2019033800122336 batch-A jenish Kubavat
Git repo: https://github.com/Jenishkubavat/dotnet-practicals

Assignment 4

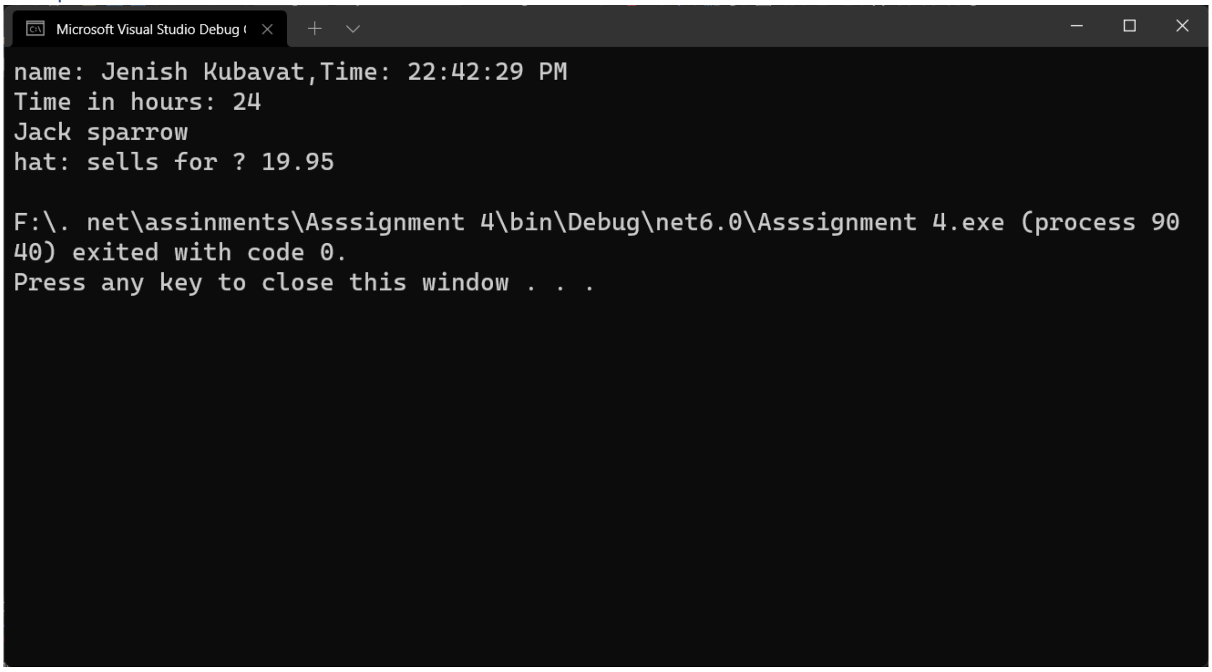
Properties

Code:

```
using System;
using Properties;
using Indexers;
namespace MainProgram
{
    class Program
     {
        public static void Main()
         {
             const string name = "Jenish Kubavat";
             Console.WriteLine($"name: {name},Time: 
{DateTime.Now.ToString("HH:mm:ss tt")}");
             TimePeriod t = new TimePeriod();
             t.Hours = 24;
             Console.WriteLine($"Time in hours: {t.Hours}");
             var person = new Person("Jack", "sparrow");
             Console.WriteLine(person.Name);
             var item = new SaleItem("hat", 19.95m);
             Console.WriteLine($"{item.Name}: sells for {item.Price:C2}");
         }


        }
      }



namespace Properties{
class TimePeriod
{
   private double _seconds;

   public double Hours
   {
       get { return _seconds / 3600; }
       set {
          if (value < 0 || value > 24)
             throw new ArgumentOutOfRangeException(
                   $"{nameof(value)} must be between 0 and 24.");

          _seconds = value * 3600;
       }
   }
}


public class Person
{
   private string _firstName;
   private string _lastName;

   public Person(string first, string last)
   {
      _firstName = first;
```

```
            _lastName = last;
        }

        public string Name => $"{_firstName} {_lastName}";
    }



    public class SaleItem
    {
        string _name;
        decimal _cost;

        public SaleItem(string name, decimal cost)
        {
            _name = name;
            _cost = cost;
        }

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

        public decimal Price
        {
            get => _cost;
            set => _cost = value;
        }
    }


        }
```

