МИНИCTEPCTBO НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии

Департамент цифровых, робототехнических систем и электроники

Отчет к практической работе № 9

Тема: **«Проектирование иерархии классов»**

Дисциплина: **«Алгоритмы и структуры данных»**

|  |  |  |
| --- | --- | --- |
|  |  | **Выполнил:** |
|  |  | Студент группы ПИЖ-б-о-23-1, направление подготовки: 09.03.04 «Программная инженерия»  Панчешный Александр Алексеевич  **Проверил:**  Доцент ДЦРСИЭ  Николаев Евгений Иванович |

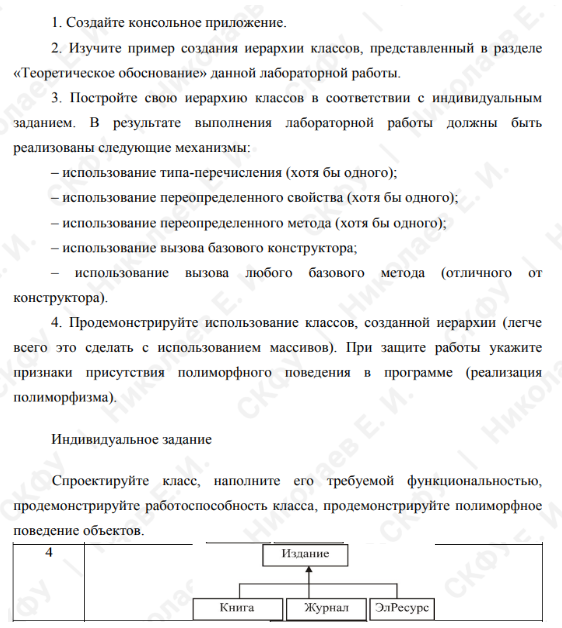
Ставрополь 2025

**Лабораторная работа 9.** **Проектирование иерархии классов**

**Цель работы:** изучить механизм организации наследования классов.

**Ход работы**

**I. Задание и исходный код индивидуального задания**

****

using System;

namespace PublicationHierarchy

{

// Тип-перечисление для типов изданий

public enum PublicationType

{

Book,

Magazine,

ElectronicResource

}

// Базовый класс "Издание"

public class Publication

{

public string Title { get; set; }

public int Year { get; set; }

public PublicationType Type { get; set; }

public Publication(string title, int year, PublicationType type)

{

Title = title;

Year = year;

Type = type;

}

public virtual void DisplayInfo()

{

Console.WriteLine($"Название: {Title}, Год: {Year}, Тип: {Type}");

}

public virtual string GetDescription()

{

return $"Издание: {Title} ({Year})";

}

}

// Класс "Книга", наследующий от "Издание"

public class Book : Publication

{

public string Author { get; set; }

public Book(string title, int year, string author) : base(title, year, PublicationType.Book)

{

Author = author;

}

public override void DisplayInfo()

{

base.DisplayInfo();

Console.WriteLine($"Автор: {Author}");

}

public override string GetDescription()

{

return $"Книга: {Title} ({Year}), Автор: {Author}";

}

}

// Класс "Журнал", наследующий от "Издание"

public class Magazine : Publication

{

public int IssueNumber { get; set; }

public Magazine(string title, int year, int issueNumber) : base(title, year, PublicationType.Magazine)

{

IssueNumber = issueNumber;

}

public override void DisplayInfo()

{

base.DisplayInfo();

Console.WriteLine($"Номер выпуска: {IssueNumber}");

}

public override string GetDescription()

{

return $"Журнал: {Title} ({Year}), Выпуск: {IssueNumber}";

}

}

// Класс "Электронный ресурс", наследующий от "Издание"

public class ElectronicResource : Publication

{

public string Url { get; set; }

public ElectronicResource(string title, int year, string url) : base(title, year, PublicationType.ElectronicResource)

{

Url = url;

}

public override void DisplayInfo()

{

base.DisplayInfo();

Console.WriteLine($"URL: {Url}");

}

public override string GetDescription()

{

return $"Электронный ресурс: {Title} ({Year}), URL: {Url}";

}

}

class Program

{

static void Main(string[] args)

