abstract class and has a method named 'a\_method' which prints "This is abstract method". Now create an object of 'SubClass' and call the abstract method and the non-abstract method. (Analyse the result)

**Solution:-**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace AssignmentAbstract

{

abstract class Project

{

public Project()

{

Console.WriteLine("This is constructor of abstract class");

}

public abstract void a\_method();

public void non\_abs()

{

Console.WriteLine("This is a normal method of abstract class");

}

}

class Subclass : Project

{

public override void a\_method()

{

Console.WriteLine("This is abstract method");

}

}

class Program

{

static void Main(string[] args)

{

Subclass sb = new Subclass();

sb.non\_abs();

sb.a\_method();

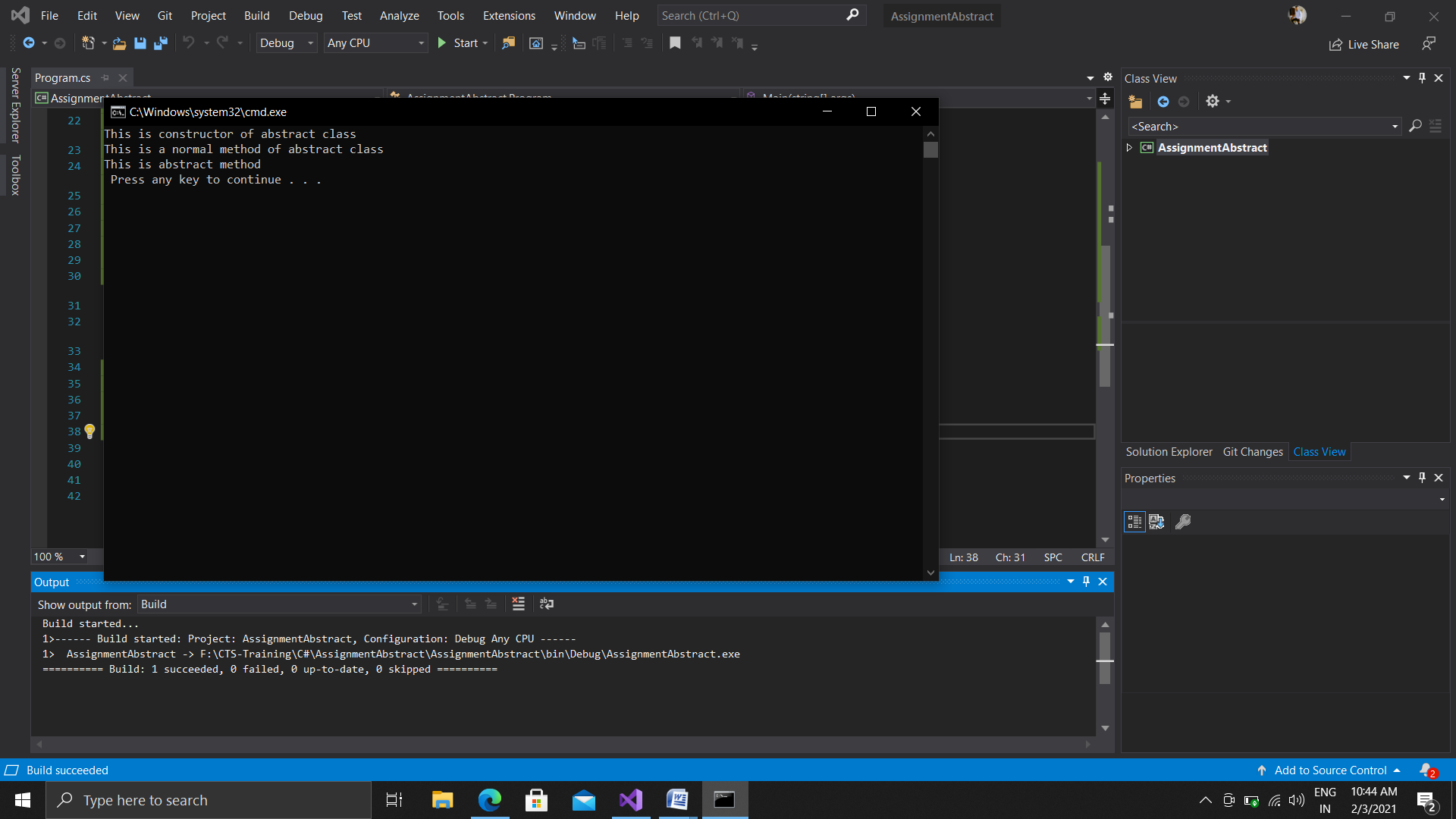
Console.ReadKey();

}

}

}

**Output:-**



3.Create an abstract class 'Animals' with two abstract methods 'cats' and 'dogs'. Now create a class 'Cats' with a method 'cats' which prints "Cats meow" and a class 'Dogs' with a method 'dogs' which prints "Dogs bark", both inheriting the class 'Animals'. Now create an object for each of the subclasses and call their respective methods.

using System;

namespace AssignmentAbstract

{

abstract class Animal

{

public abstract void cats();

public abstract void dogs();

}

class Cats : Animal

{

public override void cats()

{

Console.WriteLine("Cats meow");

}

public override void dogs()

{

}

}

class Dogs:Animal

{

public override void cats()

{

}

public override void dogs()

{

Console.WriteLine("Dog Barks");

}

}

class Program

{

static void Main(string[] args)

{

Cats c = new Cats();

c.cats();

Dogs d = new Dogs();

d.dogs();

}

}

}

**Output:-**

