Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра Информатики

Дисциплина «Программирование»

**ОТЧЕТ**

к лабораторной работе №6

на тему:

**«НАСЛЕДОВАНИЕ»**

БГУИР 6-05-0612-02 36

|  |
| --- |
| Выполнила студентка группы 353505  ДАНИЛОВ Дмитрий Игоревич |
|  |
| (дата, подпись студента) |
| Проверил ассистент каф. Информатики  РОМАНЮК Максим Валерьевич |
|  |
| (дата, подпись преподавателя) |

Минск 2024

# 1 Индивидуальное задание

Задание 1. Вариант 8.Выделить в предметной области 2-3 варианта сущности (например, "Геометрическая фигура", "Точка", "Треугольник"), отличающиеся несколькими полями и методами. Каждый класс имеет поля, свойства и методы.

1. Спроектировать UML-диаграммы классов.
2. Базовый класс для вашей иерархии объявите абстрактным. Он должен содержать абстрактные методы и методы с реализацией.
3. Один из наследников должен перегружать метод родителя.
4. Один из классов должен содержать виртуальный метод, который

переопределяется в одном наследнике и не переопределяется в другом.

5) Продемонстрировать работу всех объявленных методов.

6) Продемонстрировать вызов конструктора родительского класса при наследовании.

7) Продемонстрировать вызов метода родительского класса при его

скрытии.

8) Создать класс, закрытый для наследования.

Индивидуальное задание – игрушки.

# 2 Выполнение работы

Для выполнения данной лабораторной работы первым делом был создан проект, к которому позже добавлялись классы, реализующие данное задание. Сначала был реализован абстрактный класс «Toy», в котором были реализованы методы «Play» и «ShowInfo». От класса «Toy» наследуется два класса: «Ball» и «Doll». В обоих этих классах реализованы методы «ShowInfo» и «Play». Результат работы программы, учитывая все требования к лабораторной работе представлен на рисунке (рисунок 1). После проектирования всех классов было создана UML-диаграмма, отображающая отношения между классами (рисунок 2).

abstract class Toy

{

protected string name;

protected decimal price;

public Toy(string name, decimal price)

{

this.name = name;

this.price = price;

}

public abstract string Play();

public virtual string ShowInfo()

{

return "Название игрушки: " + name + "\nЦена: $" + price;

}

}

class Ball : Toy

{

private double diameter;

private string color;

public Ball(string name, decimal price, double diameter, string color) : base(name, price)

{

this.diameter = diameter;

this.color = color;

}

public override string Play()

{

return "Игра с мячом";

}

public override string ShowInfo()

{

return base.ShowInfo() + "\nДиаметр: " + diameter + " см" + "\nЦвет: " + color;

}

}

class Doll : Toy

{

private string material;

private decimal size;

public Doll(string name, decimal price, string material, decimal size) : base(name, price)

{

this.material = material;

this.size = size;

}

public override string Play()

{

return "Игра с куклой";

}

public override string ShowInfo()

{

return base.ShowInfo() + "\nМатериал: " + material + "\nРазмер: " + size;

}

}

using System;

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Введите данные для мяча:");

Console.Write("Название: ");

string ballName = Console.ReadLine();

Console.Write("Цена: ");

decimal ballPrice;

while (!decimal.TryParse(Console.ReadLine(), out ballPrice) || ballPrice <= 0)

{

Console.WriteLine("Некорректный ввод. Введите положительное число для цены снова: ");

}

Console.Write("Диаметр: ");

double ballDiameter;

while (!double.TryParse(Console.ReadLine(), out ballDiameter) || ballDiameter <= 0)

{

Console.WriteLine("Некорректный ввод. Введите положительное число для диаметра снова: ");

}

Console.Write("Цвет: ");

string ballColor = Console.ReadLine();

Ball ball = new Ball(ballName, ballPrice, ballDiameter, ballColor);

Console.WriteLine();

Console.WriteLine("Введите данные для куклы:");

Console.Write("Название: ");

string dollName = Console.ReadLine();

Console.Write("Цена: ");

decimal dollPrice;

while (!decimal.TryParse(Console.ReadLine(), out dollPrice) || dollPrice <= 0)

{

Console.WriteLine("Некорректный ввод. Введите положительное число для цены снова: ");

}

Console.Write("Материал: ");

string dollMaterial = Console.ReadLine();

Console.Write("Размер: ");

decimal dollSize;

while (!decimal.TryParse(Console.ReadLine(), out dollSize) || dollSize <= 0)

{

Console.WriteLine("Некорректный ввод. Введите положительное число для размера снова: ");

}

Doll doll = new Doll(dollName, dollPrice, dollMaterial, dollSize);

Console.WriteLine();

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

Console.WriteLine(ball.ShowInfo());

Console.WriteLine(ball.Play());

Console.WriteLine();

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

Console.WriteLine(doll.ShowInfo());

Console.WriteLine(doll.Play());

Console.ReadLine();

}

}

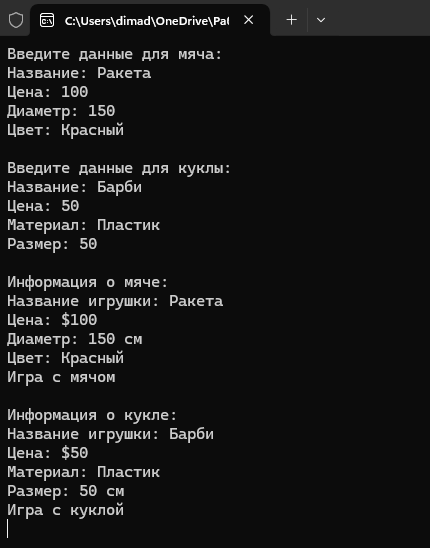


Рисунок 1 – Результат работы программы

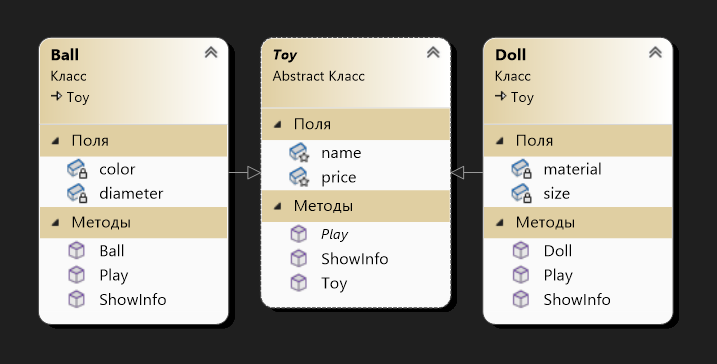


Рисунок 2 – UML-диаграмма

# Вывод

В ходе лабораторной работы мною было изучено наследование в языке C#. Мною были углублены знания по работе с классами в объектно-ориентированным языком программирования C#. Были рассмотрены обращения к классу-родителю из класса-наследник для вызова методов класса-родителя. Была изучена схема создания UML-диаграмм, которые позволяют отображать отношения между классами, что позволяет более простым способом их анализировать. Было установлено, что при создании объекта класса-наследника вызывается конструктор класса-родителя.