PRN:-2019033800120821

NAME:-PATEL PRIT SANJAYKUMAR

BATCH: - A

Roll no.: - 412052

Git repo: -

**Assignment 4: Classes and Inheritance**

Examples on Properties:

Properties.cs

namespace ne8{

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;

}

}

public class AutoImplementedSaleItem

{

public string Name

{ get; set; }

public decimal Price

{ get; set; }

}

}

Program4.cs

using System;

using ne8;

namespace MainProgram{

class Program{

public static void Main(){

const string name = "Bipin Mishra";

Console.WriteLine($"name: {name},Time: {DateTime.Now.ToString("HH:mm:ss tt")}");

TimePeriod t = new TimePeriod();

t.Hours = 1;

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

var person = new Person("Bipin", "Mishra");

Console.WriteLine(person.Name);

var item = new SaleItem("Shoes", 19.95m);

Console.WriteLine($"{item.Name}: sells for {item.Price:C2}");

var autoImplementedItem = new AutoImplementedSaleItem{ Name = "Socks", Price = 21.50m };

Console.WriteLine($"{autoImplementedItem.Name}: sells for {autoImplementedItem.Price:C2}");

}

}

}

Output:-



**Examples on Indexers:**

**Indexers.cs**

using System;

using Day = System.DayOfWeek;

namespace ne1{

public class TempRecord

{

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

};

public int Length => temps.Length;

public float this[int index]

{

get => temps[index];

set => temps[index] = value;

}

}

class DayCollection

{

string[] days = { "Sun", "Mon", "Tues", "Wed", "Thurs", "Fri", "Sat" };

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");

}

}

class DayOfWeekCollection

{

Day[] days =

{

Day.Sunday, Day.Monday, Day.Tuesday, Day.Wednesday,

Day.Thursday, Day.Friday, Day.Saturday

};

// Indexer with only a get accessor with the expression-bodied definition:

public int this[Day day] => FindDayIndex(day);

private int FindDayIndex(Day 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 a defined System.DayOfWeek value.");

}

}

}

**Program1.cs**

using System;

using ne1;

namespace ne2{

class Program{

public static void Main(string[] args){

//example-1

var tempRecord = new TempRecord();

tempRecord[3] = 58.3F;

tempRecord[5] = 60.1F;

for (int i = 0; i < 10; i++)

{

Console.WriteLine($"Element #{i} = {tempRecord[i]}");

}

//example-2

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}");

}

//example-3

var week2 = new DayOfWeekCollection();

Console.WriteLine(week2[DayOfWeek.Friday]);

try

{

Console.WriteLine(week2[(DayOfWeek)43]);

}

catch (ArgumentOutOfRangeException e)

{

Console.WriteLine($"Not supported input: {e.Message}");

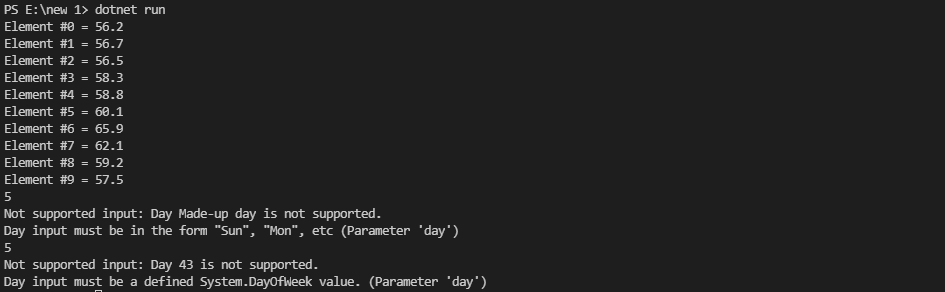
}

}

}

}

Output:-



**Simple Program to demonstrate Object:**

Program2.cs

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}";

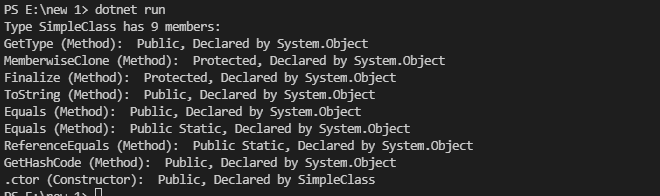
Console.WriteLine(output);

}

}

}

Output:-



**Simple Employee class to demonstrate inheritance:**

Employee.cs

using System;

namespace ne3{

public class Employee{

string \_firstName;

string \_lastName;

double \_monthlySalary;

public Employee(string firstName,string lastName,double monthlySalary){

\_firstName = firstName;

\_lastName = lastName;

\_monthlySalary = monthlySalary;

}

public string FirstName{

get{return \_firstName;}

set{\_firstName = value;}

}

public string LastName{

get{return \_lastName;}

set{\_lastName = value;}

}

public double MonthlySalary{

get{return \_monthlySalary;}

set {

\_monthlySalary = value<0?0.0:value;

}

}

public virtual void GiveRaise(double raise){

\_monthlySalary = \_monthlySalary + (\_monthlySalary\*raise/100);

}

public override string ToString(){

return $"First Name:{\_firstName}\nLast Name: {\_lastName}\nMonthly salary: {\_monthlySalary\*12}";

}

}

public class PermanentEmployee : Employee{

double \_housingRentAllowance;

double \_dearnessAllowance;

double \_providentFund;

string \_joiningDate;

string \_retirementDate;

public PermanentEmployee(string firstName,string lastName,string joiningDate,string retirementDate,double monthlySalary,double housingRentAllowance,double dearnessAllowance,double providentFund):base(firstName,lastName,monthlySalary){

this.\_housingRentAllowance = housingRentAllowance;

this.\_dearnessAllowance = dearnessAllowance;

this.\_providentFund = providentFund;

this.\_joiningDate = joiningDate;

this.\_retirementDate = retirementDate;

this.MonthlySalary = this.MonthlySalary + \_housingRentAllowance + \_dearnessAllowance;

}

public double HRA{

get=>\_housingRentAllowance;

}

public double DA{

get=>\_dearnessAllowance;

}

public double PF{

get=>\_providentFund;

}

public string JoiningDate{

get=>\_joiningDate;

set=>\_joiningDate=value;

}

public string RetirementDate{

get=>\_retirementDate;

set=>\_retirementDate=value;

}

public override void GiveRaise(double raise){

this.MonthlySalary = this.MonthlySalary + (this.MonthlySalary\*raise)/100 + \_dearnessAllowance + \_housingRentAllowance;

}

public override string ToString()

{

return $"{base.ToString()}\nJoining date: {\_joiningDate}\nRetirement date:{\_retirementDate}";

}

}

}

Program.cs

using System;

using ne3;

namespace ne4{

class Program{

public static void Main(string[] args){

PermanentEmployee e1 = new PermanentEmployee("Dev","Patel","23-08-79","01-03-21",55000.0,23000,1000,14000);

Console.WriteLine($"Information about First Employee:\n{e1}");

e1.GiveRaise(10.0);

Console.WriteLine($"Information about First Employee after 10% raise:\n{e1}");

PermanentEmployee e2 = new PermanentEmployee("Rahul","Prajapati","23-08-79","01-03-21",45000.0,21000,1200,12000);

Console.WriteLine($"Inforamtion about second Employee:\n{e2}");

e2.GiveRaise(10.0);

Console.WriteLine($"Inforamtion about second Employee after raise:\n{e2}");

}

}

}

Output:-

