**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Сибирский государственный университет науки и технологий  
имени академика М.Ф. Решетнева»**

Институт информатики и телекоммуникаций

Кафедра информатики и вычислительной техники

**ЛАБОРАТОРНАЯ РАБОТА №3**

по дисциплине «Объектно-ориентированное программирование»

Вариант 21

Выполнил студент группы БПИ20-02

Очной формы обучения Сухачев Р. А.

Руководитель: канд. техн наук, доцент Вдовенко В.В.

Дата сдачи: «\_\_\_\_\_»\_\_\_\_\_\_\_\_ 2022 г.

Дата защиты: «\_\_\_\_»\_\_\_\_\_\_\_\_2022 г.

Красноярск 2022 г.

# ЦЕЛЬ РАБОТЫ

Получить практические навыки реализации одной из базовых концепций ООП – концепции наследования классов.

# порядок выполнения работы

1. Изучить теоретические сведения и примеры, представленные в гл. 8 «Наследование» учебного пособия В. В. Вдовенко «Разработка приложений на языке С*#*»(см.с. 167–187).

2. Изучить возможности языка UML для представления классов.

3. Ответить на контрольные вопросы.

4. Разработать классы в соответствии с вариантом практического задания и создать консольное приложение.

5. Разработать **UML**-диаграмму классов для созданного приложения

6. Составить отчет в электронном виде, который должен содержать титульный лист, цель лабораторной работы, задание, ответы на контрольные вопросы, листинг программы и результаты ее работы.

# постановка задачи

1. Изучить средства языка С# для реализации отношения наследования между классами;
2. Изучить возможности языка моделирования UML для представления отношений наследования между классами;
3. Выполнить практическое задание по разработке приложения на языке С#;

При выполнении этих заданий требуется написать законченную программу, в которой реализуется простое наследование классов.   
В программенеобходимо описать базовый и производные классы.   
Базовый класс (возможно, абстрактный) с помощью виртуальных или абстрактных методов и свойств должен задавать интерфейс дляпроизводных классов. Во всех классах следует переопределить метод Equals, чтобы обеспечить сравнение значений, а не ссылок. Функция Mainдолжна содержать массив из элементов базового класса, заполненный ссылками на производные классы. В этой функции должно демонстрироваться использование всех разработанных элементов классов. Также необходимо построить UML-диаграмму классов.

21. Создать иерархию классов Электроннаявычислительнаямашина, Персональныйкомпьютер, Ноутбук. КлассЭлектроннаявычислительнаямашина должен содержать атрибуты и методы, общие для производных классов. Основная программа должна создавать массивы объектов производных классов и выводить их на экран.

1. Разработать UML-диаграмму классов.

# ХОД РАБОТЫ

1) Изучил средства языка С# для реализации отношения наследования между классами;

2) Изучил возможности языка моделирования UML для представления отношений наследования между классами;

3) Выполнил практическое задание по разработке приложения на языке С#;

Листинг программы:

using System;

namespace Lab2 // 21. Создать иерархию классов Электронная вычислительная машина, Персональный компьютер, Ноутбук.

// Класс Электронная вычислительная машина должен содержать атрибуты и методы, общие для производных классов.

// Основная программа должна создавать массивы объектов производных классов и выводить их на экран.

// Базовый класс (возможно, абстрактный) с помощью виртуальных или абстрактных методов и свойств должен задавать интерфейс для производных классов.

// Во всех классах следует переопределить метод Equals, чтобы обеспечить сравнение значений, а не ссылок.

// Функция Main должна содержать массив из элементов базового класса, заполненный ссылками на производные классы.

