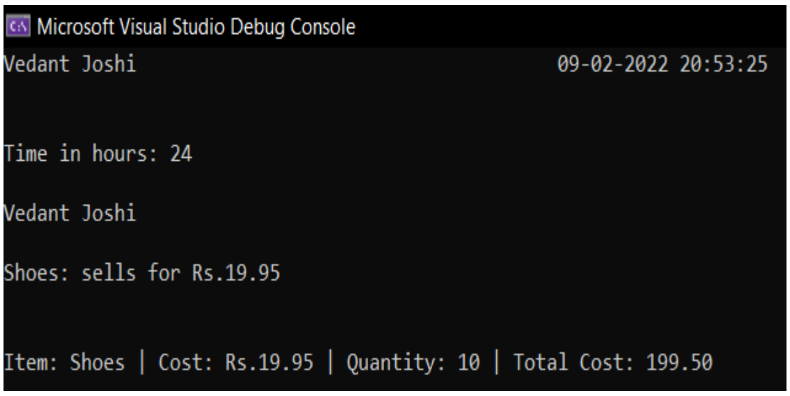
Practical – 4

Properties

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Pract3
{
    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;
            }
        }
    }

    internal 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;
        }

        public int Quantity { get; set; }
    }

    internal class Properties
    {
        public static void Main()
        {
            Console.WriteLine("Vedant Joshi
{0}\n\n",DateTime.Now);

            TimePeriod t = new TimePeriod();
            t.Hours = 24;

            Console.WriteLine($"Time in hours: {t.Hours}\n");


            var person = new Person("Vedant", "Joshi");
            Console.WriteLine(person.Name);

            var item = new SaleItem("Shoes", 19.95m);
            Console.WriteLine($"\n{item.Name}: sells for Rs.{item.Price}\n");

            item.Quantity = 10;

            Console.WriteLine($"\nItem: {item.Name} | Cost: Rs.{item.Price} |
Quantity: {item.Quantity} | Total Cost: {item.Quantity*item.Price}");

        }

    }
}
```

Output:



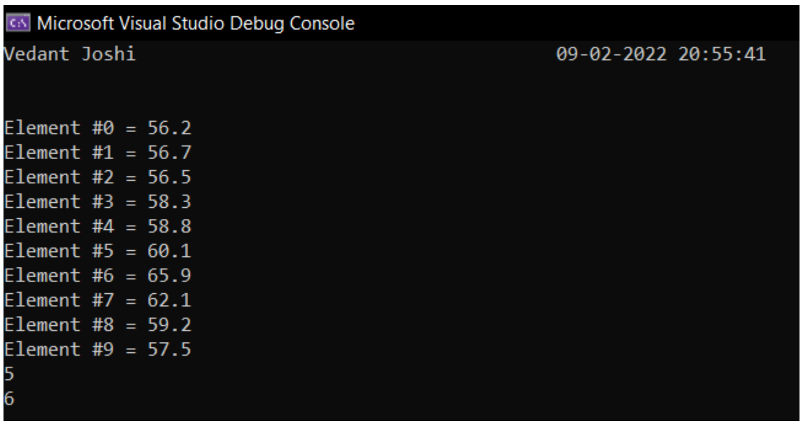
```
Vedant Joshi                              09-02-2022 20:53:25


Time in hours: 24

Vedant Joshi

Shoes: sells for Rs.19.95


Item: Shoes | Cost: Rs.19.95 | Quantity: 10 | Total Cost: 199.50
```

Indexers

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Indexers
{
    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;
        }
    }


    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");
        }
    }

    internal class Indexers {

        public static void Main()
        {
            Console.WriteLine("Vedant Joshi
{0}\n\n", DateTime.Now);

            var tempRecord = new TempRecord();

            // Use the indexer's set accessor
            tempRecord[3] = 58.3F;
            tempRecord[5] = 60.1F;

            // Use the indexer's get accessor
            for (int i = 0; i < 10; i++)
            {
                Console.WriteLine($"Element #{i} = {tempRecord[i]}");
            }

            var week = new DayCollection();
            Console.WriteLine(week["Fri"]);

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

    }
}
```

Output:



