**Отношения между классами. Механизм наследования. Интерфейсы**

**Задание 1**

**Работа с наследованием**

В каждой задаче:

* **Абстрактный базовый класс**.
* **Три наследника**.
* **Массив элементов базового класса**.
* **Бизнес-логика с массивом**.

**Создание системы сотрудников**

Разработать абстрактный класс Employee с абстрактным методом CalculateSalary(). Создать три производных класса: Manager, Developer, Intern. Создать массив Employee[].

**Код**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Task1

{

internal class Program

{

static void Main(string[] args)

{

Employee[] employees = new Employee[]

{

new Manager("Катя", 5000),

new Developer("Петя", 4000),

new Intern("Вася", 15, 160)

};

Console.WriteLine("Список сотрудников и их зарплаты:");

foreach (var employee in employees)

{

Console.WriteLine($"{employee.Name}: {employee.CalculateSalary():C}");

}

}

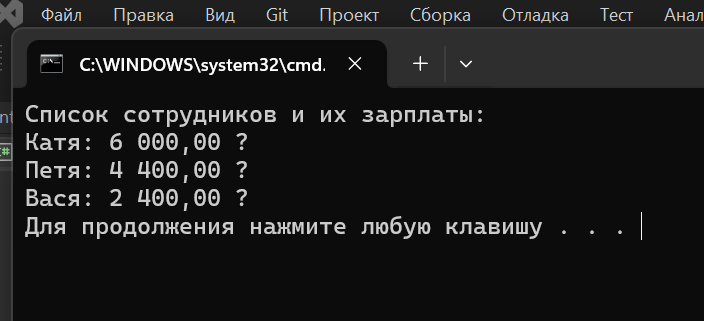
}

}

**Таблица 6.1 – Входные и выходные данные**

|  |  |
| --- | --- |
| **Входные данные** | **Выходные данные** |
| **-** | Катя, 5000, Петя, 4000, Вася, 15,160 |

**Анализ результата**

****

**Задание 2**

В каждой задаче:

* **Должны быть классы, связанные между собой отношениями ассоциации, агрегации и композиции**.
* **Бизнес-логика, связанная с массивами объектов**.

**1. Автомобиль и его компоненты**

Создать класс Car, который имеет:

* **Агрегация:** массив Wheel[] (колёса могут устанавливаться на разные автомобили).
* **Композиция:** объект Engine (двигатель создаётся внутри автомобиля).
* **Ассоциация:** связь с Driver (водитель может водить разные автомобили).  
  Создать массив автомобилей и реализовать метод Drive().

**Код**

using System;

using System.Collections.Generic;

namespace Task2

{

internal class Car

{

public Wheel[] Wheels { get; set; }

public Engine Engine { get; private set; }

public string Model { get; set; }

public Car(string model, Engine engine, Wheel[] wheels)