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 67

|  |
| --- |
| Выполнил студент группы 353503  КОХАН Артём Игоревич |
|  |
| (дата, подпись студента) |
| Проверил ассистент каф. Информатики  РОМАНЮК Максим Валерьевич |
|  |
| (дата, подпись преподавателя) |

Минск 2024

# 1 Индивидуальное задание

**Задание 1. Вариант 7.** Предметная область: Компьютерная техника.

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).

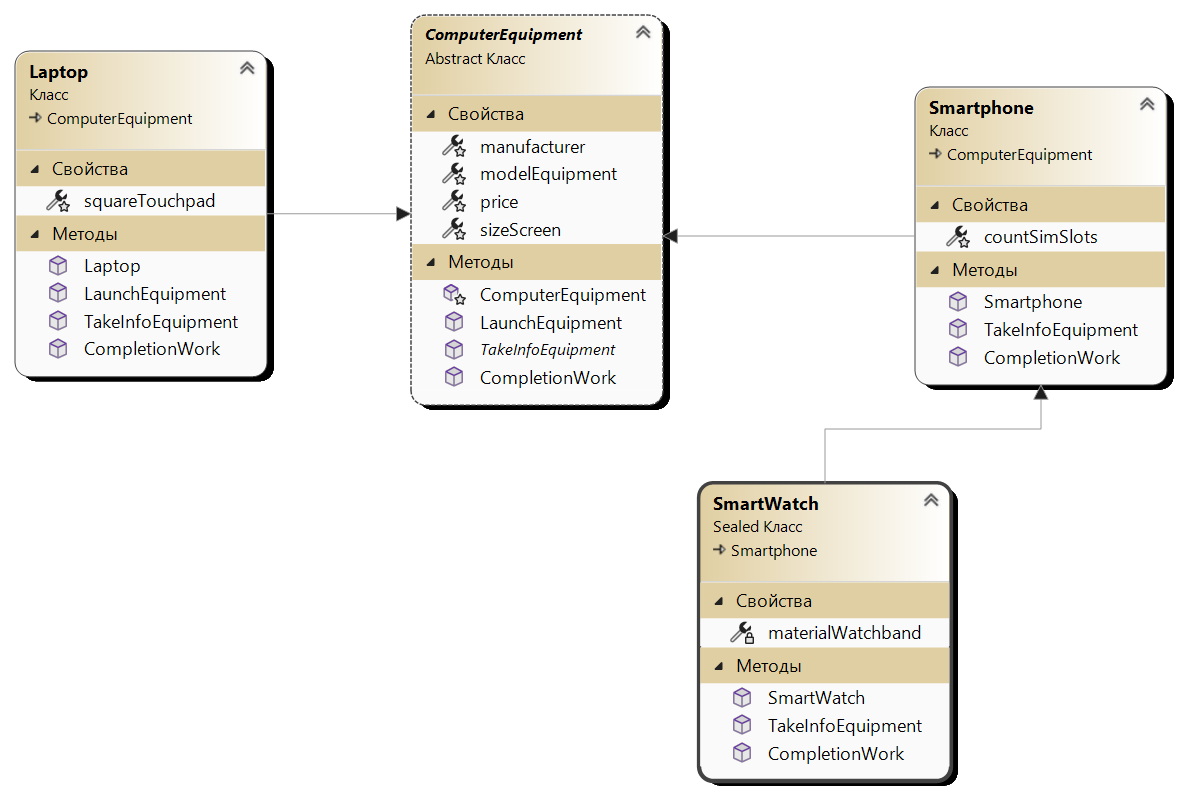


Рисунок 1 – Диаграмма классов

Для выполнения задания в проект была добавлена папка Classes, в которой находятся классы ComputerEquipment, Laptop, Smartphone, Smartwatch (cм. рисунок 2).

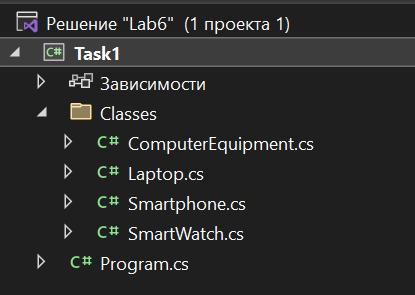


Рисунок 2 – Обозреватель решений

Рассмотрим реализацию абстрактного класса ComputerEquipment со свойствами manufacturer, modelEquipment, sizeScreen, price, 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 TakeInfoEquipment, который гарантирует, что реализация этого метода будет в производных классах. И два virtual метода LaunchEquipment, CompletionWork, которые можно будет переопределить в производных классах.

namespace Task1.Classes

{

internal abstract class ComputerEquipment

{

private protected string manufacturer { get; set; }

private protected string modelEquipment { get; set;}

private protected double sizeScreen { get; set; }

private protected double price { get; set; }

private protected ComputerEquipment(string manufacturer, string modelEquipment,

double sizeScreen, double price )