{

abstract class Electronic\_Computer

{

string name;

string firma;

double weight;

protected Electronic\_Computer()

{

Name = "Нет информации";

Firma = "Нет информации";

Weight = 1.0;

}

protected Electronic\_Computer(string name, string firma, double weight)

{

Name = name;

Firma = firma;

Weight = weight;

}

public virtual void print()

{

Console.WriteLine($"Название: {Name}, фирма: {Firma}, вес: {Weight}.");

}

public virtual void scan()

{

Console.Write("\nВведите название: ");

Name = Console.ReadLine()!;

Console.Write("\nВведите название фирмы производителя: ");

Firma = Console.ReadLine()!;

try

{

Console.Write("\nВведите вес: ");

Weight = Double.Parse(Console.ReadLine()!);

}

catch (FormatException)

{

Console.WriteLine("\nВы ввели символ, введите число!");

Console.Write("\nВведите вес: ");

Weight = Double.Parse(Console.ReadLine()!);

}

}

protected string Name

{

set

{

if (value == "")

Console.WriteLine("Значение не может быть пустым!\n");

else

name = value;

}

get { return name; }

}

protected string Firma

{

set

{

if (value == "")

Console.WriteLine("Значение не может быть пустым!\n");

else

firma = value;

}

get { return firma; }

}

protected double Weight

{

set

{

if (value < 1)

Console.WriteLine("Значение не может быть отрицательным!\n");

else

weight = value;

}

get { return weight; }

}

public override bool Equals(Object obj)

{

if ((obj == null) || !this.GetType().Equals(obj.GetType()))

{

return false;

}

else

{

Electronic\_Computer p = (Electronic\_Computer) obj;

return (Name == p.Name) && (Firma == p.Firma) && (Weight == p.Weight);

}

}

}

class PC : Electronic\_Computer

{

string corpus;

public PC()

: base()

{

Corpus = "Нет информации";

}

public PC(string name, string firma, double weight, string corpus)

: base(name, firma, weight)

{

Corpus = corpus;

}

string Corpus

{

set

{

if (value == "")

Console.WriteLine("Значение не может быть пустым!\n");

else

corpus = value;

}

get {return corpus;}

}

public override void print()

{

Console.WriteLine("Персональный компьютер");

base.print();

Console.WriteLine($"Тип корпуса: {Corpus}.\n");

}

public override void scan()

{

base.scan();

Console.Write("\nВведите тип корпуса: ");

Corpus = Console.ReadLine()!;

}

public override bool Equals(Object obj)

{

if ((obj == null) || !this.GetType().Equals(obj.GetType()))

{

return false;

}

else

{

PC p = (PC) obj;

return base.Equals((Electronic\_Computer)obj) && (Corpus == p.Corpus);

}

}

}

class Laptop: Electronic\_Computer

{

double battery;

public Laptop()

: base()

{

Battery = 1.0;

}

public Laptop(string name, string firma, double weight, double battery)

: base(name, firma, weight)

{

Battery = battery;

}

double Battery

{

set

{

if (value < 1)

Console.WriteLine("Значение не может быть отрицательным!\n");

else

battery = value;

}

get {return battery;}

}

public override void print()

{

Console.WriteLine("Ноутбук");

base.print();

Console.WriteLine($"Батарея: {Battery}.\n");

}

public override void scan()

{

base.scan();

try

{

Console.Write("\nВведите количество батареи: ");

Battery = Double.Parse(Console.ReadLine()!);

}

catch (FormatException)

{

Console.WriteLine("\nВы ввели символ, введите число!");

Console.Write("\nВведите количество батареи: ");

Battery = Double.Parse(Console.ReadLine()!);

}

}

public override bool Equals(Object obj)

{

if ((obj == null) || !this.GetType().Equals(obj.GetType()))

{

return false;

}

else

{

Laptop p = (Laptop) obj;

return base.Equals((Electronic\_Computer)obj) && (Battery == p.Battery);

}

}

}

class Program

{

static void Main(string[] args)

{

Console.Clear();

Console.Write("\nЦель данной программы заполнение данных о электронных вычислительных машинах с последующим их выводом на экран,");

Console.Write("а так же вычисление площади и объёма комнат.\n");

var A = new Electronic\_Computer[0];

int option = 0;

while (option != 10)

{

try

{

Console.Write("\n1. Заполнение массива данных;\n2. Сравнение объектов;\n10. ВЫХОД.\nВвод: ");

option = Convert.ToInt32(Console.ReadLine()!);

}

catch (FormatException)

{

option = 11;

}

switch (option)

{

case 1:

{

int size = 0;

try

{

Console.Write("\nВведите количество электронных вычислительных машин: ");

size = Int32.Parse(Console.ReadLine()!);

}

catch (FormatException)

{

Console.WriteLine("\nВы ввели символ, введите число!");