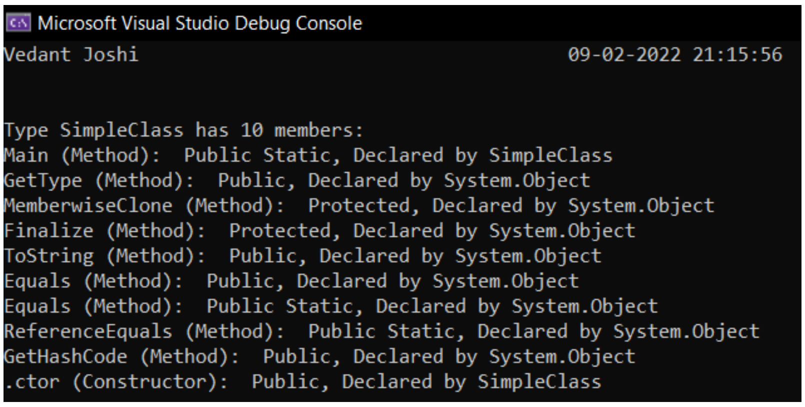
```
Vedant Joshi                                  09-02-2022 20:55:41


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
5
6
```

Methods and Its Types

Code:

```
using System;
using System.Reflection;

public class SimpleClass
{
    public static void Main()
    {
        Console.WriteLine("Vedant Joshi
{0}\n\n", DateTime.Now);
        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:



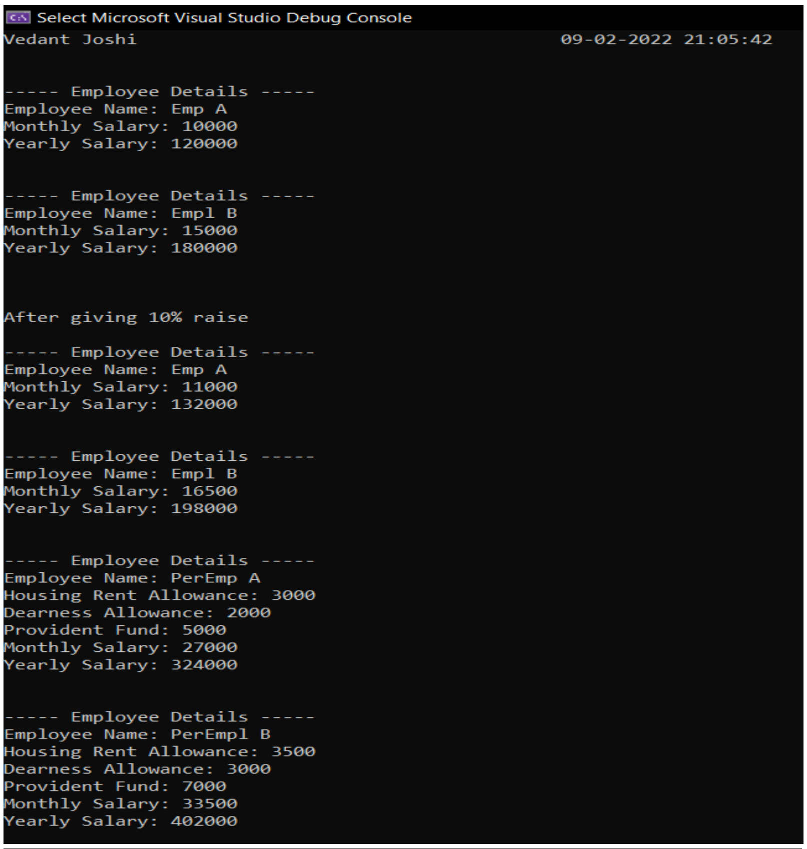
```
Vedant Joshi                                    09-02-2022 21:15:56