Output:



```
name: Jenish Kubavat,Time: 22:42:29 PM
Time in hours: 24
Jack sparrow
hat: sells for ? 19.95

F:\. net\assinments\Asssignment 4\bin\Debug\net6.0\Asssignment 4.exe (process 90
40) exited with code 0.
Press any key to close this window . . .
```

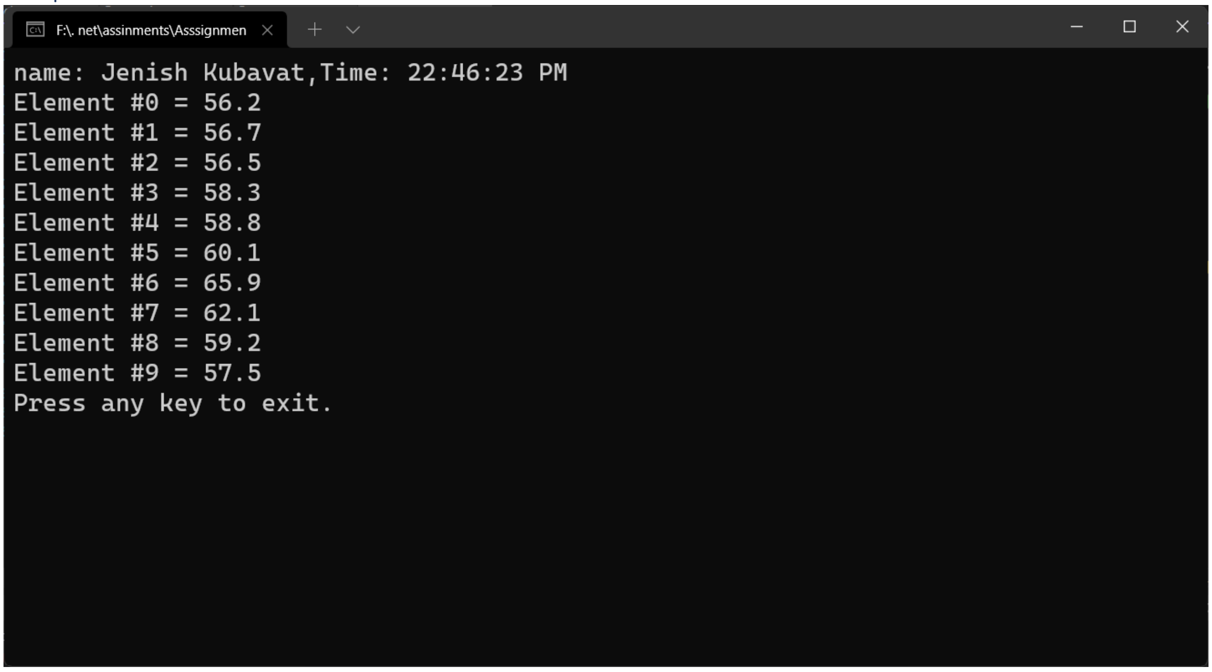
Indexers

Example 1

Code:

```
class Program
    {
        static void Main()
        {
            var tempRecord = new TempRecord();

            // Use the indexer's set accessor
            tempRecord[3] = 58.3F;
            tempRecord[5] = 60.1F;
            const string name = "Jenish Kubavat";
            Console.WriteLine($"name: {name},Time:
{DateTime.Now.ToString("HH:mm:ss tt")}");
            // Use the indexer's get accessor
            for (int i = 0; i < 10; i++)
            {
                Console.WriteLine($"Element #{i} = {tempRecord[i]}");
            }

            // Keep the console window open in debug mode.
            Console.WriteLine("Press any key to exit.");
            Console.ReadKey();
        }
       }
using System;
namespace Indexers
{
    // example one
    public class TempRecord
    {
        // Array of temperature values
        float[] temps = new float[10]
        {
        56.2F, 56.7F, 56.5F, 56.9F, 58.8F,
        61.3F, 65.9F, 62.1F, 59.2F, 57.5F
        };

        // To enable client code to validate input
        // when accessing your indexer.
        public int Length => temps.Length;

        // Indexer declaration.
        // If index is out of range, the temps array will throw the exception.
        public float this[int index]
        {
            get => temps[index];
            set => temps[index] = value;
        }
       }
    }
```