Console.Write("\nВведите количество электронных вычислительных машин: ");

size = Int32.Parse(Console.ReadLine()!);

}

A = new Electronic\_Computer[size];

for (int i = 0; i < size; i++)

{

int key;

Console.Write("\nВыберите какой объект создать и заполнить.\n1. Персональный компьютер;\n2. Ноутбук.\nВвод: ");

key = Int32.Parse(Console.ReadLine()!);

if (key == 1)

{

A[i] = new PC();

A[i].scan();

}

else if (key == 2)

{

A[i] = new Laptop();

A[i].scan();

}

else

Console.WriteLine("\nНеверный ввод. Введите 1 или 2.\n");

}

for (int i = 0; i < size; i++)

{

Console.WriteLine($"\n№{i+1}");

A[i].print();

}

}

break;

case 2:

{

if (A.Length <= 0)

{

Console.WriteLine("\nСначала заполните массив!");

break;

}

else if (A.Length == 1)

{

Console.WriteLine("\nУ вас только один объект!");

break;

}

try

{

int first = 0, second = 0;

try

{

Console.Write("\nВведите номер первого объекта сравнения: ");

first = Int32.Parse(Console.ReadLine()!);

}

catch (FormatException)

{

Console.WriteLine("\nВы ввели символ, введите число!");

Console.Write("\nВведите номер первого объекта сравнения: ");

first = Int32.Parse(Console.ReadLine()!);

}

try

{

Console.Write("\nВведите номер второго объекта сравнения: ");

second = Int32.Parse(Console.ReadLine()!);

}

catch (FormatException)

{

Console.WriteLine("\nВы ввели символ, введите число!");

Console.Write("\nВведите номер второго объекта сравнения: ");

second = Int32.Parse(Console.ReadLine()!);

}

if (first == second)

{

Console.WriteLine("\nВы выбрали один и тот же объект!");

break;

}

if (A[first-1].Equals(A[second-1]))

Console.WriteLine("\nОбъекты равны!");

else

Console.WriteLine("\nОбъекты не равны!");

}

catch (IndexOutOfRangeException e)

{

Console.WriteLine();

Console.WriteLine(e.Message);

Console.WriteLine("Введите правильный номер объекта массива!");

break;

}

}

break;

case 10:

{

//exit

break;

}

default:

{

Console.WriteLine("\nНеверный ввод.\nВведите число 1 или 10.");

break;

