**Самостоятельная работа 3**

**Тема: Реализация иерархий классов**

**Задание 1**

Создайте в Visual Studio проект консольного типа.

Реализуйте в проекте иерархии классов, представленные в задании ниже.

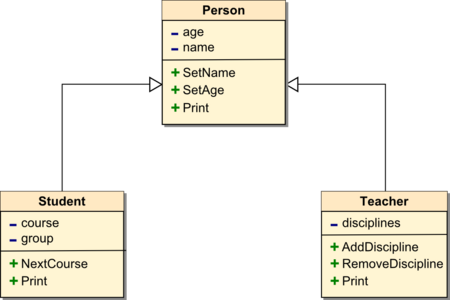
Требования:  
Каждый класс обязан содержать конструктор с параметрами.  
Каждый класс обязан переопределять метод ToString() так, чтобы он создавал строку с полной информацией об объекте - экземпляре класса.

Каждый класс определять в отдельном файле

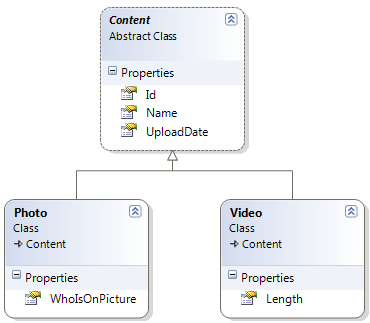
Каждый набор файлов с классами, связанными наследованием, размещать в отдельном подкаталоге проекта. Подкаталог нужно называть по шаблону: Hierarchy\_номер\_иерархии, например: Hierarchy01.

Иерархии классов

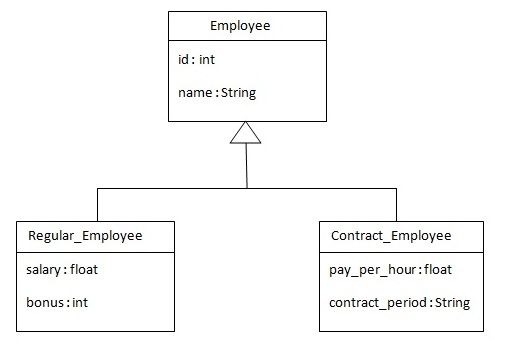
Иерархия 1



Иерархия 2



Иерархия 3



В методе Main() создайте массив типа Person на четыре элемента, добавьте в массив два экземпляра класса Student и два экземпляра класса Teacher, каждый экземпляр должен быть проинициализирован с помощью конструкторов адекватными, читаемыми значениями его полей/свойств.

Используя foreach, обойдите элементы массива, получите строку с полной информацией о каждом элементе массива и выполните вывод этой информации в консоль, разделяя выводимую информацию об элементах одной пустой строкой.

Листинг Person.cs

using System;

namespace HierarchyClasses.Hierarchy1

{

class Person

{

private int age;

public int Age {set => age = value;}

private string name;

public string Name { set => name = value; }

public Person(int age, string name)

{

Age = age;

Name = name;

}

public virtual void Print()

{

Console.WriteLine(ToString());

}

public override string ToString()

{

return $"Name: {name} Age: {age}";

}

}

}

Листинг Student.cs

using System;

namespace HierarchyClasses.Hierarchy1

{

class Student : Person

{

private int course;

private string group;

public Student(int course1, string group1, int age, string name)

:base (age, name)

{

course = course1;

group = group1;

}

public void NextCourse()

{

course++;

}

public override string ToString()

{

return base.ToString() + $"Course: {course} Group {group}";

}

public override void Print()

{

Console.WriteLine(ToString());

}

}

}

Листинг Teacher.cs

using System;

namespace HierarchyClasses.Hierarchy1

{

class Teacher : Person

{

private string[] disciplines;

public Teacher(int age, string name, params string[] disciplines)

: base (age, name)

{

this.disciplines = disciplines;

}

public void AddDiscipline(string discipline)

{

Array.Resize(ref disciplines, disciplines.Length + 1);

disciplines[disciplines.Length - 1] = discipline;