Output:



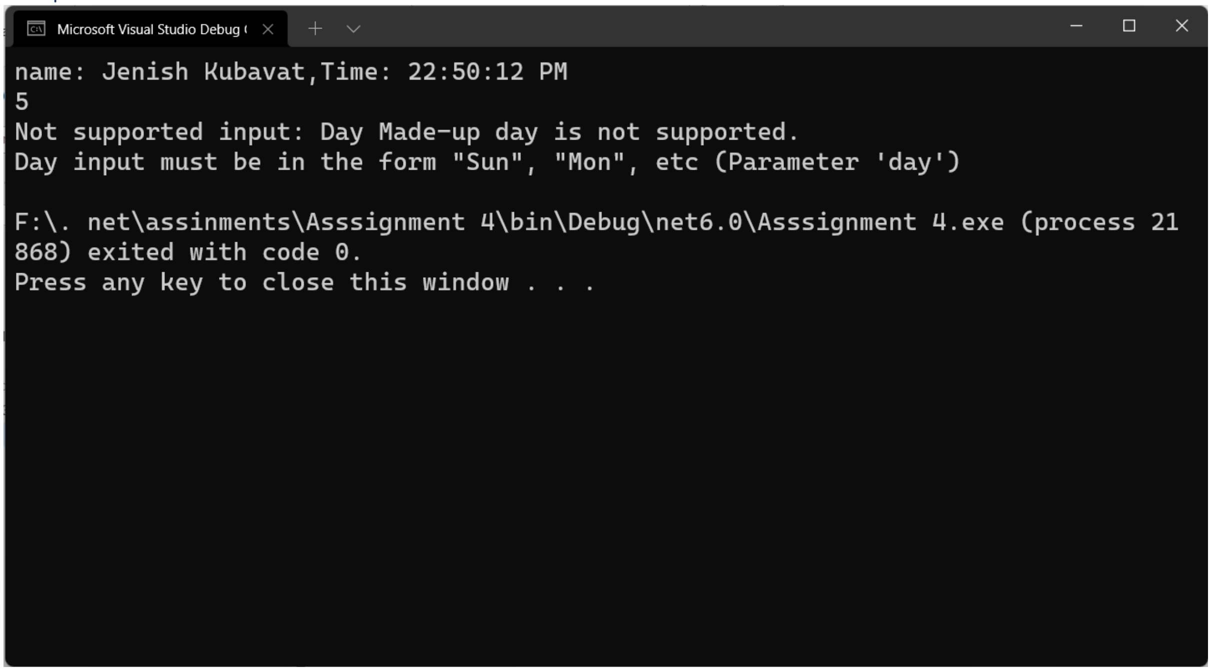
```
name: Jenish Kubavat,Time: 22:46:23 PM
Element #0 = 56.2
Element #1 = 56.7
Element #2 = 56.5
Element #3 = 58.3
Element #4 = 58.8
Element #5 = 60.1
Element #6 = 65.9
Element #7 = 62.1
Element #8 = 59.2
Element #9 = 57.5
Press any key to exit.
```

Example :3

Code:

```
class Program
    {
        static void Main()
        {
            const string name = "Jenish Kubavat";
            Console.WriteLine($"name: {name},Time: 
{DateTime.Now.ToString("HH:mm:ss tt")}");
            var week = new DayCollection();
            Console.WriteLine(week["Fri"]);

            try
            {
                Console.WriteLine(week["Made-up day"]);
            }
            catch (ArgumentOutOfRangeException e)
            {
                Console.WriteLine($"Not supported input: {e.Message}");
            }
        }

       }



using System;
namespace Indexers
{ class DayCollection
    {
        string[] days = { "Sun", "Mon", "Tues", "Wed", "Thurs", "Fri", "Sat" };

        // Indexer with only a get accessor with the expression-bodied 
definition:
        public int this[string day] => FindDayIndex(day);

        private int FindDayIndex(string day)
        {
            for (int j = 0; j < days.Length; j++)
            {
                if (days[j] == day)
                {
                    return j;
                }
            }

            throw new ArgumentOutOfRangeException(
                nameof(day),
                $"Day {day} is not supported.\nDay input must be in the form
\"Sun\", \"Mon\", etc");
        }
    }
}
```

