ASSIGNMENT

In class Lab 1 رابط تسليم حل المختبر قبل محاضرة العملي

اساليب كائنية المنحى
رابط تسليم حل المختبر قبل محاضرة العملي In class Lab 1

Done: Make a submission

Opened: Saturday, 4 March 2023, 2:00 AM
Due: Tuesday, 14 March 2023, 2:00 PM

Write an application that allows a user to input the height and width of a rectangle and output the area and perimeter. Use methods for entering the values, performing the computations, and displaying the results. Results should be formatted and printed in a tabular display.

Lab 1/week1

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

// lab 1

namespace lab_1

{

   class Program

   {

      static void Main(string[] args)

      {

         double L, W;

         GetValues(out L, out W);

         double area = ComputeArea(L, W);

         double peri = ComputePerimeter(L,W);

         DisplayResults(L, W, area, peri);

         Console.ReadKey();

      }
```

```
        static void GetValues(out double L, out double W)
        {
            Console.Write("Enter the L: ");
            L = double.Parse(Console.ReadLine());
            Console.Write("Enter the W: ");
           W = double.Parse(Console.ReadLine());
            Console.WriteLine();
        }
        static double ComputeArea(double L, double W)
        {
            return L * W;
        }
        static double ComputePerimeter(double L, double W)
        {
            return (L + W) * 2.0;
        }
        static void DisplayResults(double L, double W, double area, double peri)
        {
            Console.WriteLine("Rectangle Measurements\n");
            Console.WriteLine("Area: {0:F2}", area);
            Console.WriteLine("Peri: {0:F2}", peri);
        }
    }
}
```

……

ASSIGNMENT

In class Week 3 Lab رابط تسليم حل المختبر قبل محاضرة العملي اخر موعد للتسليم الثلاثاء 8 صباحا

Opened: Friday, 17 March 2023, 3:00 PM
Due: Tuesday, 21 March 2023, 8:00 AM

Write a class that simulates coin tossing. For each toss of the coin the program should print Heads or Tails. Then make a driver program witch will create a coin object and toss the coin 100 times, and count the number of times each side of the coin appears. Print the results. The program should call the flip() method of the coin object that takes no arguments and returns 0 for tails and 1 for heads.

Lab 2/week3

```
public class lab2
  {
    static void Main(string[] args)
    {
      int heads = 0, tails = 0, flip;
      Coin coin1 = new Coin();
      for (int i = 1; i <= 100; i++)
      {
        flip = coin1.flip();
        // The result heads
        if (flip == 1)
          heads++;
        // The result is tails
        else
          tails++;
      }
      Console.WriteLine("heads = " + heads);
      Console.WriteLine("tails = " + tails);
      Console.WriteLine("result = ");
```

```
        }
    }
    public class Coin
    {
        public int result;
        Random rnd = new Random();
        public int flip()
        {
            result = rnd.Next(1, 3);
            return result;
        }
    }
}
```

……

Opened: Friday, 24 March 2023, 8:00 AM
Due: Tuesday, 28 March 2023, 8:00 AM

Write a class called Rectangle that simulates a rectangle shape. The Rectangle class has the following items

1. Two attributes length and width (assume integer data type for the attributes)
2. Constructor to create and initialize the objects (Make sure you validate the value before you assign it to the instance variable as length and width can not be less or equal zero)
3. Two service methods to calculate and return the area and the circumference of the rectangle
4. Getter and Setter for each attribute

Then make a driver program(client) witch will do the following

1. create a rectangle object with a length and width interred by the user then
2. call the appropriate service methods to calculate area and circumference of the rectangle object and print the results
3. use the setters to change the width and length
4. call the appropriate service methods again to calculate the new area and circumference of the rectangle object and print the results

Lab 3 /week4

```
using System;

class Rectangle {

    private int _length;

    private int _width;

    public Rectangle(int length, int width) {

        if (length <= 0 || width <= 0) {

            throw new ArgumentException("Length and width must be greater than
zero.");

        }

        _length = length;

        _width = width;

    }

    public int Length {

        get { return _length; }

        set {

            if (value <= 0) {

                throw new ArgumentException("Length must be greater than zero.");
```

```
            }
            _length = value;
        }
    }
    public int Width {
        get { return _width; }
        set {
            if (value <= 0) {
                throw new ArgumentException("Width must be greater than zero.");
            }
            _width = value;
        }
    }
    public int GetArea() {
        return _length * _width;
    }
    public int GetCircumference() {
        return 2 * (_length + _width);
    }
}
class Program {
    static void Main() {
        Console.Write("Enter the length of the rectangle: ");
        int length = int.Parse(Console.ReadLine());
        Console.Write("Enter the width of the rectangle: ");
```

```
        int width = int.Parse(Console.ReadLine());

        Rectangle rectangle = new Rectangle(length, width);

        Console.WriteLine("Area of the rectangle: {0}", rectangle.GetArea());

        Console.WriteLine("Circumference of the rectangle: {0}",
rectangle.GetCircumference());

        Console.Write("Enter the new length of the rectangle: ");

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

        rectangle.Length = newLength;

        Console.Write("Enter the new width of the rectangle: ");

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

        rectangle.Width = newWidth;

        Console.WriteLine("New area of the rectangle: {0}", rectangle.GetArea());

        Console.WriteLine("New circumference of the rectangle: {0}",
rectangle.GetCircumference());

