using System.Collections;

using System.Security.Cryptography;

enum TimeFrame { Year, TwoYears, Long } //Определить тип TimeFrame - перечисление(enum) со значениями Year, TwoYears, Long.

class Program

{

static void Main(string[] args)

{

//1.Создать один объект типа ResearchTeam, преобразовать данные в текстовый вид с помощью метода ToShortString() и вывести данные

ResearchTeam MyTeam = new ResearchTeam();

Console.WriteLine(MyTeam.ToShortString());

//2.Вывести значения индексатора для значений индекса TimeFrame.Year, TimeFrame.TwoYears, TimeFrame.Long

Console.WriteLine(TimeFrame.Year);

Console.WriteLine(TimeFrame.TwoYears);

Console.WriteLine(TimeFrame.Long);

//3.Присвоить значения всем определенным в типе ResearchTeam свойствам, преобразовать данные в текстовый вид с помощью метода ToString() и вывести данные.

ResearchTeam Cats = new ResearchTeam("Пушистый кот", "НИИ Иследования котов", 1543264, TimeFrame.Year);

Console.WriteLine(Cats.ToShortString());

Console.WriteLine(Cats[TimeFrame.Year]);

Console.WriteLine(Cats[TimeFrame.TwoYears]);

Console.WriteLine(Cats[TimeFrame.Long]);

Cats.theme = "Коты";

Cats.nameofogr = "Пушистые";

Cats.number = 1155777;

Cats.resDur = TimeFrame.TwoYears;

Console.WriteLine(Cats.ToString());

//4.С помощью метода AddPapers (params Paper []) добавить элементы в список публикаций и вывести данные объекта ResearchTeam.

MyTeam.AddPapers(new Paper[2] { new Paper("лаба по c#", new Person("Дамир", "Гимадеев", new DateTime(1999, 09, 18)), new DateTime(2016, 11, 13)), new Paper() });

Console.WriteLine(MyTeam.ToString());

//5. Вывести значение свойства, которое возвращает ссылку на публикацию с самой поздней датой выхода;

Console.WriteLine(Cats.LastPublic);

//6.Сравнить время выполнения операций с элементами одномерного, двумерного прямоугольного и двумерного ступенчатого массивов с одинаковым числом элементов типа Paper.

Paper[] odnomer = new Paper[10];

Paper[,] dvumer = new Paper[2, 5];

Paper[][] stup = new Paper[4][];

stup[0] = new Paper[1];

stup[1] = new Paper[2];

stup[2] = new Paper[3];

stup[3] = new Paper[4];

long milliseconds = DateTime.Now.Ticks;

for (int p = 0; p < 10000; p++)

{

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

{

odnomer[i] = new Paper();

}

}

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

{

Console.WriteLine(odnomer[i] + " ");

}

milliseconds = DateTime.Now.Ticks - milliseconds;

Console.WriteLine("odnomer: " + milliseconds + "\n");

Console.WriteLine("\n");

milliseconds = DateTime.Now.Ticks;

for (int p = 0; p < 10000; p++)

{

for (int i = 0; i < 2; i++)

{

for (int j = 0; j < 5; j++)

{

dvumer[i, j] = new Paper();

}

}

}

for (int i = 0; i < 2; i++)

{

for (int j = 0; j < 5; j++)

{

Console.WriteLine(dvumer[i, j] + " ");

}

}

milliseconds = DateTime.Now.Ticks - milliseconds;

Console.WriteLine("dvumer: " + milliseconds + "\n");

Console.WriteLine("\n");

milliseconds = DateTime.Now.Ticks;

for (int p = 0; p < 10000; p++)

{

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < i + 1; j++)

{

stup[i][j] = new Paper();

}

}

}

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < i + 1; j++)

{

Console.WriteLine(stup[i][j] + " ");

}

}

milliseconds = DateTime.Now.Ticks - milliseconds;

Console.WriteLine("stup: " + milliseconds + "\n");

Console.ReadKey();

}

}

class Paper

{

public string NameP { get; set; }

public Person Author { get; set; }

public DateTime Data { get; set; }

public Paper(string name, Person author, DateTime data)

{

NameP = name;

Author = author;

Data = data;

}

// конструктор без параметров, инициализирующий все свойства класса некоторыми значениями по умолчанию

public Paper() : this("Мартин Иден", new Person(), new DateTime(1889, 6, 1))

{ }

// перегруженную версию виртуального метода string To-String() для формирования строки со значениями всех полей класса

public override string ToString()

{

return string.Format("Author {0} write book {1}. Data = {2}", Author, NameP, Data);

}

}

class Person//Определить класс Person, который имеет

{

private string \_Name; //закрытое поле типа string, в котором хранится имя;

private string \_Surname;//закрытое поле типа string, в котором хранится фамилия

private System.DateTime \_Birthday;//закрытое поле типа System.DateTime для даты рождения

//конструктор c тремя параметрами типа string, string, DateTime для инициализации всех полей класса

public Person(string Name, string Surname, DateTime Birthday)

{

\_Name = Name;

\_Surname = Surname;

\_Birthday = Birthday;

}

//конструктор без параметров, инициализирующий все поля класса некоторыми значениями по умолчанию.

public Person() : this("Джек", "Лондон", new DateTime(1998, 11, 12))

{

}

// В классе Person определить свойства c методами get и set:

//свойство типа string для доступа к полю с именем;

public string PersName

{

get

{

return \_Name;

