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 6-05-0612-02 124

|  |
| --- |
| Выполнил студент группы 453504  ЯРЦЕВ Александр Александрович |
|  |
| (дата, подпись студента) |
| Проверил ассистент каф. Информатики  РОМАНЮК Максим Валерьевич |
|  |
| (дата, подпись преподавателя) |

Минск 2025

# 1 Индивидуальное задание

**Задание 1. Вариант 9.** Предметная область: Товары.

Для заданной предметной области реализовать следующие задачи:

– выделить в предметной области 2-3 варианта сущности, отличающиеся несколькими полями и методами. Каждый класс имеет поля, свойства и методы;

– cпроектировать UML-диаграммы классов;

– базовый класс для вашей иерархии объявите абстрактным. Он

должен содержать абстрактные методы и методы с реализацией;

– один из наследников должен перегружать метод родителя;

– один из классов должен содержать виртуальный метод, который

переопределяется в одном наследнике и не переопределяется в другом;

– продемонстрировать работу всех объявленных методов;

– продемонстрировать вызов конструктора родительского

класса при наследовании;

– продемонстрировать вызов метода родительского класса при его

скрытии;

– создать класс, закрытый для наследования;

# 2 Выполнение работы

Перед выполнением работы следует разработать диаграмму классов для наглядного выполнения поставленной задачи (см. рисунок 1).

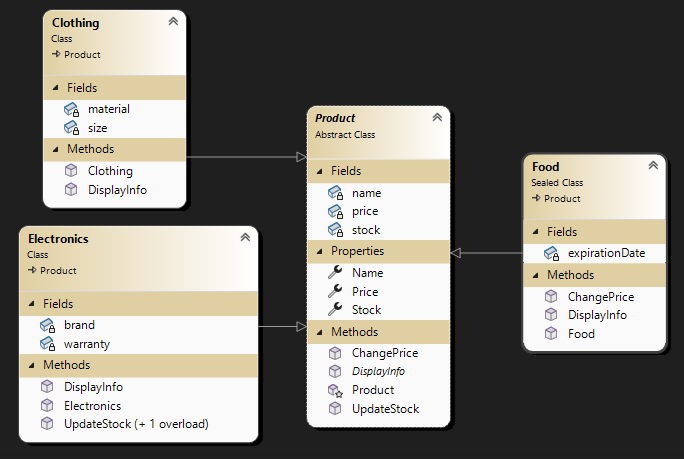


Рисунок 1 – Диаграмма классов

Для выполнения задания в папку Solution Items были добавлены классы Clothing, Electronics, Food, Product (cм. рисунок 2).

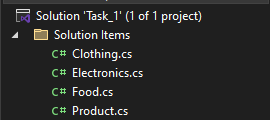


Рисунок 2 – Обозреватель решений

Рассмотрим реализацию абстрактного класса Product со свойствами Name, Price и Stock, так как они с модификатором доступа private protected, то эти поля, помимо базового класса, будут доступны в производных классах текущей сборки. Помимо этого, в классе реализован абстрактный метод DisplayInfo, который гарантирует, что реализация этого метода будет в производных классах. И один virtual метод UpdateStock, который можно будет переопределить.

namespace Shop

{

public abstract class Product

{

private string name;

private decimal price;

private int stock;

protected Product(string name, decimal price, int stock)

{

this.name = name;

this.price = price;

this.stock = stock;

}

public string Name { get => name; set => name = value; }

public decimal Price { get => price; set => price = value; }

public int Stock { get => stock; set => stock = value; }

public abstract void DisplayInfo(); // Абстрактный метод

public virtual void UpdateStock(int quantity) // Виртуальный метод

{

Stock += quantity;

Console.WriteLine($"{Name} stock updated to {Stock}");

}

public void ChangePrice(decimal newPrice) // Метод с реализацией

{

Price = newPrice;

Console.WriteLine($"{Name} price changed to {Price}");

}

}

}

Сперва взглянем на реализацию класса наследника Electronics, где помимо свойств, которые доступны из базового класса, имеются свойства Brand и Warranty. Конструктор класса Electronics в своей реализации вызывает конструктор базового класса, это позволяет значительно экономить код. В классе также переопределены методы DisplayInfo и UpdateStock.