}

}

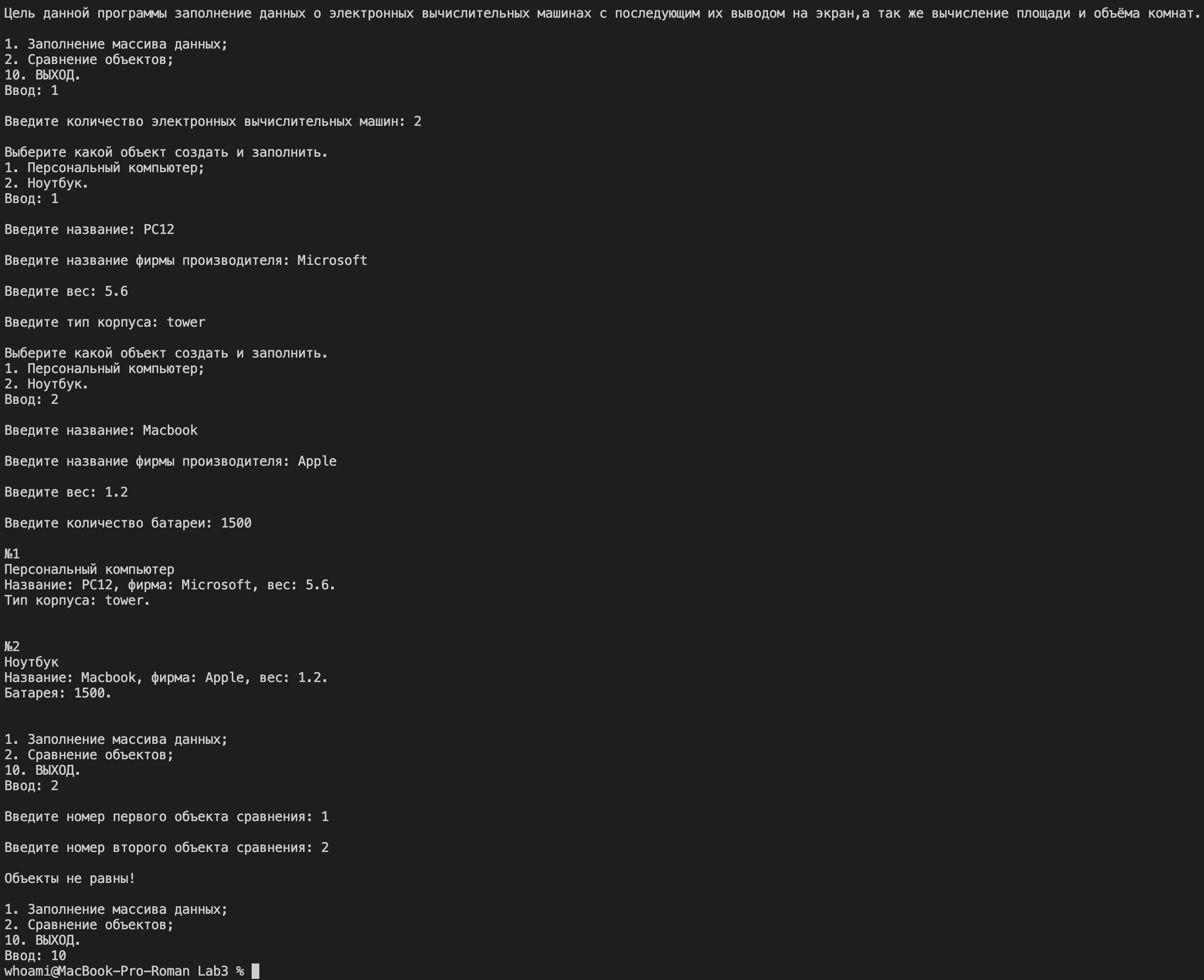
}

}

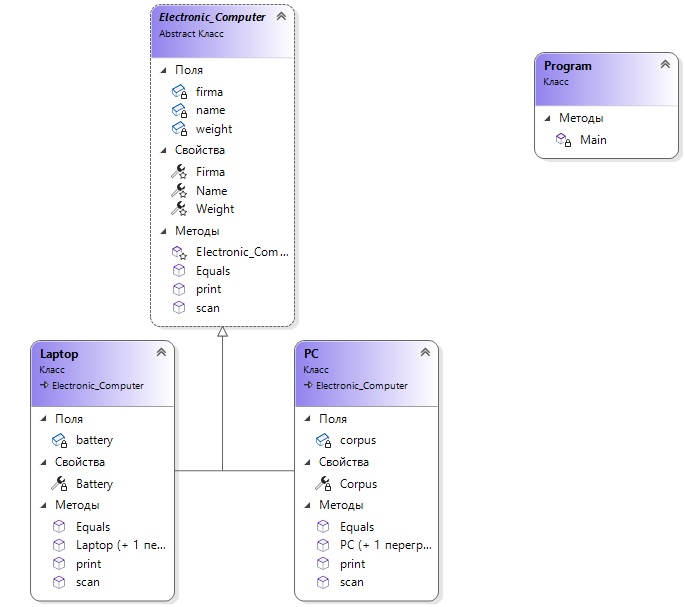
}

}

Работа программы:



4) Разработал UML-диаграмму классов.



**ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. Дайте определение простого наследования классов.

«Простое» наследование, иногда называемое одиночным наследованием, описывает родство между двумя классами: один из которых наследует второму. Из одного класса могут выводиться многие классы, но даже в этом случае подобный вид взаимосвязи остается «простым» наследованием.

2. Запишите объявление производного класса на языке C#.

class Название\_Класса\_Наследника : Базовый\_Класс {}

3. В чем состоит различие между базовым и производным классами?

Класс, члены которого наследуются, называется базовымклассом, а класс, который наследует эти члены, называется производным классом.

