**Самостоятельная работа 3**

**Тема: Реализация иерархий классов**

**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 зарплат 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 User.cs

using System;

namespace InheritanceClasses\_17.\_02.\_20

{

class User

{

protected string name;

protected int age;

public void GetName(string value)

{

name = value;

}

public string GetName()

{

return name;

}

public void GetAge(int value)

{

age = value;

}

public int GetAge()

{

return age;

}

public User(int age, string name)

{

GetAge(age);

GetName(name);

}

}

}

Листинг Worker.cs

using System;

namespace InheritanceClasses\_17.\_02.\_20

{

class Worker: User

{

private double salary;

public void SetSalary(double value)

{

salary = value;

}

public double GetSalary()

{

return salary;

}

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

:base(age, name)

{

SetSalary(salary);

}

}

}

Листинг Student.cs

using System;

namespace InheritanceClasses\_17.\_02.\_20

{

class Student: User

{

private double scholarship;

private int course;

public void SetSholarship(double value)

{

scholarship = value;

}

public double GetScholarship()

{

return scholarship;

}

public void SetCourse(int value)

{

course = value;

}

public int GetCourse()

{

return course;

}

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

:base(age, name)

{

SetSholarship(scholarship);

SetCourse(course);

}

}

}

Листинг Driver.cs

using System;

namespace InheritanceClasses\_17.\_02.\_20

{

class Driver: Worker

{

private int stage;

private char category;

public void SetStage(int value)

{

stage = value;

}

public int GetDStage()

{

return stage;

}

public void SetCategory(char value)

{

if (value != 'A' || value != 'B' || value != 'C')

throw new Exception("Invalid value Driver category");

else

category = value;

}

public char GetCategoryDriver()

{

return category;

}

public Driver(int age, string name, double salary, int stage, char category)

:base(age, name, salary)

{

SetCategory(category);

SetStage(stage);

}

}

}

Листинг Program.cs

using System;

namespace InheritanceClasses\_17.\_02.\_20

{

class Program

{

static void Main(string[] args)

{

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

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

double sumSalaries = worker01.GetSalary() + worker02.GetSalary();

Console.WriteLine($"Сумма зарплат составляет: {sumSalaries}");

Console.ReadKey();

}

}

}