Output:



```
name: Jenish Kubavat,Time: 22:50:12 PM
5
Not supported input: Day Made-up day is not supported.
Day input must be in the form "Sun", "Mon", etc (Parameter 'day')

F:\. net\assinments\Asssignment 4\bin\Debug\net6.0\Asssignment 4.exe (process 21
868) exited with code 0.
Press any key to close this window . . .
```

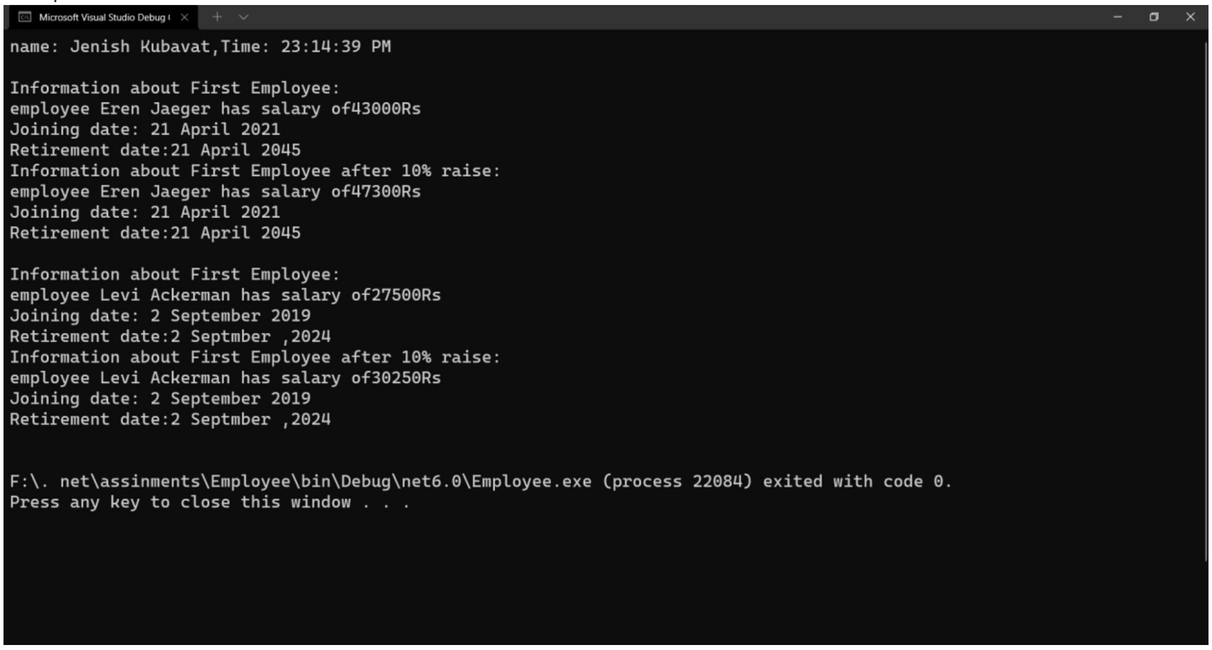
Employee

Code

```
using System;
using Employee;
namespace MainProgram
{
    class Program
    {
        public static void Main(string[] args)
        {
            PermanentEmployee employee1 = new PermanentEmployee("Eren", "Jaeger",
30000, 10000, 3000, 2000, "21 April 2021", "21 April 2045");
            PermanentEmployee employee2 = new PermanentEmployee("Levi",
"Ackerman", 20000, 5000, 2500, 1000, "2 September 2019", "2 Septmber ,2024");
            Console.WriteLine($"Information about First Employee:\n{employee1}");

            employee1.giveRaise(10);
            Console.WriteLine($"Information about First Employee after 10%
raise:\n{employee1}\n");
            Console.WriteLine($"Information about First Employee:\n{employee2}");
            employee2.giveRaise(10);
            Console.WriteLine($"Information about First Employee after 10%
raise:\n{employee2}\n");

        }
    }
   }

using System;
namespace Employee
{
     class Employee
    {
        internal string firstName;
        internal string lastName;
        internal double salary;

        internal Employee(string firstName ,string lastName ,double salary )
        {
            this.firstName =firstName;
            this.lastName =lastName;
            if (salary > 0)
            {
                this.salary = salary;
            }
            else
            {
                this.salary = 0.0;
            }
        }
        internal string FirstName
        {
            get { return firstName; }
            set { firstName = value; }
        }
        internal string LastName
        {
            get { return lastName; }
            set { lastName = value; }
        }
        internal double Salary
        {
            get { return salary; }
            set { salary = value; }
        }
        internal virtual double giveRaise(double raise)
        {
            return salary += ((salary * raise) / 100);

        }
        public override string ToString()
        {
            return $"employee {firstName} {lastName} has salary of{salary}Rs";
        }
    }
    class PermanentEmployee : Employee
    {
        private double hoursingRentAllowance;
        private double dearnessAllowance;