    }

}
```

.....

Opened: Friday, 31 March 2023, 9:00 AM
Due: Tuesday, 4 April 2023, 8:00 AM

Objectives

- Be able to design a class
- Be able to instantiate an object from a class
- Be able to create and use an overloaded constructors

Pre-Lab Questions

1. Give an example of a good use of an overloaded constructor .
2. How many overloaded method can a class have?
3. How can you make sure a method is properly overloaded?
4. Consider a class maths and we had a property called as sum. b which is the reference to a maths object and we want the statement Console.WriteLine(b.sum)to fail. Which among the following is the correct solution to ensure this functionality?
   a) Declares sum property with only get accessor
   b) Declares sum property with only set accessor
   c) Declares sum property with both set and get accessor
   d) Declares sum property with both set, get and normal accessor
5. Consider the following constructor definition:
   public MyClass(String name, int ID, int year)

{

name = name;

ID = ID;

year = year;

}

If name, ID, and year are instance variables of the class, is this definition legal? Come up with two different ways of writing this to make it both legal and more comprehensible.

Activities:
Develop a class called Date with the following members and methods:

1. private data members: day, month, year
2. properties for each private data member
3. methods:
   a. A print method that prints the date in traditional format – "mm/dd/yyyy"
   b. A set date method, which takes 3 arguments, day, month and year and assigns them to the data members
   c. 2 constructors:

a default constructor – that sets the date to October 1 1979

an overloaded constructor that takes 3 arguments, and assigns them to the day, month and year data members

Develop a class called TestDate which does the following:

4. Create three Date objects, and initialize them as follows:
   a. Default date object
   b. A date object with the following date: 04/29/04
   c. And a date object with the following date: June 18th 2006
5. Print the objects
6. Ask the user for a date, and create a new date object that represents the user given date.
7. Print the date object of (6).

Lab 4 / week5

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

// lab 5

namespace lab5

{

    class Program

    {

        static void Main(string[] args)

        {

            Date date1 = new Date();

            Date date2 = new Date(29, 4, 2004);

            Date date3 = new Date(18, 6, 2006);

            Console.WriteLine("date1 day is : {0}", date1.Day);

            date1.PrintDate();
```

```
        date2.PrintDate();
        date3.PrintDate();
        date1.Day = 12;
    }
}
class Date
{
    private int day, month, year;
    public Date()
    {
        day = 1;
        month = 10;
        year = 1979;
    }
    public Date(int d, int m, int y)
    {
        if (d > 0 && d < 32)
            day = d;
        if (m > 0 && m < 13)
            month = m;
        if (y > 0)
            year = y;
    }
    public void PrintDate()
    {
```

```
    Console.WriteLine(month + "/" + day + "/" + year);
}
public int Day
{
    get
    {
        return day;
    }
    set
    {
        if (value > 0 && value <= 31)
            day = value;
    }
}
public int Month
{
    get
    {
        return month;
    }
    set
    {
        if (value > 0 && value <= 12)
            month = value;
    }
```

```
        }
        public int Year
        {
            get
            {
                return year;
            }
            set
            {
                if (value > 0)
                    year = value;
            }
        }
        public void SetDate(int d, int m, int y)
        {
            if (d > 0 && d < 32)
                day = d;
            if (m > 0 && m < 13)
                month = m;
            if (y > 0)
                year = y;
        }
    }
}
```

……

Opened: Saturday, 8 April 2023, 12:00 AM
Due: Tuesday, 11 April 2023, 8:00 AM

Objectives

- To demonstrates the use of a STATIC variable AND a STATIC getter method for it
- To demonstrate PRIVATE service methods called only by public methods
- To demonstrate the use of **this** to reference instance variables

Activities: Box Class

In this exercise, you will write a class that models a Box based on the following UML description.

Box CLASS (UML description)

- count : int STATIC
---------- OBJECT ATTRIBUTES
- length : int
- width : int
- height : int
- weight : int
- boxNumber : int
---------- OBJECT BEHAVIORS
+ getCount () : int STATIC
+ getBoxNumber () : int

(setters not needed)

+ Box (length : int, width : int, height : int, weight : int, boxNumber : int)

+ findVolume () : int
+ findSurfaceArea () : int
+ findFootprint () : int

+ isOverWeight (limit : int) : boolean

+ BoxInfo() : String

- surfaceArea (dimen1 : int, dimen2 : int) : int

Write the class assuming a box object is described by five pieces of instance data (*length, width, height, weight,* and *boxNumber* as integer) and one piece of static variable count to hold the number of available box objects

The class should have the following methods:

a. A constructor that has five parameter that have the same name as instance variables..

b. Two getters a static method getCount( ) that returns count and a method getBoxNumber( ) to return boxNumber

c. findVolume( ) to return the volume of the box as integer; findSurfaceArea( ) to return the surface area of the box as integer, findFootprint( ) to return the foot print of the box as an integer(the area of the bottom of the box) and isOverWeight (limit : int) to return true if weight is greater than limit and false otherwise.

d. A private method surfaceArea(dimen1 : int, dimen2 : int) which will be used by the findSurfaceArea( ) to support in calculating the

if weight is greater than limit and false otherwise.

d. A private method surfaceArea(dimen1 : int, dimen2 : int) which will be used by the findSurfaceArea( ) to support in calculating the total surface area.

e. A BoxInfo method that returns a string containing complete information of the box.

2. Write a program that uses 3 box objects . Your program should do the following:

c. Read in the information of the three boxs and construct an object for each.
d. Print the info of each box (you will use the BoxInfo method here).
e. Print the number of Box objects created
f. print if a box object is over the limit of 3kg or not

Lab 5 /week6

```
using System;


namespace BoxExercise
{
    class Box
    {
        private int length;
        private int width;
        private int height;
        private int weight;
        private int boxNumber;
        private static int count;


        public Box(int length, int width, int height, int weight, int boxNumber)
        {
            this.length = length;
            this.width = width;
```

```
    this.height = height;
    this.weight = weight;
    this.boxNumber = boxNumber;
    count++;
}
public static int GetCount()
{
    return count;
}
public int GetBoxNumber()
{
    return boxNumber;
}
public int FindVolume()
{
    return length * width * height;
}
public int FindSurfaceArea()
{
    int side1 = surfaceArea(length, width);
    int side2 = surfaceArea(length, height);
    int side3 = surfaceArea(width, height);
    return 2 * (side1 + side2 + side3);
}
public int FindFootprint()
```

```
        {
            return surfaceArea(length, width);
        }
        public bool IsOverWeight(int limit)
        {
            return weight > limit;
        }
        private int surfaceArea(int dimen1, int dimen2)
        {
            return dimen1 * dimen2;
        }
        public string BoxInfo()
        {
            return $"Box number: {boxNumber}, Length: {length}, Width: {width},
Height: {height}, Weight: {weight}";
        }
    }
}
```

……

Opened: Friday, 14 April 2023, 8:00 AM
Due: Tuesday, 18 April 2023, 8:00 AM

Objectives

- Be able to design an aggregation and a composition class relationships
- Be able to use objects made up of other objects (Aggregation)
- Be able to write methods that pass and return objects

Activities

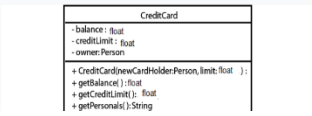
A credit card is an object that is very common. It is simple, but not overtly so. Attributes of the credit card include information about the owner, as well as a balance and credit limit. These things would be our instance fields. A credit card allows you to make payments and charges. These would be methods. As we have seen before, there would also be other methods associated with this object in order to construct the object and access its fields.

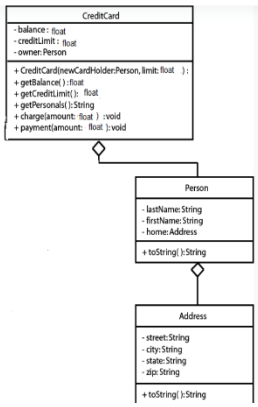
Examine the following UML diagram . Notice that the instance fields in the CreditCard class are other types of objects, a Person object . We can say that the CreditCard "has a" Person, which means aggregation, and the Person object "has a" Address object as one of its instance fields. This aggregation structure can create a very complicated object. We will try to keep this lab reasonably simple.

