**Лабораторная работа № 6. Наследование**

**Задание 1.** Составить программу с одним родительским классом и потомком. Все поля должны быть закрытыми. Базовый класс должен содержать конструкторы с параметрами, методы доступа к закрытым полям, вывод полей и указанный в таблице метод. Производный класс содержит дополнения и изменения, организовать вывод новых полей потомка, при этом имена методов совпадают с именами методов базового класса. Составить тестирующую программу с выдачей результатов. Создать объекты базового и производного типов. В программе должна выполняться проверка всех разработанных элементов класса.

9. Базовый класс: Студент (поле: имя, средний балл s)

Метод: Стипендия 300000+10000([s]-5)

Потомок: Магистр (поле – специальность)

Изменения в потомках: Увеличить стипендию на m руб.

using System;

// Базовый класс (родительский класс)

class Student

{

// Закрытые поля

private string name;

private double averageGrade;

// Конструктор с параметрами

public Student(string name, double averageGrade)

{

this.name = name;

this.averageGrade = averageGrade;

}

// Метод доступа к закрытым полям

public string GetName()

{

return name;

}

public double GetAverageGrade()

{

return averageGrade;

}

// Метод для расчета стипендии

public double CalculateScholarship()

{

return 300000 + 10000 \* (averageGrade - 5);

}

}

// Производный класс (потомок)

class Master : Student

{

// Дополнительное поле для специальности магистра

private string specialty;

// Конструктор с параметрами

public Master(string name, double averageGrade, string specialty)

: base(name, averageGrade)

{

this.specialty = specialty;

}

// Метод для увеличения стипендии на m рублей

public double IncreaseScholarship(double m)

{

return CalculateScholarship() + m;

}

// Метод для вывода информации о магистре

public void DisplayMasterInfo()

{

Console.WriteLine($"Магистр {GetName()}, специальность: {specialty}");

Console.WriteLine($"Стипендия: {CalculateScholarship()} рублей");

}

}

class Program

{

static void Main()

{

// Создаем объект базового класса (Студент)

Student student = new Student("Иван", 4.8);

// Создаем объект производного класса (Магистр)

Master master = new Master("Анна", 5.2, "Информационные технологии");

// Выводим информацию о студенте и магистре

Console.WriteLine("Информация о студенте:");

Console.WriteLine($"Имя: {student.GetName()}, Средний балл: {student.GetAverageGrade()}");

Console.WriteLine($"Стипендия: {student.CalculateScholarship()} рублей");

Console.WriteLine("\nИнформация о магистре:");

master.DisplayMasterInfo();

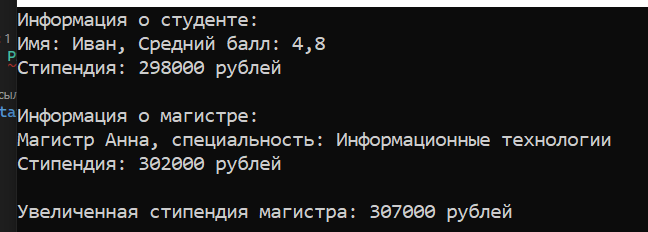
// Увеличиваем стипендию магистра на 5000 рублей

double increasedScholarship = master.IncreaseScholarship(5000);

Console.WriteLine($"\nУвеличенная стипендия магистра: {increasedScholarship} рублей");

}

}



**Контрольные вопросы:**

1. **В чем состоит принцип наследования?**

Принцип наследования заключается в том, что класс может наследовать свойства и методы от другого класса, который называется базовым классом или родительским классом. Это позволяет создавать иерархию классов, где дочерние классы наследуют общие характеристики от родительских.

1. **Какие члены класса наследуются?**

Члены класса, которые наследуются, включают в себя:

* Поля (переменные): Дочерний класс может наследовать поля из родительского класса.
* Методы: Дочерний класс может наследовать методы, определенные в родительском классе.
* Свойства: Свойства (getters и setters) также наследуются.

1. **Что представляет собой защищенный доступ?**

Защищенный доступ (protected access) представляет собой уровень доступа, при котором члены класса видны только внутри класса и его дочерних классов. Он обозначается символом \_ перед именем члена (например, \_protectedField).

1. **Как происходит вызов конструкторов базового класса?**

Вызов конструкторов базового класса происходит с помощью ключевого слова super. В дочернем классе мы можем вызвать конструктор родительского класса, чтобы выполнить инициализацию общих полей.

1. **Что такое сокрытие имен при наследовании?**

Сокрытие имен при наследовании (name hiding) возникает, когда дочерний класс определяет член с тем же именем, что и в родительском классе. В этом случае член родительского класса становится недоступным в дочернем классе.

1. **Как получить доступ к сокрытому члену базового класса?**

Для доступа к сокрытым членам базового класса можно использовать ключевое слово base. Например, base.MethodName() вызовет метод из родительского класса.