**ASSESSMENT C#**

1. Answer all the problems by applying OOP concepts in all places applicable

a. Count the number of vowels in a given string

class Program

{

public static void Main(string[] args)

{

string str;

Console.WriteLine("Enter the string :");

str = Console.ReadLine().ToLower();

Vowels vowels = new Vowels(str);

Console.WriteLine("NUmber of Vowels in the Given String is : "+ vowels.str\_vowels());

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assessment

{

internal class Vowels

{

private int count=0;

private string str;

public Vowels(string str)

{

this.Str = str;

}

public int Count { get => count; set => count = value; }

public string Str { get => str; set => str = value; }

public int str\_vowels()

{

for (int i = 0; i < Str.Length; i++)

{

if (Str[i] == 'a' || Str[i] == 'e' || Str[i] == 'i' || Str[i] == 'o' || Str[i] == 'u')

{

Count++;

}

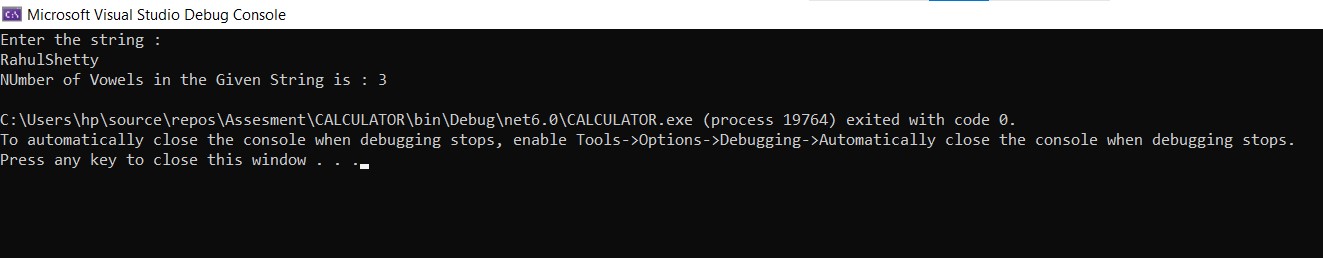
}

return Count;

}

}

}



b. Create a ATM class that would allow the end user to perform cash withdrawal and

deposit operations in sequence not by choice and print the balance in the account after

every operation.

class Program

{

public static void Main(string[] args)

{

double balance,withdraw,deposit;

int accountno;

string name;

Console.WriteLine("Enter the account number :");

accountno = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Enter the Name : ");

name = Console.ReadLine();

Console.WriteLine("Enter the Initial balance : ");

balance= Convert.ToDouble(Console.ReadLine());

ATM atm = new ATM(name, accountno, balance);

Console.WriteLine("Enter the Withdrawal Amount :");

withdraw= Convert.ToDouble(Console.ReadLine());

atm.withdrawal(withdraw);

Console.WriteLine("Enter the Deposit Amount :");

deposit = Convert.ToDouble(Console.ReadLine());

atm.Deposit(deposit);

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assessment

{

internal class ATM

{

private string name;

private double balance;

private int accountno;

public ATM(string name,int accountno, double balance)

{

this.Name = name;

this.Accountno = accountno;

this.Balance = balance;

}

public string Name { get => name; set => name = value; }

public double Balance { get => balance; set => balance = value; }

public int Accountno { get => accountno; set => accountno = value; }

public void withdrawal(double withdraw)

{

if (withdraw < this.Balance)

{

this.Balance = this.Balance - withdraw;

Console.WriteLine("balance after withdrawal : " + this.Balance);

}

else

{

Console.WriteLine("Your balance amount {0} is less than widrawal amount {1}",this.Balance,withdraw);

}

}

public void Deposit(double deposit)