Task : Designing aggregation and Passing Objects

1. Create a CreditCard class according to the following UML Diagram . It should have data fields that include an owner of type Person, a balance of type float, and a creditLimit of type float.
2. It should have a constructor that has two parameters, a Person to initialize the owner and a float value to initialize the creditLimit. The balance can be initialized to a value of zero. Remember you are passing in objects (pass by reference), so you have passed in the address to an object.
3. It should have accessor methods to get the balance and the available credit.
4. It should have an accessor method to get the information about the owner, but in the form of a String that can be printed out. This can be done by calling the toString method for the owner (who is a Person).
5. It should have a method that will charge to the credit card by adding the amount of Money in the parameter to the balance if it will not exceed the credit limit. If the credit limit will be exceeded, the amount should not be added, and an error message can be printed to the console.
6. It should have a method that will make a payment on the credit card by subtracting the amount of Money in the parameter from the balance.
7. Write a CreditCardDriver class that will create a CreditCard object and use the charge, payment and getPersonals()

Compile, debug, and test it out completely





Lab 6 /week7

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

// lab 7

namespace lab_7

{

    class Program

    {

        static void Main(string[] args)

        {

            Address ad1 = new Address("ahmed Al", "Irbid", "ramtha", "12345");

            Person p1 = new Person("Omar", "Mahmoud", ad1);

            CreditCard c1 = new CreditCard(p1, 1000);

            Console.WriteLine("Card Info \n" + c1.getPerson());

            c1.charge(100);

            c1.payment(50);

            Console.WriteLine("Card Action : " + c1.getBalance());
```

```
        Console.ReadKey();
    }
}
class CreditCard
{
    private float balance, creditLimit;
    private Person owner;
    public CreditCard(Person CardHolder, float limit)
    {
        owner = CardHolder;
        creditLimit = limit;
    }
    public float getBalance()
    {
        return balance;
    }
    public float getCreditLimit()
    {
        return creditLimit;
    }
    public string getPerson()
    {
        return owner.tostring();
    }
    public void charge(float amount)
```

```
    {
        if (amount <= (creditLimit - balance))
        {
            balance = balance + amount;
        }
        else
            Console.WriteLine("you cant add it is over limit");
    }
    public void payment(float amount)
    {
        balance -= amount;
        Console.WriteLine("Balance debited : " + balance);
    }
}
class Person
{
    private string lastName, firstName;
    Address home;
    public Person(string firstName, string lastName, Address home)
    {
        this.firstName = firstName;
        this.lastName = lastName;
        this.home = home;
    }
    public string tostring()
```

```
        {
            return "Owner Info : " + firstName + " " + lastName + home.toString();
        }
    }
    class Address
    {
        private string street, city, state, zip;
        public Address(string street, string city, string state, string zip)
        {
            this.street = street;
            this.city = city;
            this.state = state;
            this.zip = zip;
        }
        public string toString()
        {
            return "\n Address Info : " + street + " , " + city + " , " + state + " , " + zip;
        }
    }
}
```

.....

Assignment I: Creating Classes and Object, Method Overloading, this reference and Static Members

Implement the following class diagram that represents an employee with its data and methods

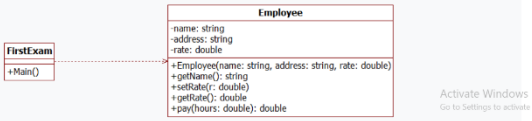
Step 1

Data: name, address and rate (rate is the cost of employee /hour)

Methods: a constructor witch used to initialize all data when an object is created by using the this reference as needed, getName, setRate, getRate, and pay (the pay method will return the cost of employee `hours * rate`).

Then write an application (client class) called EmpTest that will

1. Create two instances of the employee class using the following date
   Samer Amman 10.00 , Ali Zarqa 10.00
2. Invoke the pay method on them and display the result, note that the pay method needs hours worked as an argument substitute any number of hours you wish.



+pay(hours: double): double

Step 2

Data: Add a static private integer variable count to keep track of the number of employ objects created.

Methods:

1. Modify the constructor so it will increment count each time an object is created.
2. Add a static getCount method to return the value of count.
3. Using method overloading add another method called pay that takes two parameters' hours and bonus.

Modify EmpTest by

1. Add a statement to display number of employ objects created.
2. Invoke pay method on the created object with 40, 50 hours and a bonus of 50Jd, 60Jd respectively.

Assinement 1

```
namespace assignment_1
{
    class Program
    {
        static void Main(string[] args)
        {
            Employee emp1 = new Employee("Samer", "Amman", 10.00);
            Employee emp2 = new Employee("Ali", "Zarqa", 10.00);

            Console.WriteLine("Employee Info \n {0}", emp1.emploInfo());
            Console.WriteLine("Employee Rate is : " + emp1.pay(60));
            Console.WriteLine("Employee Bouns is : " + emp1.pay(40, 50));

            Console.WriteLine("--------------------------");

            Console.WriteLine("Employee Info \n {0}", emp2.emploInfo());
            Console.WriteLine("Employee Rate is : " + emp2.pay(70));
            Console.WriteLine("Employee Bouns is : " + emp2.pay(50, 60));
```

```
        Console.WriteLine("-------------------------");


        Console.WriteLine("Number of Employee : " + Employee.getCount());
    }
}
class Employee
{
    private string name;
    private string address;
    private double rate;
    private static int count;
    public Employee(string name, string address, double rate)
    {
        this.name = name;
        this.address = address;
        this.rate = rate;
        count++;
    }
    public string getName()
    {
        return name;
    }
    public string getAddress()
    {
        return address;
```

```
}
public double getRate()
{
    return rate;
}
public void setRate(double R)
{
    R = rate;
}
public double pay(double hours)
{
    return hours * rate;
}
public double pay(double bouns, double hours)
{
    return hours * rate + bouns;
}
public static int getCount()
{
    return count;
}
public string emploInfo()
{
    return "Employee Name is : " + name + "\n" +
  "Employee Address is : " + address;
```