{

Console.WriteLine("Вызван конструктор базового класса\n");

this.manufacturer = manufacturer;

this.modelEquipment = modelEquipment;

this.sizeScreen = sizeScreen;

this.price = price;

}

public abstract void TakeInfoEquipment();

public virtual void LaunchEquipment() =>

Console.WriteLine("Launch device\n");

public virtual void СompletionWork() =>

Console.WriteLine("Completion of work(parent method)\n");

}

}

Сперва взглянем на реализацию класса наследника Laptop, где помимо свойств, которые доступны из базового класса, имеется свойство squereTouchpad. Конструктор класса Laptop в своей реализации вызывает конструктор базового класса, это позволяет значительно экономить код. В классе также реализован метод TakeInfoEquipment и переопределены методы

LaunchEquipment, СompletionWork.

namespace Task1.Classes

{

internal class Laptop : ComputerEquipment

{

private protected double squareTouchpad { get; set; }

public Laptop(string manufacturer, string modelEquipment,

double sizeScreen, double price, double squareTouchpad) : base(manufacturer, modelEquipment,

sizeScreen, price)

{

Console.WriteLine("Вызван конструктор класса Laptop\n");

this.squareTouchpad = squareTouchpad;

}

public override void TakeInfoEquipment()

{

Console.WriteLine($"1. Производитель: {manufacturer}\n" +

$"2. Модель: {modelEquipment}\n" +

$"3. Размеры экрана: {sizeScreen}\n" +

$"4. Цена: {price}$\n" +

$"5. Размеры тачпада: {squareTouchpad}\n");

}

public override void LaunchEquipment() =>

Console.WriteLine("Launch laptop\n");

public new void СompletionWork() =>

Console.WriteLine("Close the laptop lid\n");

}

}

Рассмотрим реализацию ещё одного наследника Smartphone абстрактного класса ComputerEquipment, где реализовано скрытие метода СompletionWork базового класса с помощью ключевого слово new.

namespace Task1.Classes

{

internal class Smartphone : ComputerEquipment

{

private protected double countSimSlots { get; set; }

public Smartphone(string manufacturer, string modelEquipment,

double sizeScreen, double price, double countSimSlots) : base(manufacturer, modelEquipment,

sizeScreen, price)

{

Console.WriteLine("Вызван конструктор класса Smartphone\n");

this.countSimSlots = countSimSlots;

}

public override void TakeInfoEquipment()

{

Console.WriteLine($"1. Производитель: {manufacturer}\n" +

$"2. Модель: {modelEquipment}\n" +

$"3. Размеры экрана: {sizeScreen}\n" +

$"4. Цена: {price}$\n" +

$"5. Количество слотов для сим карт: {countSimSlots}\n");

}

public new void СompletionWork()

{

Console.WriteLine("The main button was pressed(Smartphone method)");

base.СompletionWork();

}

}

}

Рассмотрим класс SmartWatch, где продемонстрировано в чём произошло изменение после использования ключевого слова new для скрытия метода базового класса. Стоит обратить внимание, что класс у нас Sealed, что позволяет предотвратить создание наследников этого класса.

namespace Task1.Classes

{

internal sealed class SmartWatch : Smartphone

{

private string materialWatchband { get; set; }

public SmartWatch(string manufacturer, string modelEquipment,

double sizeScreen, double price, double countSimSlots, string materialWatchband) :

base(manufacturer, modelEquipment,

sizeScreen, price, countSimSlots)

{

Console.WriteLine("Вызван конструктор класса SmartWatch\n");

this.materialWatchband = materialWatchband;

}

public override void TakeInfoEquipment()

{

Console.WriteLine($"1. Производитель: {manufacturer}\n" +

$"2. Модель: {modelEquipment}\n" +

$"3. Размеры экрана: {sizeScreen}\n" +

$"4. Цена: {price}$\n" +

$"5. Количество слотов для сим карт: {countSimSlots}\n" +

$"6. Материал ремешка часов: {materialWatchband}\n");

}

public new void СompletionWork() =>

base.СompletionWork();

}

}

Результат работы классов описан в классе Program, чтобы наглядно разобраться в структуре созданных классов.

namespace Task1

{

file class Program

{

static void Main(string[] args)

{

Laptop firsrMacLaptop = new Laptop("Apple", "PowerBook", 10, 6500, 5);

firsrMacLaptop.TakeInfoEquipment();

firsrMacLaptop.LaunchEquipment();

Console.WriteLine("СompletionWork for Laptop:");

firsrMacLaptop.СompletionWork();

Laptop victus = new Laptop("HP", "Victus 16", 16, 2735.535, 12.5);

victus.TakeInfoEquipment();

victus.LaunchEquipment();

Console.WriteLine("СompletionWork for Laptop:");

victus.СompletionWork();

Smartphone mySmartphone = new Smartphone("Honor", "20", 6.1, 369, 2);

mySmartphone.TakeInfoEquipment();

mySmartphone.LaunchEquipment();