Type SimpleClass has 10 members:
Main (Method):  Public Static, Declared by SimpleClass
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
```

Employee Class

Code:

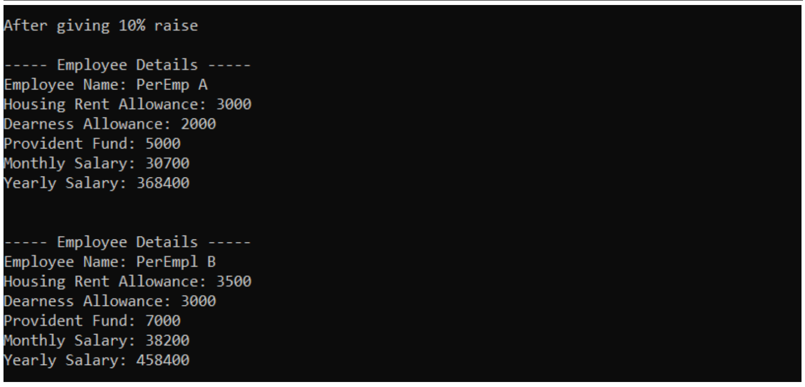
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Pract3
{
    public class Employee
    {
        private string _firstName, _lastName;
        private double _monthlySalary = 0.0, _yearlySalary = 0.0;

        public Employee(string firstName, string lastName, double monthlySalary)
        {
            this._firstName = firstName;
            this._lastName = lastName;
            if (monthlySalary < 0) this._monthlySalary = 0.0;
            else
            {
                this._monthlySalary = monthlySalary;
                _yearlySalary = monthlySalary * 12;
            }
        }

        public string FirstName
        {
            get
            {
                return this._firstName;
            }
            set
            {
                this._firstName = value;
            }
        }
        public string LastName
        {
            get
            {
                return this._lastName;
            }
            set
            {
                this._lastName = value;
            }
        }
        public double MonthlySalary
        {
            get
            {
                return this._monthlySalary;
            }
            set
            {
                if (value < 0)
                {
                    this._monthlySalary = 0.0;
                    this._yearlySalary = 0.0;
                }
                else
                {
                    this._monthlySalary = value;
                    this._yearlySalary = value*12;
                }
            }
        }

        public double YearlySalary
        {
            get => this._yearlySalary;
        }

        public virtual void giveRaise(double percentage)
        {
            this.MonthlySalary += (percentage/100) * this.MonthlySalary;
        }

        public override string ToString()
        {
            return $"----- Employee Details -----\nEmployee Name:
{this._firstName} {this._lastName}\nMonthly Salary: {this._monthlySalary}\nYearly
Salary: {this._yearlySalary}\n\n";
        }

    }

    public class PermanentEmployee : Employee
    {
        private double _hra, _da, _pf;
        private DateOnly _joiningDate, _retirementDate;
        public PermanentEmployee(string firstName, string lastName, double
monthlySalary, double hra, double da, double pf) : base(firstName, lastName,
(monthlySalary+da+hra+pf))
        {
            _hra = hra;
            _da = da;
            _pf = pf;
        }


        public string getAllowances()
        {
            return $"Housing Rent Allowance: {this._hra}\nDearness Allowance:
{this._da}";
        }

        public override void giveRaise(double percentage)
        {
            this.MonthlySalary += (percentage / 100) *
(this.MonthlySalary+this._da+this._hra+this._pf);
        }

        public override string ToString()
        {
            return $"----- Employee Details -----\nEmployee Name:
{this.FirstName} {this.LastName}\n{this.getAllowances()}\nProvident Fund:
{this._pf}\nMonthly Salary: {this.MonthlySalary}\nYearly Salary:
{this.YearlySalary}\n\n";
        }

        public string JoiningDate
        {
            get
            {
                return this._joiningDate.ToString();
            }
            set
            {
                this._joiningDate = DateOnly.ParseExact(value,"dd/mm/yyyy");
            }
        }

        public string RetirementDate
        {
            get
            {
                return this._retirementDate.ToString();
            }
            set
            {
                this._retirementDate = DateOnly.ParseExact(value, "dd/mm/yyyy");
            }
        }

        public double HRA { get => this._hra; }

        public double DA { get => this._da; }

        public double PF { get => this._pf; }

    }

        public class EmployeeTest
    {
        public static void Main()
        {

            Console.WriteLine("Vedant Joshi
{0}\n\n", DateTime.Now);

            Employee employee1 = new Employee("Emp", "A", 10000);
            Employee employee2 = new Employee("Empl", "B", 15000);

            Console.WriteLine(employee1);
            Console.WriteLine(employee2);

            employee1.giveRaise(10);
            employee2.giveRaise(10);
            Console.WriteLine("\nAfter giving 10% raise\n");

            Console.WriteLine(employee1);
            Console.WriteLine(employee2);


            PermanentEmployee perEmp1 = new PermanentEmployee("PerEmp", "A",
17000, 3000, 2000, 5000);
            PermanentEmployee perEmp2 = new PermanentEmployee("PerEmpl", "B",
20000, 3500, 3000, 7000);

            Console.WriteLine(perEmp1);
            Console.WriteLine(perEmp2);

            perEmp1.giveRaise(10);
            perEmp2.giveRaise(10);
            Console.WriteLine("\nAfter giving 10% raise\n");

            Console.WriteLine(perEmp1);
            Console.WriteLine(perEmp2);
            }
        }
}
```