```
        }

    }

}
```

.....

Opened: Sunday, 23 April 2023, 10:40 AM
Due: Thursday, 27 April 2023, 11:59 PM

Objectives

- Be able to design an aggregation and a composition class relationships
- Be able to use objects made up of other objects (Aggregation)
- Be able to write methods that pass and return objects

Activities
Task : Designing aggregation and Passing Objects

Create a University class according to the following UML Diagram. It should have data fields that include a department of type Department, and a uName of type string.

1. It should have a constructor that has one parameter, a name to initialize uName.
2. It should have a method AddDepartment() that will set a department by assigning the department object in the parameter to department reference variable. Remember you are passing in objects (pass by reference), so you have passed in the address to an object.
3. It should have an accessor method toString() to get the information about the University, but in the form of a String that can be printed out. This can be done by calling the toString method for the department (who is a Department).

Create a Department Class It should have data fields that include a professor of type Professor, and a dName of type string.

1. It should have a constructor that has one parameter, a name to initialize dName.
2. It should have a method AddProfessor() that will set a professor by assigning the professor object in the parameter to professor reference variable. Remember you are passing in objects (pass by reference), so you have passed in the address to an object.
3. It should have an accessor method toString() to get the information about the department, but in the form of a String that can be printed out. This can be done by calling the toString method for the professor (who is a Professor).

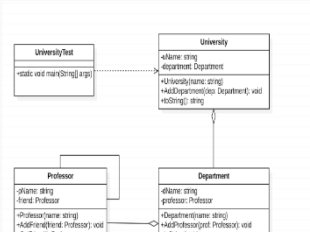
Create a Professor Class It should have data fields that include a friend of type Professor, and a pName of type string.

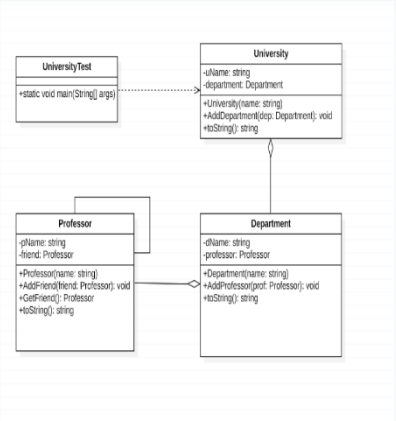
Create a Professor Class It should have data fields that include a friend of type Professor, and a pName of type string.

1. It should have a constructor that has one parameter, a name to initialize pName.
2. It should have a method AddFriend() that will set a friend by assigning the professor object in the parameter to professor reference variable. Remember you are passing in objects (pass by reference), so you have passed in the address to an object.
3. It should have an accessor method toString() to get the information about the professor, but in the form of a String that can be printed out.

Finally Write a UniversityTest class that will create a University object using data of your choice and use the toString() to return all university info (university info, department info and professor info).

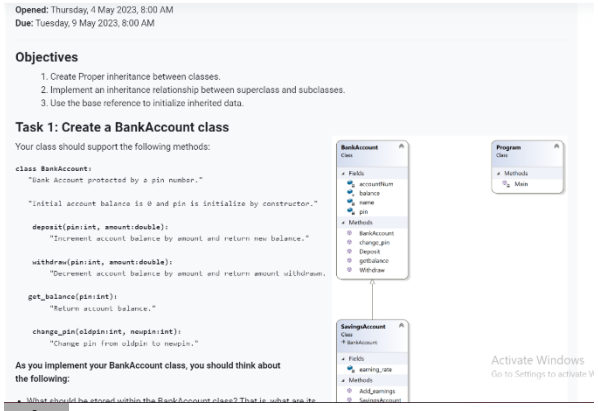
Compile, debug, and test it out completely





Lab 8

.....



Opened: Thursday, 4 May 2023, 8:00 AM
Due: Tuesday, 9 May 2023, 8:00 AM

Objectives

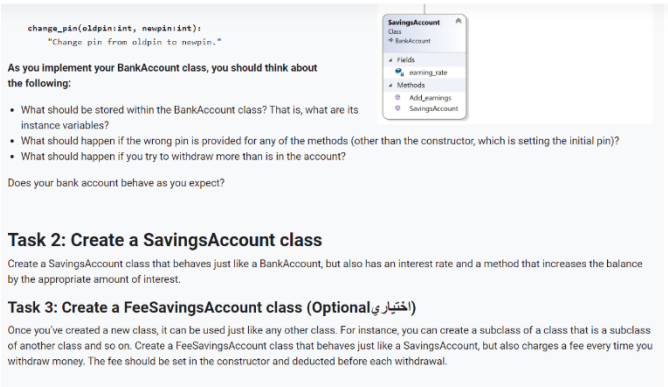
1. Create Proper inheritance between classes.
2. Implement an inheritance relationship between superclass and subclasses.
3. Use the base reference to initialize inherited data.

Task 1: Create a BankAccount class

Your class should support the following methods:

```
class BankAccount:
    "bank Account protected by a pin number."

  "initial account balance is 0 and pin is initialize by constructor."

   deposit(pin:int, amount:double):
       "Increment account balance by amount and return new balance."

   withdraw(pin:int, amount:double):
       "Decrement account balance by amount and return amount withdrawn."

  get_balance(pin:int):
       "Return account balance."

   change_pin(oldpin:int, newpin:int):
       "Change pin from oldpin to newpin."
```

As you implement your BankAccount class, you should think about the following:



```
   change_pin(oldpin:int, newpin:int):
       "Change pin from oldpin to newpin."
```

As you implement your BankAccount class, you should think about the following:

- What should be stored within the BankAccount class? That is, what are its instance variables?
- What should happen if the wrong pin is provided for any of the methods (other than the constructor, which is setting the initial pin)?
- What should happen if you try to withdraw more than is in the account?

Does your bank account behave as you expect?

Task 2: Create a SavingsAccount class

Create a SavingsAccount class that behaves just like a BankAccount, but also has an interest rate and a method that increases the balance by the appropriate amount of interest.

Task 3: Create a FeeSavingsAccount class (Optionalاختياري)

Once you've created a new class, it can be used just like any other class. For instance, you can create a subclass of a class that is a subclass of another class and so on. Create a FeeSavingsAccount class that behaves just like a SavingsAccount, but also charges a fee every time you withdraw money. The fee should be set in the constructor and deducted before each withdrawal.

Lab 9

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
// lab 9
namespace lab_9
{
   class Program

   {

      static void Main(string[] args)

      {
         SavingAccount saving1 = new SavingAccount("Khalel", "23333", 4714, 0.03);
         Console.WriteLine(saving1.Deposit(4714, 900.0));
         Console.WriteLine(saving1.withdraw(4714, 100));
         saving1.Add_earnings();
         saving1.change_pin(4714, 4444);
         Console.WriteLine(saving1.getbalance(4444));
      }
```

