Самостоятельная работа от 20.02.2020

Задание 1

Создайте в Visual Studio проект консольного типа.

Определите в проекте ряд классов, перечисленных ниже в задании.

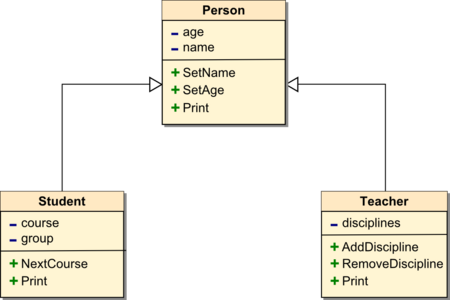
Требования:  
Каждый класс обязан содержать конструктор с параметрами.  
Каждый класс обязан переопределять метод ToString() так, чтобы он создавал строку с полной информацией об объекте - экземпляре класса.

Каждый класс определять в отдельном файле

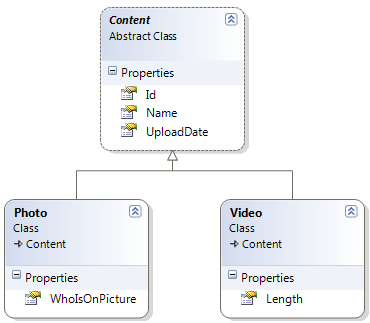
Каждый набор файлов с классами, связанными наследованием, размещать в отдельном подкаталоге проекта. Подкаталог нужно называть по шаблону: Hierarchy\_номер\_иерархии, например: Hierarchy01.

Иерархии классов

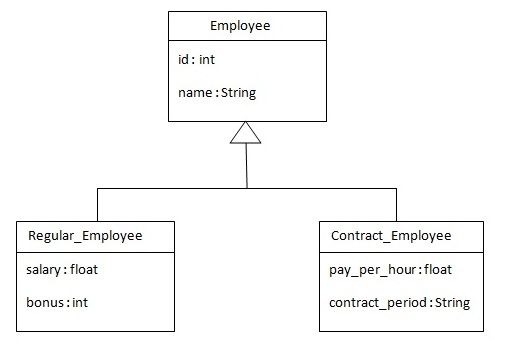
Иерархия 1



Иерархия 2



Иерархия 3



Листинг класса Program:

static void Main()

{

Person[] persons = new Person[4]

{

new Student("Петров", 15, 1, "ИСП-1"),

new Student("Иванов", 17, 2, "ИСП-2"),

new Teacher("Лобанов", 67, new string[] { "Математика", "Химия" }),

new Teacher("Казанов", 26, new string[] { "Программирование", "Информатика" })

};

foreach (Person elem in persons)

Console.WriteLine(elem);

Console.ReadKey();

}

Листинг класса Person:

private string name;

private int age;

public Person(string name, int age)

{

this.name = name;

this.age = age;

}

public void SetName(string name)

{

if (name == null || name.Length < 1)

return;

this.name = name;

}

public void SetAge(int age)

{

if (age < 1)

return;

this.age = age;

}

public virtual void Print()

{

Console.WriteLine(ToString());

}

public override string ToString()

{

return $"Имя: {name}; Возраст: {age}";

}

Листинг класса Student:

private string group;

private int course;

public Student(string name, int age, int course, string group)

: base(name, age)

{

this.course = course;

this.group = group;

}

public void NextCourse()

{

course++;

}

public override void Print()

{

Console.WriteLine(ToString());

}

public override string ToString()

{

return $"{base.ToString()}; Группа: {group}; Курс: {course}";

}

Листинг класса Teacher:

private List<string> disciplines;

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

: base(name, age)

{

if (disciplines != null)

this.disciplines = new List<string>(disciplines);

}

public void AddDiscipline(string discipline)

{

if (discipline == null || discipline.Length < 1)

return;

disciplines.Add(discipline);

}

public void RemoveDiscipline(string discipline)

{

if (discipline == null || discipline.Length < 1)

return;

if (disciplines.Contains(discipline))

disciplines.Remove(discipline);

}

public override void Print()

{

Console.WriteLine(ToString());

}

public override string ToString()

{

return $"{base.ToString()}; Дисциплины: {(disciplines != null ? string.Join(

", ", disciplines) : "")}";

}

Листинг класса Content:

private static int counter = 0;

private int id;

private string name;

private DateTime uploadtime;

public Content(string name, DateTime uploadtime)

{

id = counter;

counter++;

this.name = name;

this.uploadtime = uploadtime;

}

public override string ToString()

{

return $"ID: {id}; Имя: {name}; Время загрузки: {uploadtime}";

}

Листинг класса Photo:

private string whoisonpicture;

public Photo(string name, DateTime uploadtime, string whoisonpicture)

: base(name, uploadtime)

{

this.whoisonpicture = whoisonpicture;

}

public override string ToString()

{

return $"{base.ToString()}; Загрузил: {whoisonpicture}";

}

Листинг класса Video:

private double length;

public Video(string name, DateTime uploadtime, double length)

: base(name, uploadtime)

{

this.length = length;