Console.WriteLine("СompletionWork for SmartPhone:");

mySmartphone.СompletionWork();

Smartphone badSmartphone = new Smartphone("Xiaomi", "14", 6.36, 999, 2);

badSmartphone.TakeInfoEquipment();

badSmartphone.LaunchEquipment();

Console.WriteLine("СompletionWork for SmartPhone:");

badSmartphone.СompletionWork();

SmartWatch myWatch = new SmartWatch("Huawei", "Watch fit 2", 1.88, 100, 0, "Artificial leather");

myWatch.TakeInfoEquipment();

myWatch.LaunchEquipment();

Console.WriteLine("СompletionWork for SmartWatch:");

myWatch.СompletionWork();

}

}

}

Результат работы программы продемонстрирован ниже (см. рисунок 3 и 4).

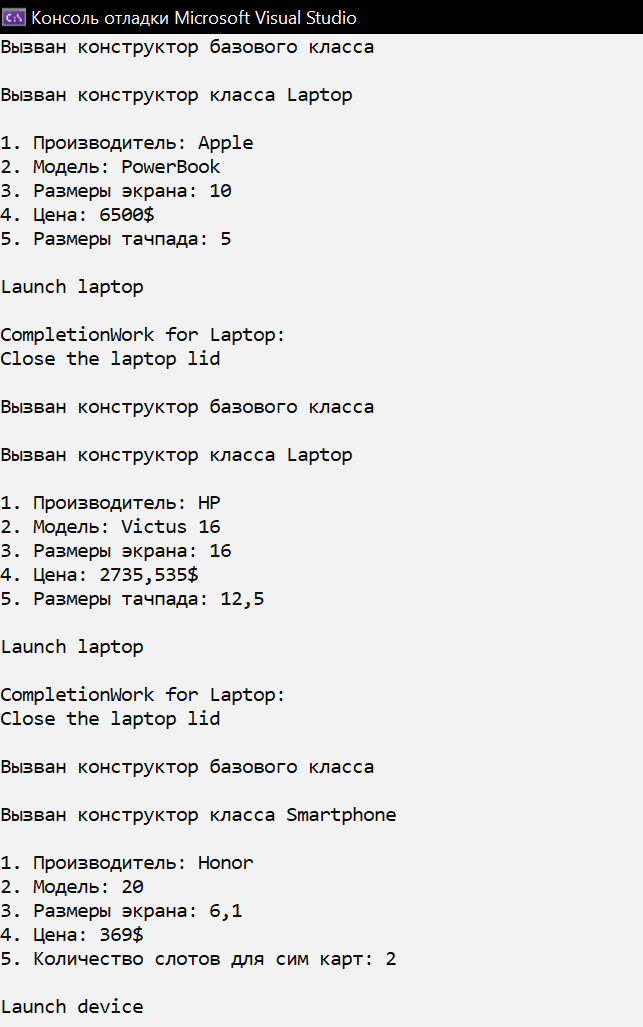


Рисунок 3 – Результат работы программы

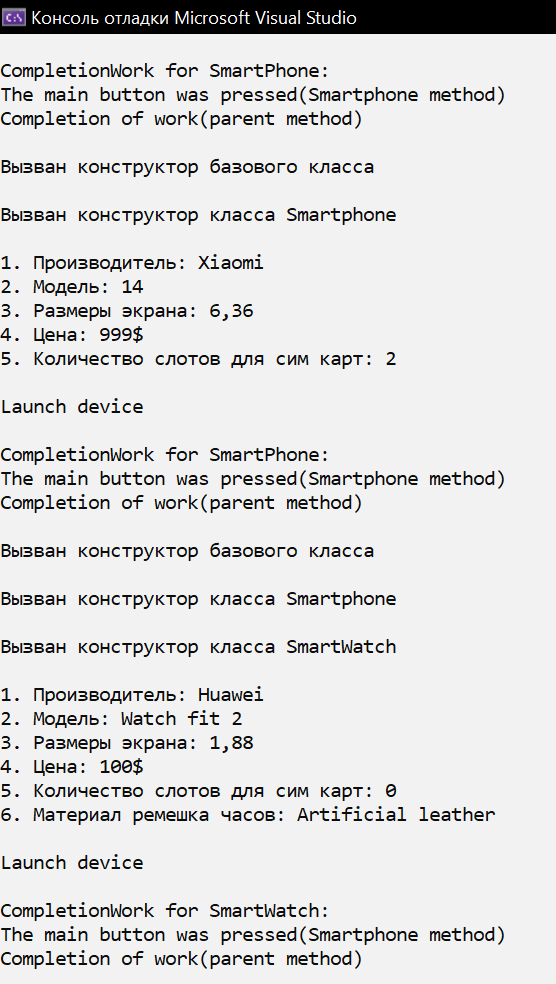


Рисунок 4 – Результат работы программы

# 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.