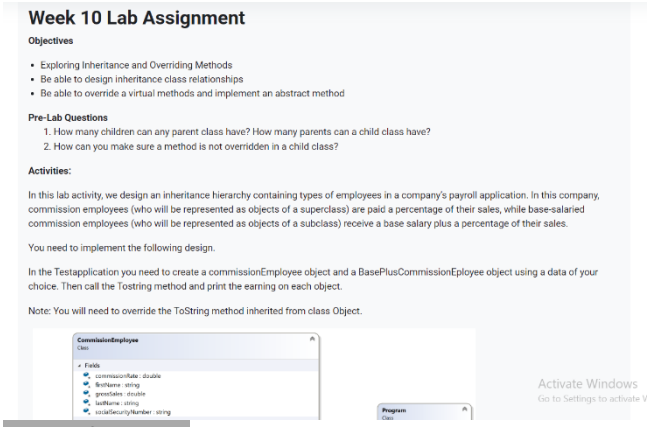
```
}

public class BankAccount

{



    private string name, accountNum;

    protected double balance;

    private int pin;

    public BankAccount(string name, string accountNum, int pin)

    {

        this.name = name;

        this.accountNum = accountNum;

        this.pin = pin;

    }

    public string Deposit(int pin, double amount)

    {

        if (this.pin == pin)
```

```
    {

        balance = balance + amount;

        return "Your balance after deposit an amount of" + amount + "is" + balance;

    }

    else

        return "Pin number dose not matchplease try again";

}

public string withdraw(int pin, double amount)

{

    if (this.pin == pin && balance - amount >= 0)

    {

        balance = balance - amount;

        return "Your balance after withdraw an amount of" + amount + "is" + balance;

    }

    else if (this.pin != pin)

        return "Pin number dose not match please try again";
```

```
        else

            return "insufficient fund please try again with different amount";
    }
    public string getbalance(int pin)
    {
        if (this.pin == pin)
            return "your balance is :" + balance;
        else
            return "wrong pin number please try again";
    }
    public void change_pin(int oldpin, int newpin)
    {
        if (pin == oldpin)
            pin = newpin;
        else
            Console.WriteLine("pin number mismatch please try again");
    }
}
public class SavingAccount : BankAccount
{
    private double earning_rate;
    public SavingAccount(string name, string accountNum, int pin, double rate)
        : base(name, accountNum, pin)
    {
        this.earning_rate = rate;
    }
    public void Add_earnings()
    {
        balance = balance + balance * earning_rate;
```

```
      }
    }

}
```

.....



Week 10 Lab Assignment

Objectives

- Exploring Inheritance and Overriding Methods
- Be able to design inheritance class relationships
- Be able to override a virtual methods and implement an abstract method

Pre-Lab Questions

1. How many children can any parent class have? How many parents can a child class have?
2. How can you make sure a method is not overridden in a child class?

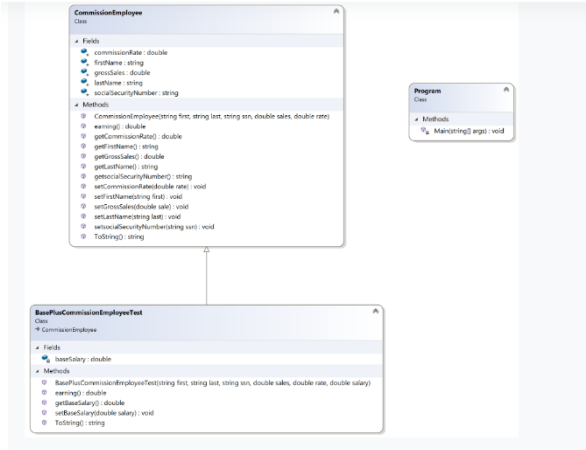
Activities:

In this lab activity, we design an inheritance hierarchy containing types of employees in a company's payroll application. In this company, commission employees (who will be represented as objects of a superclass) are paid a percentage of their sales, while base-salaried commission employees (who will be represented as objects of a subclass) receive a base salary plus a percentage of their sales.

You need to implement the following design.

In the Testapplication you need to create a commissionEmployee object and a BasePlusCommissionEployee object using a data of your choice. Then call the Tostring method and print the earning on each object.

Note: You will need to override the ToString method inherited from class Object.



Lab 10/week11

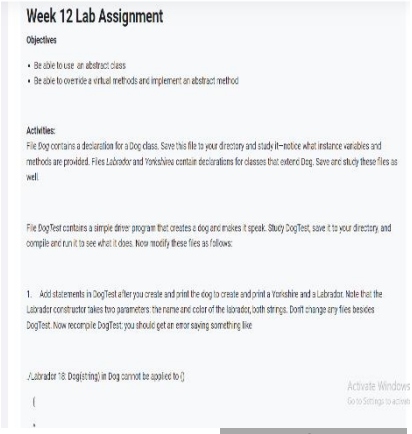
```
using System;



// lab 10

namespace ConsoleApp3

{

   class CommissionEmployee

   {

      protected String firstName;

      protected String lastName;

      protected String socialSecurityNumber;

      protected double grossSales;

      protected double commissionRate;

      public CommissionEmployee(string firstName, string lastName, string
socialSecurityNumber, double grossSales, double commissionRate)

      {

         this.firstName = firstName;

         this.lastName = lastName;

         this.socialSecurityNumber = socialSecurityNumber;
```

```
        this.grossSales = grossSales;

        this.commissionRate = commissionRate;

    }

    public void setFirstName(String first) { firstName = first; }

    public String getFirstName() { return firstName; }

    public void setLastName(String last) { lastName = last; }

    public String getLastName() { return lastName; }

    public void setSocialSecurityNumber(String ssn) { socialSecurityNumber =
ssn; }

    public String getSocialSecurityNumber() { return socialSecurityNumber; }

    public void setGrossSales(double sales) { grossSales = sales; }

    public double getGrossSales() { return grossSales; }

    public void setCommissionRate(double rate) { commissionRate = rate; }

    public double getCommissionRate() { return commissionRate; }

    public virtual double earning() { return grossSales * commissionRate; }

    public override string ToString()

    {

        return "First name:" + firstName + "Last name:" + lastName + "ssn:" +
socialSecurityNumber + "grossSales:" + grossSales + "commission rate:" +
commissionRate;

    }

  }

  class BasePlusCommissionEployee : CommissionEmployee

  {

    double baseSalary;

    public BasePlusCommissionEployee(string firstName, string lastName,
string socialSecurityNumber, double grossSales, double commissionRate, double
Salary)
```

```
        : base(firstName, lastName, socialSecurityNumber, grossSales,
commissionRate)

      {

        baseSalary = Salary;

        this.firstName = firstName;

        this.lastName = lastName;

        this.socialSecurityNumber = socialSecurityNumber;

        this.grossSales = grossSales;

        this.commissionRate = commissionRate;

      }

      public void setBaseSalary(double s) { baseSalary = s; }

      public double getBaseSalary() { return baseSalary; }

      public override string ToString()

      {

        return "First name:" + firstName + "Last name:" + lastName + "ssn:" +
socialSecurityNumber + "grossSales:" + grossSales + "commission rate:" +
commissionRate + "base salary:" + baseSalary;

      }

      public override double earning()

      {

        return baseSalary + base.earning();

      }

    }

    class TestBasePlusCommissionEployee

    {

      static void Main(string[] args)

      {
```

```
        CommissionEmployee commission = new CommissionEmployee("nnnnn", "hhhhh", "ggggg", 800, .45);

        Console.WriteLine(commission.ToString());

        Console.WriteLine("earning" + commission.earning());

        BasePlusCommissionEployee b = new BasePlusCommissionEployee("llllll", "mmmmm", "yyyyyy", 200, .12, 700);

        Console.WriteLine(b.ToString());

        Console.WriteLine("earning" + b.earning());

        Console.ReadKey();

      }

    }

}
```