        private double providentFund;
        private string joiningDate;
        private string retirementDate;
```

```
         internal PermanentEmployee( string firstName,string lastName,double 
salary, double hoursingRentAllowance, double dearnessAllowance, double 
providentFund, string joiningDate, string retirementDate) : 
base(firstName,lastName, salary)
        {
            this.hoursingRentAllowance = hoursingRentAllowance;
            this.joiningDate = joiningDate;
            this.retirementDate = retirementDate;
            this.dearnessAllowance = dearnessAllowance;
            this.providentFund = providentFund;
            this.salary = salary+ hoursingRentAllowance+ dearnessAllowance;
        }
        internal double HoursingRentAllowance
        {
            get => hoursingRentAllowance;
            set => hoursingRentAllowance = value;
        }
        internal double DearnessAllowance
        {
            get => dearnessAllowance;
            set => hoursingRentAllowance = value;

        }
        internal double ProvidentFund
        {
            get => providentFund;
            set =>  providentFund = value;
        }
        internal string JoiningDate
        {
            get => joiningDate;
            set => joiningDate = value;

        }internal string RetirementDate
        {
            get => retirementDate;
            set => retirementDate = value;
        }
        internal override double giveRaise(double raise)
        {
            return (base.giveRaise(raise)+ dearnessAllowance + 
hoursingRentAllowance);

        }
        public override string ToString()
        {
            return $"{base.ToString()}\nJoining date: {joiningDate}\nRetirement 
date:{retirementDate}";
        }

    }
}
```

Output:



```
name: Jenish Kubavat,Time: 23:14:39 PM

Information about First Employee:
employee Eren Jaeger has salary of43000Rs
Joining date: 21 April 2021
Retirement date:21 April 2045
Information about First Employee after 10% raise:
employee Eren Jaeger has salary of47300Rs
Joining date: 21 April 2021
Retirement date:21 April 2045

Information about First Employee:
employee Levi Ackerman has salary of27500Rs
Joining date: 2 September 2019
Retirement date:2 Septmber ,2024
Information about First Employee after 10% raise:
employee Levi Ackerman has salary of30250Rs
Joining date: 2 September 2019
Retirement date:2 Septmber ,2024


F:\. net\assinments\Employee\bin\Debug\net6.0\Employee.exe (process 22084) exited with code 0.
Press any key to close this window . . .
```