4. Приведите синтаксис описания наследования классов   
в общем виде. Проиллюстрируйте его фрагментом программы на языке C#.

class   Имя\_Производного\_Класса    :    спецификатор доступа    Имя\_Базового\_Класса   { /\*код\*/ } ;

class Laptop: Electronic\_Computer

{

double battery;

public Laptop()

: base()

{

Battery = 1.0;

}

public Laptop(string name, string firma, double weight, double battery)

: base(name, firma, weight)

{

Battery = battery;

}

double Battery

{

set

{

if (value < 1)

Console.WriteLine("Значение не может быть отрицательным!\n");

else

battery = value;

}

get {return battery;}

}

public override void print()

{

Console.WriteLine("Ноутбук");

base.print();

Console.WriteLine($"Батарея: {Battery}.\n");

}

public override void scan()

{

base.scan();

Console.Write("\nВведите количество батареи: ");

Battery = Double.Parse(Console.ReadLine()!);

}

public override bool Equals(Object obj)

{

if ((obj == null) || !this.GetType().Equals(obj.GetType()))

{

return false;

}

else

{

Laptop p = (Laptop) obj;

return base.Equals((Electronic\_Computer)obj) && (Battery == p.Battery);

}

}

}

5. Может ли производный класс использоваться в качестве   
базового для другого производного класса?

Да, может.

6. В каком порядке вызываются конструкторы в иерархии классов?

В иерархии классов конструкторы вызываются по порядку выведения классов: от базового к производному.

7. Каким образом выполняются создание и инициализация объектов производного класса?

1.Выделяется память для объекта дочернего класса (достаточная порция памяти для части Parent и части Child объекта класса Child);

2.Вызывается соответствующий конструктор класса Child;

3.Создается объект класса Parent с использованием соответствующего конструктора класса Parent. Если такой конструктор программистом не предоставлен, то будет использоваться конструктор по умолчанию класса Parent;

4.Список инициализации инициализирует переменные;

5.Выполняется тело конструктора класса Child;

8. Какие функции называются виртуальными? Могут ли методы-свойства класса быть виртуальными?

Виртуальная функция − это функция, которая определяется в базовом классе, а любой порожденный класс может ее переопределить.

Да, могут быть.

9. Какие особенности синтаксиса записи виртуальных функций в базовом и производных классах вам известны?

В базовом:

public virtual void print()

{

Console.WriteLine($"Название: {Name}, фирма: {Firma}, вес: {Weight}.");

}

В производном:

public override void print()

{

Console.WriteLine("Ноутбук");

base.print();

Console.WriteLine($"Батарея: {Battery}.\n");

}

10. Опишите обработку вызова виртуальной функции.

При вызове виртуального метода тип времени выполнения объекта проверяется на переопределение члена. Вызывается переопределение члена в самом дальнем классе. Это может быть исходный член, если никакой производный класс не выполнял переопределение этого члена.

11. Какой принцип ООП реализуется с помощью виртуальных функций?

Принцип полиморфизма.

12. Дайте определение абстрактного класса. Можно ли создавать объекты абстрактного класса?

Абстрактный класс – это класс, в котором объявлен хотя бы один абстрактный элемент (метод, свойство). Если в абстрактном классе объявлен абстрактныйэлемент, (метод, свойство) то перед именем такого класса ставится ключевое слово abstract.

Нет, нельзя.

13. Какой класс библиотеки FCL является базовым для всех   
остальных классов?

Класс Object.

14. Каким образом можно выполнить замещение методов класса System.Object в производных классах?

С помощью ключевого слова override.

15. В чем состоят особенности доступа к членам класса, объявленным со спецификаторомprotected?

С помощью модификатора доступа protected обеспечивается создание защищенного члена класса, доступ к которому открыт в пределах иерархии классов.

16. Каким образом отношение наследования обозначается на диаграммах классов в языке UML?

Стрелочкой от класса наследника к базовому классу.

17. Найдите в приведенном ниже фрагменте программы ошибки, связанные с реализацией механизма наследования классов и исправьте их:

**class A**

**{**

**protected int x = 1;**

**}**

**class B: A**

**{**

**int y = 2;**

**void show\_sum()**

**{**

**Console.WriteLine("x + y =" + x + y);**

**}**

**}**

# ВЫВОДЫ

Были получены практические навыки реализации одной из базовых концепций ООП – концепции наследования классов.