}

public override string ToString()

{

return $"{base.ToString()}; Продолжительность: {length}";

}

Листинг класса Employee:

private static int counter = 0;

private int id;

private string name;

public Employee(string name)

{

id = counter;

counter++;

this.name = name;

}

public override string ToString()

{

return $"ID: {id}; Имя: {name}";

}

Листинг класса ContractEmployee:

private double payPerHour;

private string contactPeriod;

public ContractEmployee(string name, double payPerHour, string contactPeriod)

: base(name)

{

this.payPerHour = payPerHour;

this.contactPeriod = contactPeriod;

}

public override string ToString()

{

return $"{base.ToString()}; Почасовая оплата: {payPerHour}; Период контракта: {contactPeriod}";

}

Листинг класса RegularEmployee:

private double salary;

private double bonus;

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

: base(name)

{

this.salary = salary;

this.bonus = bonus;

}

public override string ToString()

{

return $"{base.ToString()}; Зарплата: {salary}; Премия: {bonus}";

}

Результат работы программы:

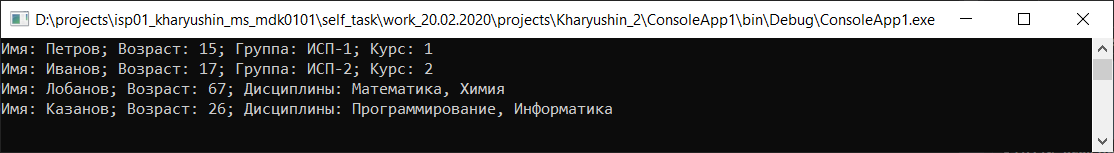


Рисунок 1 - Результат работы программы задания 1

Задание 2

Создайте новый проект консольного типа, определите в нём следующий набор классов:

* Сделайте класс User, в котором будут следующие protected поля - name (имя), age (возраст), public методы setName, getName, setAge, getAge.
* Сделайте класс Worker, который наследует от класса User и вносит дополнительное private поле salary (зарплата), а также методы public getSalary и setSalary.
* Создайте объект этого класса 'Иван', возраст 25, зарплата 1000. Создайте второй объект этого класса 'Вася', возраст 26, зарплата 2000. Найдите сумму зарплата Ивана и Васи.
* Сделайте класс Student, который наследует от класса User и вносит дополнительные private поля стипендия, курс, а также геттеры и сеттеры для них.
* Сделайте класс Driver (Водитель), который будет наследоваться от класса Worker из предыдущей задачи. Этот метод должен вносить следующие private поля: водительский стаж, категория вождения (A, B, C).

Листинг класса Program:

static void Main()

{

Worker worker01 = new Worker("Иван", 25, 1000);

Worker worker02 = new Worker("Вася", 26, 2000);

Console.WriteLine($"Сумма зарплат: {worker01.GetSalary() + worker02.GetSalary()}");

Console.ReadKey();

}

Листинг класса User:

protected string name;

protected int age;

public User(string name, int age)

{

this.name = name;

this.age = age;

}

public void SetName(string name)

{

if (name == null || name.Length < 1)

throw new ArgumentException("Имя должно быть больше одного символа!");

this.name = name;

}

public string GetName()

{

return name;

}

public void SetAge(int age)

{

if (age < 1)

throw new ArgumentException("Возраст должен быть больше 1 года!");

this.age = age;

}

public int GetAge()

{

return age;

}

Листинг класса Student:

private double scholarships;

private int course;

public Student(string name, int age, double scholarships, int course)

: base(name, age)

{

this.scholarships = scholarships;

this.course = course;

}

public double GetScholarships()

{

return scholarships;

}

public void SetScholarships(double scholarships)

{

if (scholarships < 0)

throw new ArgumentException("Степендия должна быть выше, чем ноль!");

this.scholarships = scholarships;

}

public double GetCourse()

{

return course;

}

public void SetCourse(int course)

{

if (course < 1)

throw new ArgumentException("Курс должен быть выше, чем ноль!");

this.course = course;

}

Листинг класса Worker:

private double salary;

public Worker(string name, int age, double salary)

: base(name, age)

{

this.salary = salary;

}

public void SetSalary(double salary)

{

if (salary < 0)

throw new ArgumentException("Зарплата должна быть выше, чем ноль!");

this.salary = salary;

}

public double GetSalary()

{

return salary;

}

Листинг класса Driver:

private int experience;

private char drivingCategory;

public Driver(string name, int age, double salary, int experience,

char drivingCategory)

: base(name, age, salary)

{

this.experience = experience;

this.drivingCategory = drivingCategory;

}

Результат работы программы:

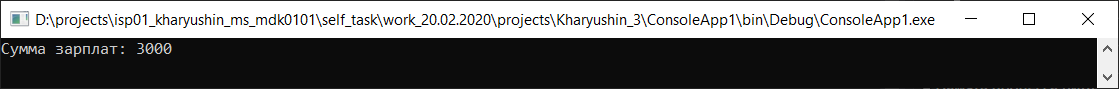


Рисунок 2 - Результат работы программы задания 2