……



Week 12 Lab Assignment

Objectives

- Be able to use an abstract class
- Be able to override a virtual methods and implement an abstract method

Activities:

File Dog contains a declaration for a Dog class. Save this file to your directory and study it—notice what instance variables and methods are provided. Files Labrador and YorkshireVew contain declarations for classes that extend Dog. Save and study these files as well.

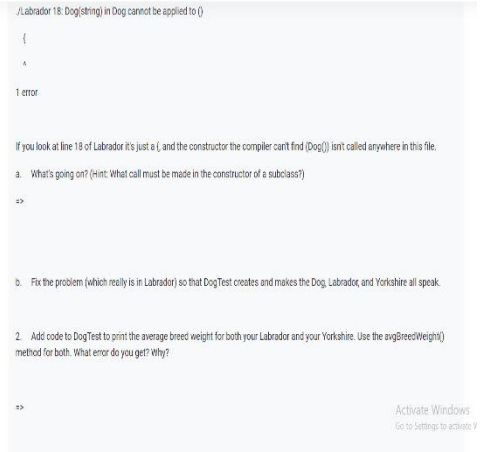
File DogTest contains a simple driver program that creates a dog and makes it speak. Study DogTest, save it to your directory, and compile and run it to see what it does. Now modify these files as follows:

1. Add statements in DogTest after you create and print the dog to create and print a Yorkshire and a Labrador. Note that the Labrador constructor takes two parameters: the name and color of the labrador, both strings. Don't change any files besides DogTest. Now recompile DogTest: you should get an error saying something like:

./Labrador 18: Dog(string) in Dog cannot be applied to ()

{

^



./Labrador 18: Dog(string) in Dog cannot be applied to ()

{

^

1 error

If you look at line 18 of Labrador it's just a {, and the constructor the compiler can't find (Dog()) isn't called anywhere in this file.

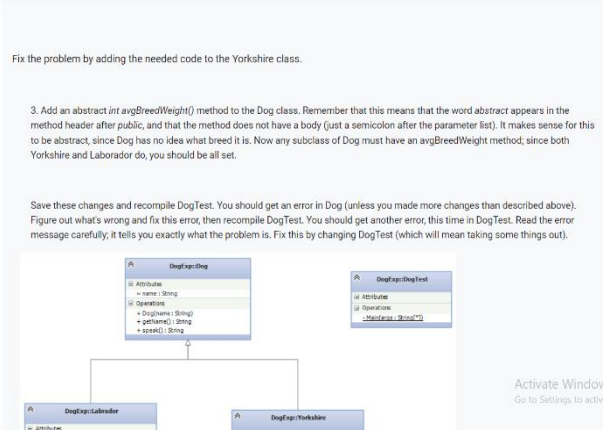
a. What's going on? (Hint: What call must be made in the constructor of a subclass?)

=>

b. Fix the problem (which really is in Labrador) so that DogTest creates and makes the Dog, Labrador, and Yorkshire all speak.

2. Add code to DogTest to print the average breed weight for both your Labrador and your Yorkshire. Use the avgBreedWeight() method for both. What error do you get? Why?

=>



Fix the problem by adding the needed code to the Yorkshire class.

3. Add an abstract *int avgBreedWeight()* method to the Dog class. Remember that this means that the word *abstract* appears in the method header after *public*, and that the method does not have a body (just a semicolon after the parameter list). It makes sense for this to be abstract, since Dog has no idea what breed it is. Now any subclass of Dog must have an avgBreedWeight method; since both Yorkshire and Laborador do, you should be all set.

Save these changes and recompile DogTest. You should get an error in Dog (unless you made more changes than described above). Figure out what's wrong and fix this error, then recompile DogTest. You should get another error, this time in DogTest. Read the error message carefully; it tells you exactly what the problem is. Fix this by changing DogTest (which will mean taking some things out).

Lab 12/week12

```
public abstract class Dog

{

    protected string name;

    public Dog(string name)

    {
```

```
        this.name = name;
    }
    public string getName()
    {
        return name;
    }
    public abstract int avgBreedWeight();
    public abstract string speak();
}
public class Labrador : Dog
{
    private string color;
    private static int breedWeight = 75;
    public Labrador(string name, string color)
        : base(name)
    {
        this.color = color;
    }
    public override string speak()
    {
        return "WOOF";
    }
    public override int avgBreedWeight()
    {
        return breedWeight;
```

```
    }
}
public class Yorkshire : Dog
{
    public Yorkshire(string name)
        : base(name)
    {
    }
    public override string speak()
    {
        return "woof";
    }
    public override int avgBreedWeight()
    {
        return 10; // Replace with appropriate breed weight for Yorkshire
    }
}
public class DogTest
{
    public static void Main(string[] args)
    {
        Labrador labrador = new Labrador("Max", "Black");


        Console.WriteLine(labrador.getName() + " says " + labrador.speak());
```

```
        Yorkshire yorkshire = new Yorkshire("Buddy");



        Console.WriteLine(yorkshire.getName() + " says " + yorkshire.speak());

        Console.WriteLine("Labrador average breed weight: " +
labrador.avgBreedWeight());

        Console.WriteLine("Yorkshire average breed weight: " +
yorkshire.avgBreedWeight());

    }

}
```

……

ASSIGNMENT

In class Week 12 Part 2 Lab رابط تسليم حل المختبر قبل محاضرة العملي اخر موعد للتسليم الثلاثاء 8 صباحا

كتابة المنحى

In class Week 12 Part 2 Lab رابط تسليم حل المختبر قبل محاضرة العملي اخر موعد للتسليم الثلاثاء 8 صباحا

Done: Make a submission

Opened: Friday, 19 May 2023, 9:54 AM
Due: Tuesday, 23 May 2023, 8:00 AM

Objectives

At the conclusion of this programming assignment, participants should be able to:

- Design, implement and test classes in C# which apply polymorphism
- Apply inheritance and polymorphism to model and simulate Animals and Pets

Pre-Lab Questions

1. What advantages does polymorphism provide?
2. If the class Utensil is declared as abstract, with Fork, Knife, and Spoon as its child classes, are the following declarations legal? Why or why not?

```
Utensil myFork;

Utensil mySpoon = new Utensil();
```



```
Utensil mySpoon = new Utensil();

Knife myKnife = new Utensil();

Spoon mySpork = new Fork();
```

Activities:

1. Below is the incomplete source code for the MorphingDogs class. Fill in the missing code so that every time the loop is executed, the user decides what type of dog myDog will be.

