**ПРАКТИЧЕСКАЯ РАБОТА №10**

**Наследование классов**

**Цель работы**: научиться формировать классы-наследники, обращаться к открытым полям и методам класса-родителя, переопределять методы класса-родителя, обращаться к конструктору класса-родителя.

**Вариант 2:**

Разработать иерархию классов. Выделить родительский класс. Описать конструкторы в каждом классе. Создать перегруженные и виртуальные методы, а также методы скрывающие методы родителя. Продемонстрировать работу объектов классов в приложении.

**Перечень классов:**

1. студент, преподаватель, персона;
2. компьютер, материнская плата, винчестер.

**Ход работы:**

static void Main(string[] args)

{

///1111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111

//Person person1 = new Person("Джон Дой", 40); // Создаем объект "person1" класса "Персона"

//Student student1 = new Student("Питер Паркер", 20, "123456789", "Компьютерный специалист"); // Создаем объект "student1" класса "Студент"

//Teacher teacher1 = new Teacher("Рукабуд Альхаламулакурапидокаров", 50, "АРАОК", "Профессор"); // Создаем объект "teacher1" класса "Преподаватель"

//Console.WriteLine(person1.GetInfo()); // Выводим информацию о персоне

//Console.WriteLine(); // Переход на новую строку

//Console.WriteLine(student1.GetInfo()); // Выводим информацию о студенте

//student1.Study(); // Вызываем метод "Study" для объекта "student1"

//Console.WriteLine(); // Переход на новую строку

//Console.WriteLine(teacher1.GetInfo()); // Выводим информацию о преподавателе

//teacher1.Teach(); // Вызываем метод "Teach" для объекта "teacher1"

///22222222222222222222222222222222222222222222222222222222222222222222222222222222222222222222

Computer computer1 = new Computer("My Computer", "ASUS"); // Создаем объект "computer1" класса "Computer"

Motherboard motherboard1 = new Motherboard("ROG Strix Z690-A Gaming WiFi", "ASUS", "Intel Z690", "ATX"); // Создаем объект "motherboard1" класса "Motherboard"

HardDrive hardDrive1 = new HardDrive("WD Blue WD10EZEX", "Western Digital", "1TB", "SATA III"); // Создаем объект "hardDrive1" класса "HardDrive"

Console.WriteLine(computer1.GetInfo()); // Выводим информацию о компьютере

Console.WriteLine(); // Переход на новую строку

Console.WriteLine(motherboard1.GetInfo()); // Выводим информацию о материнской плате

Console.WriteLine(); // Переход на новую строку

Console.WriteLine(hardDrive1.GetInfo()); // Выводим информацию о винчестере

Console.ReadKey();

}

public class Student : Person // Определяем класс "Студент", наследуемый от "Персона"

{

private string StudentNumber; // Поле "Номер студенческого" (приватное)

private string Major; // Поле "Специальность" (приватное)

public Student(string Name, int Age, string StudentNumber, string Major) // Конструктор класса "Студент"

: base(Name, Age) // Инициализация базового класса

{

this.StudentNumber = StudentNumber; // Инициализация поля "Номер студенческого"

this.Major = Major; // Инициализация поля "Специальность"

}

public override string GetInfo() => // Переопределение метода "GetInfo"

base.GetInfo() + $"\nНомер студента: {StudentNumber}\nСпециальность: {Major}"; // Возвращает информацию о студенте

public void Study() => Console.WriteLine($"{Name} обучается!"); // Метод "Study" (выводит сообщение, что студент учится)

}

public class Teacher : Person // Определяем класс "Преподаватель", наследуемый от "Персона"

{

private string Department; // Поле "Кафедра" (приватное)

private string Position; // Поле "Должность" (приватное)

public Teacher(string Name, int Age, string Department, string Position) // Конструктор класса "Преподаватель"

: base(Name, Age) // Инициализация базового класса

{

this.Department = Department; // Инициализация поля "Кафедра"

this.Position = Position; // Инициализация поля "Должность"

}

public override string GetInfo() => // Переопределение метода "GetInfo"

base.GetInfo() + $"\nКафедра: {Department}\nДолжность: {Position}"; // Возвращает информацию о преподавателе

public void Teach() => Console.WriteLine($"{Name} преподает!"); // Метод "Teach" (выводит сообщение, что преподаватель преподает)

}

public class Person // Определяем класс "Персона"

{

protected string Name; // Поле "Имя" (защищенное)

protected int Age; // Поле "Возраст" (защищенное)

public Person(string Name, int Age) // Конструктор класса "Персона"

{

this.Name = Name; // Инициализация поля "Имя"

this.Age = Age; // Инициализация поля "Возраст"

}

public virtual string GetInfo() => $"Имя: {Name}, Возраст: {Age}"; // Виртуальный метод "GetInfo" (возвращает информацию о персоне)

}

public class Motherboard : Computer // Определяем класс "МатеринскаяПлата", наследуемый от "Компьютер"

{

private string Chipset; // Поле "Чипсет" (приватное)

private string FormFactor; // Поле "Форм-фактор" (приватное)

public Motherboard(string Name, string Manufacturer, string Chipset, string FormFactor) // Конструктор класса "МатеринскаяПлата"

: base(Name, Manufacturer) // Инициализация базового класса

{

this.Chipset = Chipset; // Инициализация поля "Чипсет"

this.FormFactor = FormFactor; // Инициализация поля "Форм-фактор"

}

public override string GetInfo() => // Переопределение метода "GetInfo"

base.GetInfo() + $"\nChipset: {Chipset}\nForm Factor: {FormFactor}"; // Возвращает информацию о материнской плате

}

public class HardDrive : Computer // Определяем класс "Винчестер", наследуемый от "Компьютер"

{

private string Capacity; // Поле "Объем" (приватное)

private string Interface; // Поле "Интерфейс" (приватное)

public HardDrive(string Name, string Manufacturer, string Capacity, string Interface) // Конструктор класса "Винчестер"

: base(Name, Manufacturer) // Инициализация базового класса

{

this.Capacity = Capacity; // Инициализация поля "Объем"

this.Interface = Interface; // Инициализация поля "Интерфейс"

}

public override string GetInfo() => // Переопределение метода "GetInfo"

base.GetInfo() + $"\nCapacity: {Capacity}\nInterface: {Interface}"; // Возвращает информацию о винчестере

}

public class Computer // Определяем класс "Компьютер"

{

protected string Name; // Поле "Название" (защищенное)

protected string Manufacturer; // Поле "Производитель" (защищенное)

public Computer(string Name, string Manufacturer) // Конструктор класса "Компьютер"

{

this.Name = Name; // Инициализация поля "Название"

this.Manufacturer = Manufacturer; // Инициализация поля "Производитель"

}

public virtual string GetInfo() => $"Name: {Name}, Manufacturer: {Manufacturer}"; // Виртуальный метод "GetInfo" (возвращает информацию о компьютере)

}

**Вывод**: Научился формировать классы-наследники, обращаться к открытым полям и методам класса-родителя, переопределять методы класса-родителя, обращаться к конструктору класса-родителя.