namespace Shop

{

public class Electronics : Product

{

private string brand;

private int warranty;

public Electronics(string name, double price, int stock, string brand, int warranty) : base(name, price, stock)

{

this.brand = brand;

this.warranty = warranty;

}

public string Brand

{

get => brand;

set => brand = value;

}

public int Warranty

{

get => warranty;

set => warranty = value;

}

public override void DisplayInfo()

{

Console.WriteLine($"Electronics: {Name}, Brand: {Brand}, Price: {Price}, Stock: {Stock}, Warranty: {Warranty} months");

}

public override void UpdateStock(int quantity)

{

if (quantity < 0 && Stock + quantity < 0)

{

Console.WriteLine($"Cannot reduce stock for {Name}. Insufficient stock.");

return;

}

base.UpdateStock(quantity);

}

}

}

Рассмотрим реализацию ещё одного наследника Clothing абстрактного класса Product, где метод UpdateStock не переопределяется, но переопределяется метод DisplayInfo, добавлены свойства Size и Material.

namespace Shop

{

public class Clothing : Product

{

private string size;

private string material;

public Clothing(string name, double price, int stock, string size, string material) : base(name, price, stock)

{

this.size = size;

this.material = material;

}

public string Size

{

get => size;

set => size = value;

}

public string Material

{

get => material;

set => material = value;

}

public override void DisplayInfo()

{

Console.WriteLine($"Clothing: {Name}, Size: {Size}, Material: {Material}, Price: {Price}, Stock: {Stock}");

}

// UpdateStock не переопределяется, используется реализация базового класса

}

}

Теперь рассмотрим реализацию класса Food, который имеет ключевое слово sealed, то есть класс Food закрыт для наследования.

namespace Shop

{

public sealed class Food

{

private string name;

private decimal price;

private DateTime expirationDate;

public Food(string name, decimal price, DateTime expirationDate)

{

this.name = name;

this.price = price;

this.expirationDate = expirationDate;

}

public string Name

{

get => name;

set => name = value;

}

public decimal Price

{

get => price;

set => price = value;

}

public DateTime ExpirationDate

{

get => expirationDate;

set => expirationDate = value;

}

public void DisplayInfo()

{

Console.WriteLine($"Food: {Name}, Price: {Price}, Expires: {ExpirationDate:dd-MM-yyyy}");

}

}

}

Покажем реализацию класса Program, где мы показываем реализацию нашей программы.

namespace Shop

{

public abstract class Product

{

private string name;

private decimal price;

private int stock;

protected Product(string name, decimal price, int stock)

{

this.name = name;

this.price = price;

this.stock = stock;

}

public string Name { get => name; set => name = value; }

public decimal Price { get => price; set => price = value; }

public int Stock { get => stock; set => stock = value; }

public abstract void DisplayInfo(); // Абстрактный метод

public virtual void UpdateStock(int quantity) // Виртуальный метод

{

Stock += quantity;

Console.WriteLine($"{Name} stock updated to {Stock}");

}

public void ChangePrice(decimal newPrice) // Метод с реализацией

{

Price = newPrice;

Console.WriteLine($"{Name} price changed to {Price}");

}

}

}

Результат работы программы продемонстрирован ниже (см. рисунок 3).

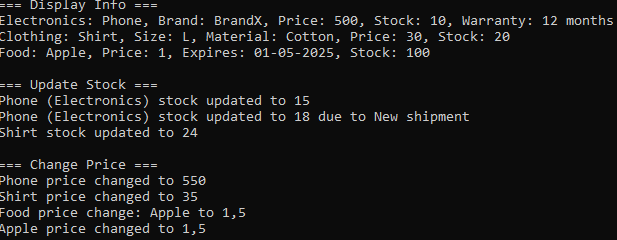


Рисунок 3 – Результат работы программы

# Вывод

В ходе лабораторной работы были изучены принципы построения диаграмм классов при наследовании классов. Изучены такие понятия абстрактные классы, виртуальные методы, классы, закрытые для наследования, а также изменение работы метода родительского класса при его скрытии.