}

public void RemoveDiscipline(string discipline)

{

for (int i = 0; i < disciplines.Length; i++)

{

if (disciplines[i] == discipline)

{

disciplines[i] = disciplines[disciplines.Length - 1];

Array.Resize(ref disciplines, disciplines.Length - 1);

break;

}

}

}

public override string ToString()

{

string res = base.ToString() + " Disciplines: ";

foreach (string item in disciplines)

{

res += $"{item}";

if (item != disciplines[disciplines.Length - 1])

res += ", ";

}

return res;

}

public override void Print()

{

Console.WriteLine(ToString());

}

}

}

Листинг Content.cs

using System;

namespace HierarchyClasses.Hierarchy2

{

abstract class Content

{

public int Id { get; set; }

public string Name { get; set; }

public DateTime UploadDate { get; set; }

public Content(int id, string name, DateTime uploadDate)

{

Id = id;

Name = name;

UploadDate = uploadDate;

}

public override string ToString()

{

return $"Id: {Id} Name: {Name} Upload date: {UploadDate} ";

}

}

}

Листинг Photo.cs

using System;

namespace HierarchyClasses.Hierarchy2

{

class Photo : Content

{

public string WhoIsOnPicture { get; set; }

public Photo(int id, string name, DateTime uploadDate, string whoIsOnPicture = null)

:base (id, name, uploadDate)

{

WhoIsOnPicture = whoIsOnPicture;

}

public override string ToString()

{

if (WhoIsOnPicture != null)

return base.ToString() + $"Person On Picture: {WhoIsOnPicture}";

return base.ToString();

}

}

}

Листинг Video.cs

using System;

namespace HierarchyClasses.Hierarchy2

{

class Video : Content

{

public TimeSpan Length { get; set; }

public Video(int id, string name, DateTime uploadDate, TimeSpan length)

: base(id, name, uploadDate)

{

Length = length;

}

public override string ToString()

{

return base.ToString() + $"Length: {Length}";

}

}

}

Листинг Employee.cs

using System;

namespace HierarchyClasses.Hierarchy3

{

class Employee

{

public int Id { get; set; }

public string Name { get; set; }

public Employee(int id, string name)

{

Id = id;

Name = name;

}

public override string ToString()

{

return $"ID: {Id} Name: {Name}";

}

}

}

Листинг Regular\_Employee.cs

using System;

namespace HierarchyClasses.Hierarchy3

{

class RegularEmployee: Employee

{

public double Salary { get; set; }

public int Bonus { get; set; }

public RegularEmployee(int id, string name, double salary, int bonus)

:base(id, name)

{

Salary = salary;

Bonus = bonus;

}

public override string ToString()

{

return base.ToString() + $" Salary: {Salary} Bonus: {Bonus}";

}

}

}

Листинг Conrtact\_Employee.cs

using System;

namespace HierarchyClasses.Hierarchy3

{

class ContractEmployee: Employee

{

public double PayPerHour { get; set; }

public string ContractPeriod { get; set; }

public ContractEmployee(int id, string name, double payPerHour, string contractPeriod)

:base(id, name)

{

PayPerHour = payPerHour;

ContractPeriod = contractPeriod;

}

public override string ToString()

{

return base.ToString() + $" Pay per hour: {PayPerHour} Contract period: {ContractPeriod}";

}

}

}

Листинг Program.cs

using System;

using HierarchyClasses.Hierarchy1;

namespace HierarchyClasses

{

class Program

{

static void Main(string[] args)

{

Person[] people = {new Student(2, "ISP-02", 17, "Makushkin E.V."),

new Student(2, "ISP-01", 17, "Vasura K.A." ),

new Teacher(50, "Guslev M.V.", "MDK 01.01", "MDK 01.02", "Discrete Math"),

new Teacher(20, "Zhitkov N.S.", "AAS", "Computer's Network")};

foreach (Person item in people)

{

Console.WriteLine(item.ToString());

}

Console.ReadKey();

}

}

}