Note that this class makes use of Dog.cs, Labrador.cs, and Yorkshire.cs from the last lab. They need to be copied into your current directory for this to work.

```
using System;


// An example of Dynamic Binding using polymorphism
//*****************************************************************


public class MorphingDogs
{
  public static void Main(string[] args)
  {
      ------- myDog; // What type must dog be declared as so it can be any type of dog?
```


Lab 12p2 /week13

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

// lab 13

namespace lab_13

{

    public class MorphingDogs

    {

        public static void Main(string[] args)

        {

            Dog myDog; // What type must dog be declared as so it can be any type of dog?

            String ans = "y";

            int choice;

            while (ans == "y")

            {
```

```
            Console.WriteLine("Choose a Breed:\n1. Enter 1 for Labrador\n2. 2 for
Yorkshire");

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

            //The compiler cannont know at compile time what type myDog will be.

            //It is determined at run time every time the while loop is executed.

            if (choice == 1)

                myDog = new Labrador("nana", "white");

            else

                myDog = new Yorkshire("momo");

            Console.WriteLine(myDog.speak());

            Console.Write("Try again? ");

            ans = Console.ReadLine();

        }

    }

    public abstract class Dog

    {

        protected internal string name;

        protected static int breedWeight = 75;

        public Dog(string name)

        {

            this.name = name;

        }

        public string getName()

        {

            return name;

        }
```

```
    public virtual string speak()
    {
        return "Woof";
    }
    public static int avgBreedWeight()
    {
        return breedWeight;
    }
}
public class Labrador : Dog
{
    private string color;
    public Labrador(string name, string color) : base(name)
    {
        this.color = color;
    }
    public override string speak()
    {
        return "WOOF2";
    }
}
public class Yorkshire : Dog
{
    public Yorkshire(string name) : base(name)
    {
```

```
            this.name = name;
        }
        public override string speak()
        {
            return "woof3";
        }
    }
  }
}
```

.....

ASSIGNMENT

In class Week 14 Lab رابط تسليم حل المختبر قبل محاضرة العملي اخر موعد للتسليم الثلاثاء 8 صباحا

اساليب كائنية المنحى

رابط تسليم حل المختبر قبل محاضرة العملي In class Week 14 Lab
اخر موعد للتسليم الثلاثاء 8 صباحا

Done: Make a submission

Opened: Friday, 26 May 2023, 9:54 AM
Due: Tuesday, 30 May 2023, 8:00 AM

Objectives

At the conclusion of this programming assignment, participants should be able to:

- Design, implement and test classes in C# which apply polymorphism
- Apply inheritance and polymorphism to model and simulate Animals and Pets

Pre-Lab Questions

1. How many interfaces can any one class implement?

Activities:
Create an interface called dog with only one abstract method speak(), then redesign week 13 lab solution the Labrador and Yorkshire classes and make them implement the Dog interface. Once done run the MorphingDogs class you did in week 13 lab and see if it works as before.

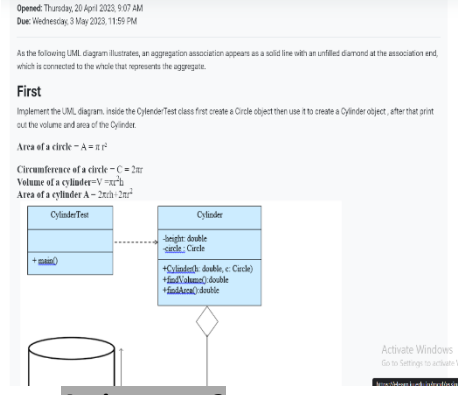
Lab14 /week 14

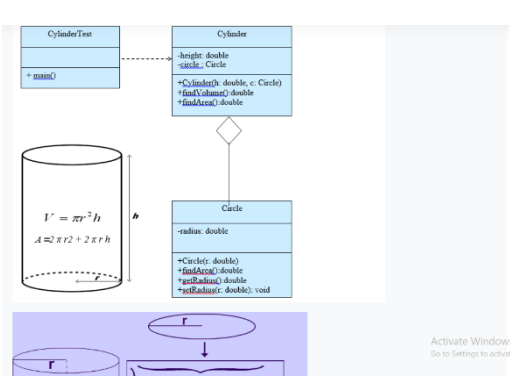
```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

// lab 14

namespace lab_14

{

   public class MorphingDogs

   {

      public static void Main(string[] args)

      {

         Dog myDog;

         String ans = "y";

         int choice;
```

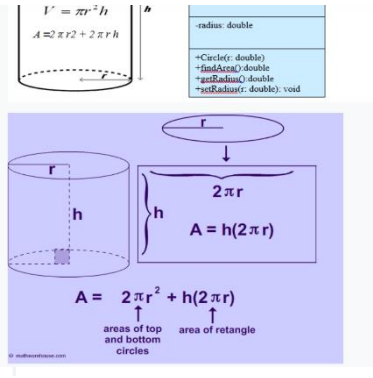
```
        while (ans == "y")
        {
            Console.WriteLine("Choose a Breed:\n1. Enter 1 for Labrador\n2. 2 for
Yorkshire");
            choice = int.Parse(Console.ReadLine());
            if (choice == 1)
                myDog = new Labrador("nana", "white");
            else
                myDog = new Yorkshire("momo");
            Console.WriteLine(myDog.speak());
            Console.Write("Try again? ");
            ans = Console.ReadLine();
        }
    }
    public interface Dog
    {
        string speak();
        int avgBreedWeight();
    }
    public class Labrador : Dog
    {
        protected internal string name;
        private string color;
        private static int breedWeight = 75;
        public Labrador(string name, string color)
        {
```

```
        this.name = name;
        this.color = color;
    }
    public string speak()
    {
        return "WOOF2";
    }
    public int avgBreedWeight()
    {
        return breedWeight;
    }
    public string getName()
    {
        return name;
    }
}
public class Yorkshire : Dog
{
    protected internal string name;
    private static int breedWeight = 75;
    public Yorkshire(string name)
    {
        this.name = name;
    }
    public string speak()
```