{

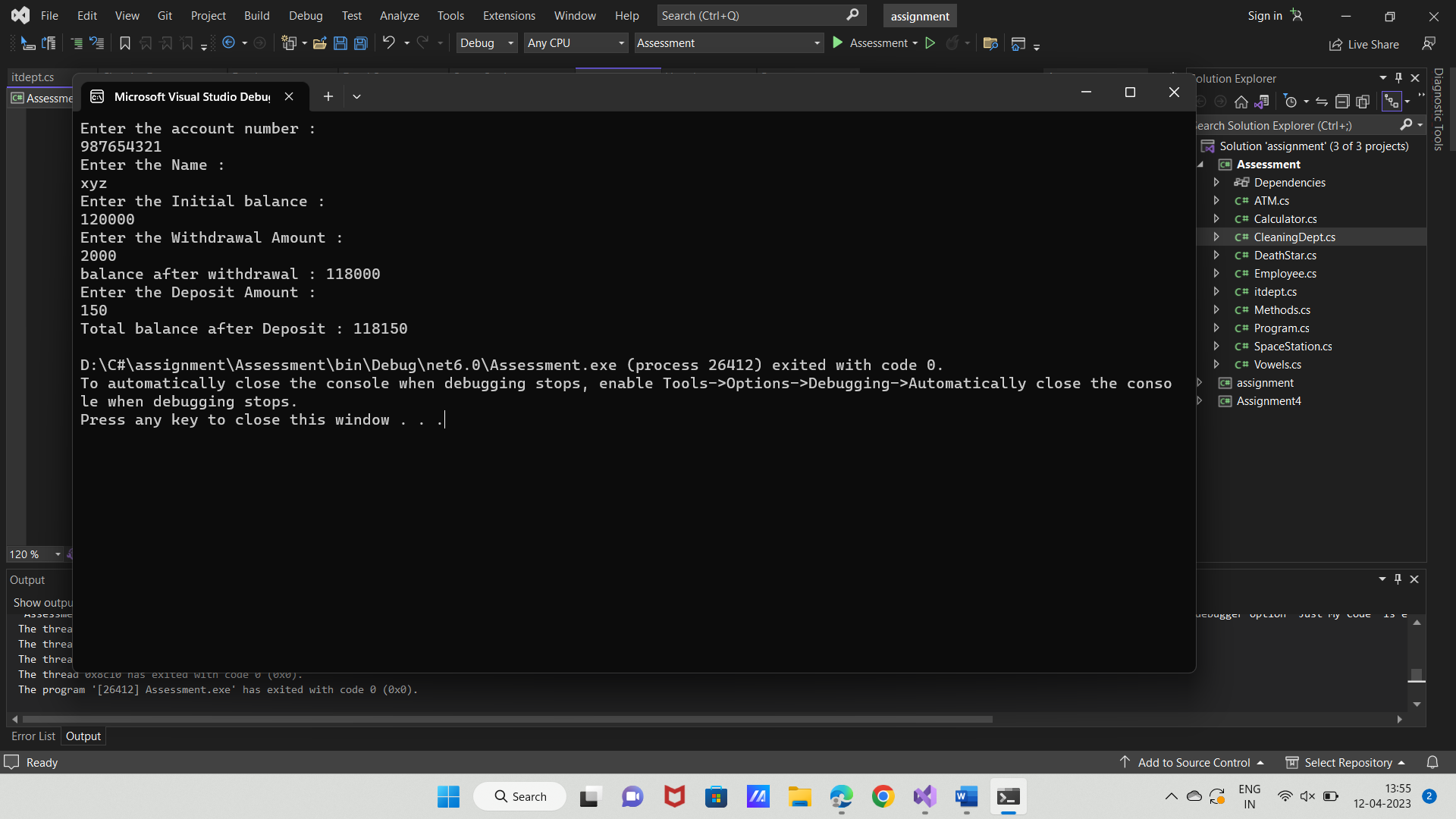
this.Balance += deposit;

Console.WriteLine("Total balance after Deposit : " + this.balance);

}

}

}



2. Implement hierarchical inheritance in a Company with employees of different departments

segregated in the child classes. Write method to calculate the employees bonus (assume bonus

is different for every department and is calculated from basic salary). (basic = 10000 bonus=10%

bonus amt=10000\*10% = 1000)

/\*Company parent class\*/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assesment

{

internal class Company

{

private string name;

protected string Name { get => name; set => name = value; }

public Company(string name)

{

this.Name = name;

}

}

}

/\*Data child class \*/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assesment

{

internal class Data : Company

{

private double bonuspercent = 0.5;

private double salary;

public double Bonuspercent { get => bonuspercent; set => bonuspercent = value; }

public double Salary { get => salary; set => salary = value; }

public Data(string name , double salary) : base(name)

{

this.Salary = salary;

}

public double bonus()

{

double bonusamount = Salary \* bonuspercent ;

return bonusamount;