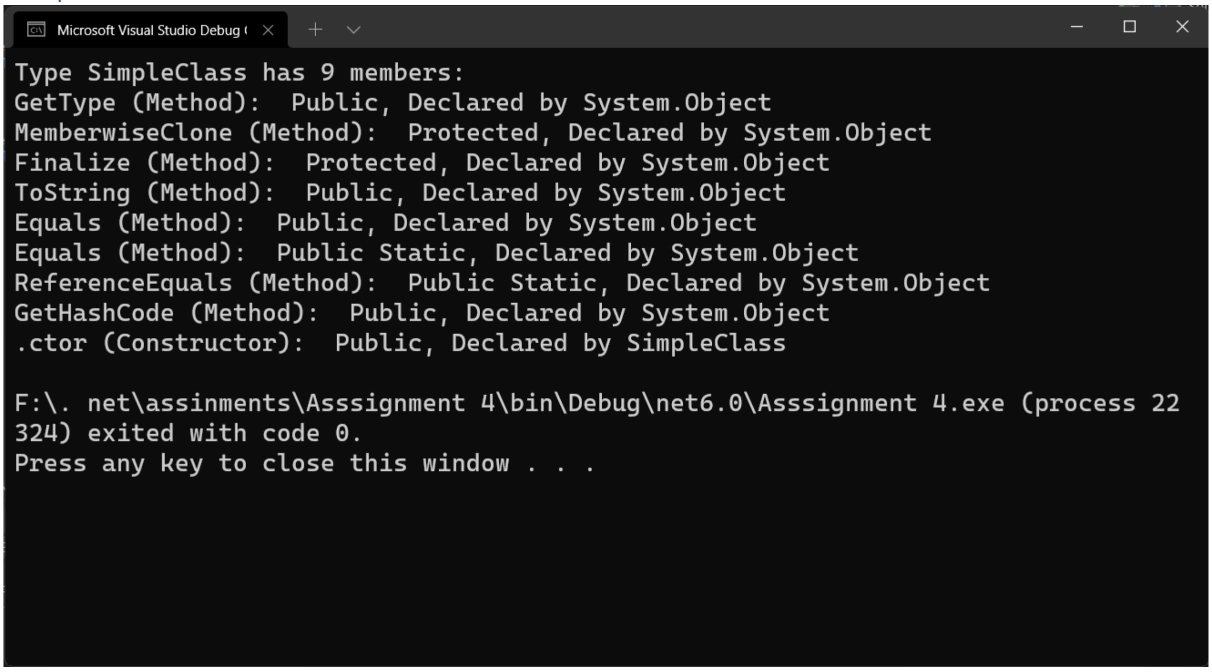
Sample class

Code:

```
using System;
using System.Reflection;


public class SimpleClass
{

}
public class SimpleClassExample
{
    public static void Main()
    {
        Type t = typeof(SimpleClass);
        BindingFlags flags = BindingFlags.Instance | BindingFlags.Static |
BindingFlags.Public |
                             BindingFlags.NonPublic |
BindingFlags.FlattenHierarchy;
        MemberInfo[] members = t.GetMembers(flags);
        Console.WriteLine($"Type {t.Name} has {members.Length} members: ");
        foreach (var member in members)
        {
            string access = "";
            string stat = "";
            var method = member as MethodBase;
            if (method != null)
            {
                if (method.IsPublic)
                    access = " Public";
                else if (method.IsPrivate)
                    access = " Private";
                else if (method.IsFamily)
                    access = " Protected";
                else if (method.IsAssembly)
                    access = " Internal";
                else if (method.IsFamilyOrAssembly)
```

```
                    access = " Protected Internal ";
                if (method.IsStatic)
                    stat = " Static";
            }
            var output = $"{member.Name} ({member.MemberType}): {access}{stat},
Declared by {member.DeclaringType}";
            Console.WriteLine(output);
        }
    }
}
```

Output:



```
Type SimpleClass has 9 members:
GetType (Method):  Public, Declared by System.Object
MemberwiseClone (Method):  Protected, Declared by System.Object
Finalize (Method):  Protected, Declared by System.Object
ToString (Method):  Public, Declared by System.Object
Equals (Method):  Public, Declared by System.Object
Equals (Method):  Public Static, Declared by System.Object
ReferenceEquals (Method):  Public Static, Declared by System.Object
GetHashCode (Method):  Public, Declared by System.Object
.ctor (Constructor):  Public, Declared by SimpleClass

F:\. net\assinments\Asssignment 4\bin\Debug\net6.0\Asssignment 4.exe (process 22
324) exited with code 0.
Press any key to close this window . . .
```