```
        {
            return "woof3";
        }
        public int avgBreedWeight()
        {
            return breedWeight;
        }
        public string getName()
        {
           return name;
        }
      }
   }
}
```







Assinement 2

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
// Assignment 2
namespace Assignment_2
{
    class Program
    {
        static void Main(string[] args)
        {
            double h = 3;
            double r = 3;
            Circle c1 = new Circle(r);
            Cylinder cc1 = new Cylinder(h, r);
            Console.WriteLine("Circumference of Circle =  " + c1.Circumference());
            Console.WriteLine("Area of Circle =  " + c1.Area());

            Console.WriteLine("Volume of a cylinder =  " + cc1.FindValume());
            Console.WriteLine("Area of a cylinder =  " + cc1.FindArea());
        }
    }
```

```
class Cylinder
{
    private double hieght, radius, Circumference;
    private double π = 3.14;

    public Cylinder(double h, double r)
    {
        hieght = h;
        this.radius = r;
    }
    public double FindValume()
    {
        return π * radius * radius * hieght;
    }
    public double FindArea()
    {
        return (Circumference * hieght) + (2 * π * radius * radius);
    }
}
class Circle
{
    private double radius;
    private double π = 3.14;

    public Circle(double r)
    {
        radius = r;

    }

    public double getRadius()
```

```
        {
            return radius;
        }
        public void setRadius(double r)
        {
            radius = r;
        }
        public double Area()
        {
            return (π * radius) * (π * radius); ;
        }
        public double Circumference()
        {
            return 2 * π * radius;
        }

    }
}
```

……

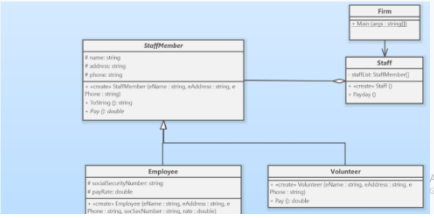
Assignment 3: Comprehensive Assignment.

Objectives

- Be able to use an abstract class
- Be able to override a virtual methods and implement an abstract method
- Be able to define abstract classes for the purpose of inheritance
- Be able to Implement abstract methods in the child class.
- Be able to consider inheritance design guidelines when designing for inheritance

Activities:

1. Write a C# program according to the following UML class diagram:



2. In the constructor of the staff class create a staffList array of size 6 and initialize the array with the following employee

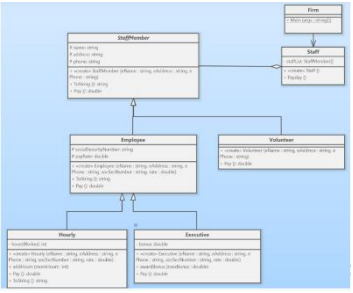
- Executive("Sam", "123 Main Line", "555-0469", "123-45-6789", 2423.07)
- Employee("Salma", "456 Off Line", "555-0101", "987-65-4321", 1246.15)
- Employee("Ali", "789 Off Rocker", "555-0000", "010-20-3040", 1169.23)
- Hourly("Diane", "678 Fifth Ave.", "555-0690", "958-47-3625", 10.55)
- Volunteer("Noor", "987 10th Ave.", "555-8374")
- Volunteer("Khalid", "321 maka st", "555-7282")

Then

- Use the method awardBonus to award 500Jd bonus to the executive Sam
- Use the method addHours to add 40 hours to the Hourly employee Diane

3. Write a code for the Payday() method of the Staff class to loop through the array elements to execute Pay() and ToString method on each employee

4. In the main method of the Firm class create a staff object called personal and call the Payday() on that object.



Assinement 3

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

// Assignment 3

namespace Assignment_3
{
    class Firm
    {
        static void Main(string[] args)
        {
            Staff personal;
            personal = new Staff();
            personal.payDay();
            Console.ReadKey();
        }
```

```
}
class Staff
{
    private StaffMember[] staffList;
    public Staff()
    {
        staffList = new StaffMember[6];
        staffList[0] = new Executive("Sam", "123 Main Line", "555-0469", "123-45-6789",
2423.07);
        staffList[1] = new Employee("Salma", "456 Off Line", "555-0101", "987-65-4321",
1246.15);
        staffList[2] = new Employee("Ali", "789 Off Rocker", "555-0000", "010-20-3040",
1169.23);
        staffList[3] = new Hourly("Diane", "678 Fifth Ave.", "555-0690", "958-47-3625",
10.55);
        staffList[4] = new Volunteer("Noor", "987 10th Ave.", "555-8374");
        staffList[5] = new Volunteer("Khalid", "321 maka st", "555-7282");
        ((Executive)staffList[0]).awardBonus(300);
        ((Hourly)staffList[3]).Addhours(70);
    }
    public void payDay()
    {
        for (int i = 0; i < 6; i++)
        {
            Console.WriteLine(staffList[i].ToString());
            Console.WriteLine(staffList[i].pay());
        }
    }
}
abstract class StaffMember
{
    protected string eName;
```

```
    protected string eAddress;
    protected string ePhone;
    public StaffMember(string name, string address, string phone)
    {
        eName = name;
        eAddress = address;
        this.ePhone = phone;
    }
    public override string ToString()
    {
        return "Name : " + eName + "      Address : " + eAddress + "      Phone :" + ePhone;
    }
    abstract public double pay();
}
class Employee : StaffMember
{
    protected string socSecNumber;
    protected double rate;
    public Employee(string name, string address, string phone, string socialSecurityNumber,
double payRate) : base(name, address, phone)
    {
        socSecNumber = socialSecurityNumber;
        rate = payRate;
    }
    public override string ToString()
    {
        return base.ToString() + "      socialSecurityNumber : " + socSecNumber;
    }
    public override double pay()
    {
        return rate;
```

```
    }
}
class Volunteer : StaffMember
{
    public Volunteer(string name, string address, string phone)
        : base(name, address, phone)
    {
    }
    public override double pay()
    {
        return 0;
    }
    public override string ToString()
    {
        return base.ToString();
    }
}
class Hourly : Employee
{
    protected int hoursworked;
    public Hourly(string name, string address, string phone, string socialSecurityNumber,
double payRate)
        : base(name, address, phone, socialSecurityNumber, payRate)
    {
    }

    public void Addhours(int moreHours)

    {
        hoursworked += moreHours;
    }
```

```
        public override double pay()
        {
            return base.pay() + hoursworked;
        }
        public override string ToString()
        {
            return base.ToString();
        }
    }
    class Executive : Employee
    {
        private double Bouns;
        public Executive(string name, string address, string phone, string socialSecurityNumber,
double payRate)
            : base(name, address, phone, socialSecurityNumber, payRate)
        {
        }
        public void awardBonus(double execBouns)
        {
            Bouns += execBouns;
        }
        public override double pay()
        {
            return base.pay() + Bouns;
        }
        public override string ToString()
        {
            return base.ToString();
        }
    }
}
```