}

}

}

/\* Hr child class \*/

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assesment

{

internal class Hr : Company

{

private double bonuspercent = 0.5;

private double salary;

public double Bonuspercent { get => bonuspercent; set => bonuspercent = value; }

public double Salary { get => salary; set => salary = value; }

public Hr(string name, double salary) : base(name)

{

this.Salary = salary;

}

public double bonus()

{

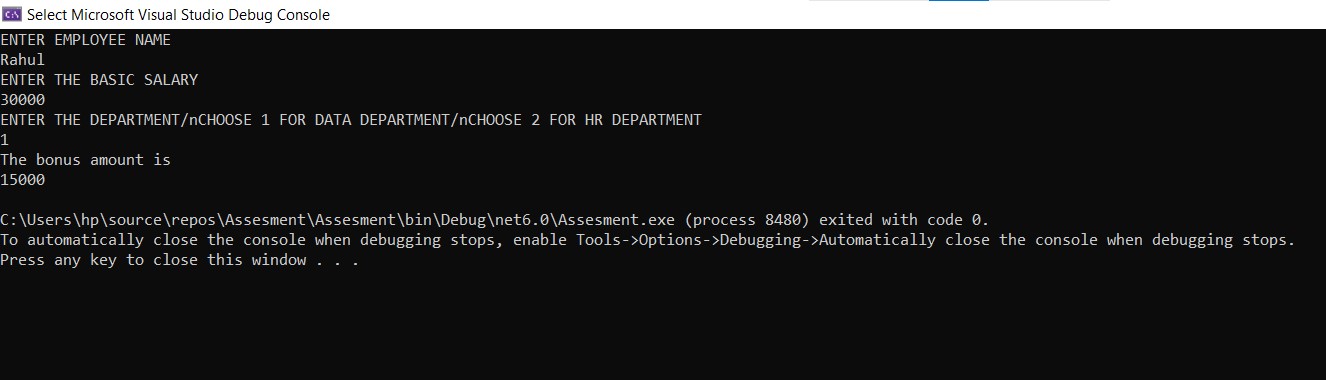
double bonusamount = Salary \* bonuspercent;

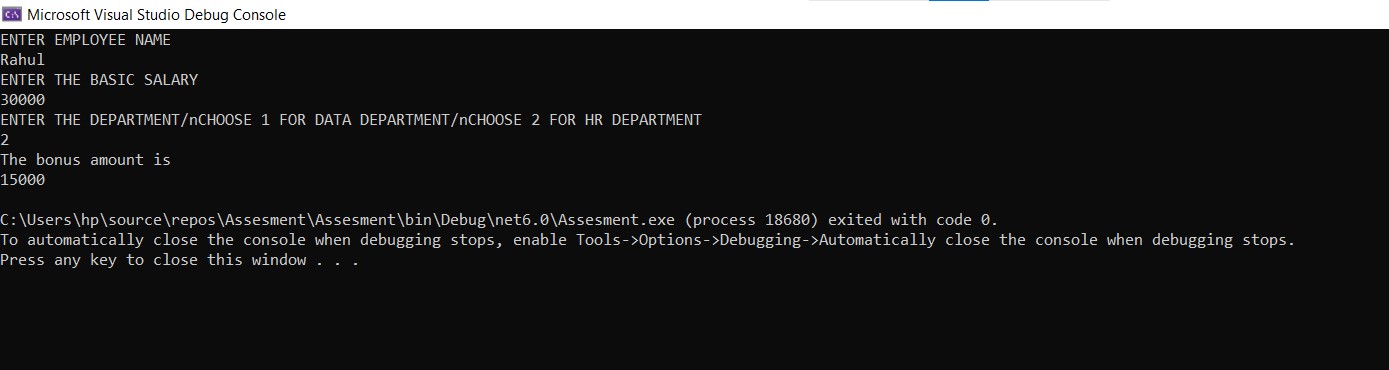
return bonusamount;

}

}

}





3.

a. Create a class named SpaceStation that is abstract

b. On that abstract class, add a abstract method called FireLaser

c. Create a derived class called DeathStar that implements the FireLaser method to write &quot;Pew

Pew&quot; to the Console followed by a new line

class Program

{

public static void Main(string[] args)

{

DeathStar ds = new DeathStar();

ds.FireLaser();

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assessment

{

abstract class SpaceStation

{

public abstract void FireLaser();

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Assessment

{

internal class DeathStar : SpaceStation

{

public override void FireLaser()

{

Console.WriteLine("Pew Pew");

}

}

}