{

// Массив изданий

Publication[] publications = new Publication[]

{

new Book("Мастер и Маргарита", 1967, "Булгаков М.А."),

new Magazine("Наука и жизнь", 2023, 5),

new ElectronicResource("Википедия", 2023, "https://ru.wikipedia.org")

};

// Демонстрация полиморфизма

foreach (var publication in publications)

{

publication.DisplayInfo();

Console.WriteLine(publication.GetDescription());

Console.WriteLine();

}

Console.ReadKey();

}

}

}

**II. Контрольные вопросы**

**1. Что такое наследование реализации? Как описать синтаксически наследование реализации?**

Наследование реализации - это механизм объектно-ориентированного программирования, позволяющий одному классу (производному) наследовать поля и методы другого класса (базового). Производный класс может использовать, изменять или расширять функциональность базового класса.

Синтаксически наследование реализации описывается с помощью двоеточия (:) после имени производного класса и указания имени базового класса.

**Пример:**

public class BaseClass

{

public int BaseField;

public void BaseMethod() { }

}

public class DerivedClass : BaseClass

{

public int DerivedField;

public void DerivedMethod() { }

}

**2. Для чего используется ключевое слово base?**

Ключевое слово base используется для доступа к членам базового класса из производного класса. Оно позволяет:

* Вызывать конструктор базового класса из конструктора производного класса.
* Вызывать методы базового класса, переопределенные в производном классе.
* Получать доступ к полям базового класса, скрытым в производном классе.

**3. Можно ли переопределить метод класса? Свойства класса? Данные класса?**

* **Метод класса:** Можно переопределить метод класса, если он объявлен как virtual в базовом классе.
* **Свойства класса:** Можно переопределить свойства класса, если они объявлены как virtual в базовом классе.
* **Данные класса (поля):** Нельзя переопределить данные класса. Вместо этого можно скрыть поля базового класса, объявив поля с тем же именем в производном классе.

**4. Как переопределить метод в производном классе?**

Для переопределения метода в производном классе необходимо:

1. Объявить метод в базовом классе как virtual.
2. Объявить метод с той же сигнатурой в производном классе с модификатором override.

**Пример:**

public class BaseClass

{

public virtual void MyMethod() { }

}

public class DerivedClass : BaseClass

{

public override void MyMethod() { }

}

**5. Для чего используется ключевое слово virtual?**

Ключевое слово virtual используется для объявления метода или свойства в базовом классе, который может быть переопределен в производном классе.

**6. Для чего используется ключевое слово override?**

Ключевое слово override используется для объявления метода или свойства в производном классе, который переопределяет виртуальный метод или свойство базового класса.

**7. Как поменять цвет фона в консольном приложении?**

Для изменения цвета фона в консольном приложении используется свойство Console.BackgroundColor.

**Пример:**

Console.BackgroundColor = ConsoleColor.Blue;

Console.Clear(); // Очищает консоль и применяет новый цвет фона

**8. Укажите ошибки (если есть) в представленном фрагменте:**

public class Parent

{

int a;

}

class Child : Parent

{

int b;

void E()

{

b = a;

}

}

**Ответ:**

* **Отсутствует модификатор доступа для поля a:** По умолчанию поля класса имеют модификатор доступа private, поэтому поле a недоступно в производном классе Child.
* **Отсутствует модификатор доступа для метода E():** По умолчанию методы класса имеют модификатор доступа private, поэтому метод E() доступен только внутри класса Child.

**9. Сколько базовых классов может быть у любого класса в C#?**

В C# класс может иметь только один базовый класс (одиночное наследование).

**10. Даны определения классов на языке C#:**

public class Parent

{

protected int a;

Parent()

{

a = 0;

}

}

class Child1 : Parent

{

int c;

public Child1() : base()

{

}

}

**Вопрос:** Существуют ли ошибки в данном коде (укажите ошибки, если есть)?

**Ответ:**

* **Отсутствует модификатор доступа для поля c:** По умолчанию поля класса имеют модификатор доступа private, поэтому поле c доступно только внутри класса Child1.
* **Конструктор Parent() объявлен как internal:** По умолчанию конструкторы класса имеют модификатор доступа internal, поэтому он доступен только внутри текущей сборки.