Output:



Select Microsoft Visual Studio Debug Console

```
Vedant Joshi                                        09-02-2022 21:05:42


----- Employee Details -----
Employee Name: Emp A
Monthly Salary: 10000
Yearly Salary: 120000


----- Employee Details -----
Employee Name: Empl B
Monthly Salary: 15000
Yearly Salary: 180000



After giving 10% raise

----- Employee Details -----
Employee Name: Emp A
Monthly Salary: 11000
Yearly Salary: 132000


----- Employee Details -----
Employee Name: Empl B
Monthly Salary: 16500
Yearly Salary: 198000


----- Employee Details -----
Employee Name: PerEmp A
Housing Rent Allowance: 3000
Dearness Allowance: 2000
Provident Fund: 5000
Monthly Salary: 27000
Yearly Salary: 324000


----- Employee Details -----
Employee Name: PerEmpl B
Housing Rent Allowance: 3500
Dearness Allowance: 3000
Provident Fund: 7000
Monthly Salary: 33500
Yearly Salary: 402000
```



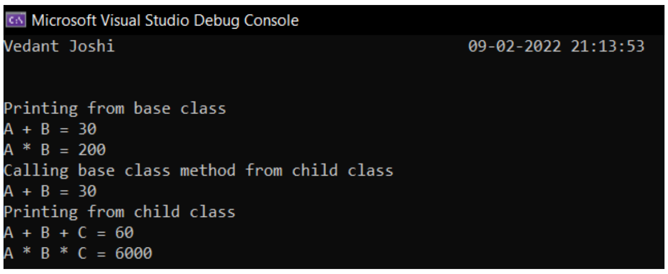
```
After giving 10% raise

----- Employee Details -----
Employee Name: PerEmp A
Housing Rent Allowance: 3000
Dearness Allowance: 2000
Provident Fund: 5000
Monthly Salary: 30700
Yearly Salary: 368400


----- Employee Details -----
Employee Name: PerEmpl B
Housing Rent Allowance: 3500
Dearness Allowance: 3000
Provident Fund: 7000
Monthly Salary: 38200
Yearly Salary: 458400
```

Method Overloading and Method Hiding

Code:

```
using System;

namespace Pract3{
    public class A
    {
        public int getSum(int x, int y) {
            return x + y;
        }

        public int getProduct(int x, int y)
        {
            return x * y;
        }
    }
    public class B : A
    {
        public int getSum(int x, int y, int z)
        {
            return x + y + z;
        }

        new public int getProduct(int x, int y, int z)
        {
            return x * y * z;
        }
    }

    public class Pract3
    {
        public static void Main()
        {
            Console.WriteLine("Vedant Joshi
{0}\n\n", DateTime.Now);

            A a = new A();
            B b = new B();
            Console.WriteLine("Printing from base class");
            Console.WriteLine("A + B = {0}", a.getSum(10,20));
            Console.WriteLine("A * B = {0}", a.getProduct(10, 20));
            Console.WriteLine("Calling base class method from child class");
            Console.WriteLine("A + B = {0}", b.getSum(10, 20));
            Console.WriteLine("Printing from child class");
            Console.WriteLine("A + B + C = {0}", b.getSum(10, 20, 30));
            Console.WriteLine("A * B * C = {0}", b.getProduct(10, 20, 30));

        }
    }
}
```

Output:



```
Vedant Joshi                                          09-02-2022 21:13:53


Printing from base class
A + B = 30
A * B = 200
Calling base class method from child class
A + B = 30
Printing from child class
A + B + C = 60
A * B * C = 6000
```