}

}

// свойство типа string для доступа к полю с фамилией;

public string PersonSurname

{

get

{

return \_Surname;

}

}

//свойство типа DateTime для доступа к полю с датой рождения;

public DateTime PersonBirthday

{

get

{

return \_Birthday;

}

}

//свойство типа int c методами get и set для получения информации(get) и изменения(set) года рождения в закрытом поле типа DateTime, в котором хранится дата рождения.

public int intBirthday

{

get

{

return Convert.ToInt32(\_Birthday);

}

set

{

\_Birthday = Convert.ToDateTime(value);

}

}

//перегруженную(override) версию виртуального метода string ToString() для формирования строки со значениями всех полей класса;

public override string ToString()

{

return string.Format("{0} {1} was born {2}", \_Name, \_Surname, \_Birthday);

}

// виртуальный метод string ToShortString(), который возвращает строку, содержащую только имя и фамилию.

public string ToShortString()

{

return string.Format("{0} {1}", \_Name, \_Surname);

}

}

class ResearchTeam //Определить класс ResearchTeam, который имеет

{

private string Theme; // закрытое поле типа string c названием темы исследований;

private string NameOfOrganisation; // закрытое поле типа string с названием организации;

private int RegNumber; //• закрытое поле типа int – регистрационный номер;

private TimeFrame ResearchDuration; //закрытое поле типа TimeFrame для информации о продолжительности исследований;

private Paper[] ListOfPubl; // закрытое поле типа Paper[], в котором хранится список публикаций.

System.Collections.ArrayList Publications = new ArrayList(); //публикаций список

//В классе ResearchTeam определить конструкторы:

//конструктор c параметрами типа string, string, int, TimeFrame для инициализации соответствующих полей класса;

public ResearchTeam(string InvestigationTheme, string Organisation, int RegistrationNumber, TimeFrame InvestigationDuration)

{

ResearchDuration = InvestigationDuration;

Theme = InvestigationTheme;

RegNumber = RegistrationNumber;

NameOfOrganisation = Organisation;

}

// конструктор без параметров, инициализирующий поля класса значениями по умолчанию.

public ResearchTeam() : this("Котики", "КГЭУ", 13, TimeFrame.Year) { }

//В классе ResearchTeam определить свойства c методами get и set

//свойство типа string для доступа к полю с названием темы исследований

public string theme

{

get

{

return Theme;

}

set

{

Theme = value;

}

}

// свойство типа string для доступа к полю с названием организации

public string nameofogr

{

get

{

return NameOfOrganisation;

}

set

{

NameOfOrganisation = value;

}

}

//свойство типа int для доступа к полю с номером регистрации

public int number

{

get

{

return RegNumber;

}

set

{

RegNumber = value;

}

}

//Cвойство типа TimeFrame для доступа к полю с продолжительностью исследований

public TimeFrame resDur

{

get

{

return ResearchDuration;

}

set

{

ResearchDuration = value;

}

}

//Cвойство типа Paper[] для доступа к полю со списком публикаций по те-ме исследований

public IReadOnlyList<Paper> listOfPublications

{

get

{

return ListOfPubl;

}

}

//свойство типа Paper ( только с методом get), которое возвращает ссылку на публикацию с самой поздней датой выхода; если список публикаций пустой, свойство возвращает значение null;

public Paper LastPublic

{

get

{

if (Publications.Count == 0)

{

return null;

}

int MaxIndex = 0;

DateTime MaxDateTime = ((Paper)Publications[0]).Data;

for (int i = 0; i < Publications.Count; i++)

{

if (((Paper)Publications[i]).Data > MaxDateTime)

{

MaxIndex = i;

MaxDateTime = ((Paper)Publications[i]).Data;

}

}

Console.WriteLine("paper"+Publications[MaxIndex]);

return (Paper)Publications[MaxIndex];

}

}

//индексатор булевского типа (только с методом get) с одним параметром типа TimeFrame; значение индексатора равно true, если значение поля с информацией о продолжительности исследований совпадает со значением индекса, и false в противном случае;

public bool this[TimeFrame timeFrame]

{

get

{

bool rez;

if (timeFrame == ResearchDuration)

{

rez = true;

}

else

{

rez = false;

}

return rez;

}

}

//метод void AddPapers(params Paper[] ) для добавления элементов в список публикаций

public void AddPapers(params Paper[] AdditionalPapers)

{

Publications.AddRange(AdditionalPapers);

}

//перегруженную версию виртуального метода string ToString() для формирования строки со значениями всех полей класса, включая список публикаций;

//public override string ToString()

//{

// return string.Format("\nTheme: {0}\nNameOfOrg: {1}\nNumberOfRed: {2}\nLast: {3}\nPublications: {4} ", Theme, RegNumber, ResearchDuration, Publications);

//}

public override string ToString()

{

string stringListOfPublications = "";

foreach (Paper pap in Publications)

{

stringListOfPublications += pap.ToString() + "\r\n";

}

return base.ToString() + string.Format("\r\n Тема: {0}, Длительность: {1} \r\n Организация:{2} \r\n Публикация: {3}", Theme, ResearchDuration, NameOfOrganisation, stringListOfPublications);

}

//виртуальный метод string ToShortString(), который формирует строку со значениями всех полей класса без списка публикаций.

public string ToShortString()

{

return base.ToString() + string.Format("\r\n Theme: {0}, Duration: {1} \r\n ", Theme, ResearchDuration);

}

}