|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №** | | | | | | **9** |
|  | | | | | | | | |
| По дисциплине: | | Технологии программирования | | | | | | |
|  | | | | | | | | |
| Выполнил студент группы: | | | | ПИЖ-б-о-21-1 | | | | |
| ФИО полностью: | | | Кувшин Ирина Анатольевна | | | | | |
| Номер индивидуального варианта: | | | | | | 8 | | |
| Название лабораторной работы: | | | | | ПРОЕКТИРОВАНИЕ ИЕРАРХИИ КЛАССОВ | | | |

**I.**

**Учебная задача\_1:**

Код программы:

using System;

namespace L\_9

{

class Program

{

static void Main(string[] args)

{

// 6 question

/\*MainObject ob1 = new MainObject();

ob1.Draw();

Parent ob2 = new Parent();

ob2.Draw();\*/

Console.BackgroundColor = ConsoleColor.White;

Console.ForegroundColor = ConsoleColor.Blue;

Console.Clear();

Console.Title = "Лабораторная работа №11";

Human h = new Human("Иванов Иван Иванович", 50);

h.Plus(5);

h.Minus(1);

h.DrawObject();

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

Car car = new Car("Hyndai", "ix35", 120);

car.Plus(25);

car.Minus(11);

car.DrawObject();

Console.ReadKey();

}

}

// 5 & 6 questions

/\*public class MainObject : IString, IDraw

{

int data;

public void Draw()

{

Console.WriteLine(data.ToString());

}

public string GetStr()

{

return "Данные: " + data.ToString();

}

}

public interface IDraw

{

void Draw();

}

public interface IString

{

string GetStr();

}

public class Parent

{

protected int a;

}\*/

public interface ICalculate

{

void Plus(int pPlus);

void Minus(int pMinus);

}

public interface IVisual

{

string Name { get; set; }

void DrawObject();

}

public class Human : ICalculate, IVisual

{

public Human(string pFIO, int pAge)

{

FIO = pFIO;

Age = pAge;

}

private string FIO;

private int Age;

public string Name

{

get

{

return FIO + " : " + Age.ToString();

}

set

{

FIO = value;

}

}

public void Plus(int pPlus)

{

Age += pPlus;

}

public void Minus(int pMinus)

{

Age -= pMinus;

}

public void DrawObject()

{

Console.WriteLine

(

" o \n" +

" ----- \n" +

" | \n" +

" / \\ \n" +

" / \\ \n"

);

Console.WriteLine(Name);

}

}

public class Car : ICalculate, IVisual

{

public Car(string pManuFacturer, string pModel, int pVelocity)

{

Manufacturer = pManuFacturer;

Model = pModel;

Velocity = pVelocity;

}

private string Manufacturer;

private string Model;

private int Velocity;

public string Name

{

get

{

return Manufacturer + " - " + Model + " : " + Velocity.ToString() + "km/h";

}

set

{

Model = value;

}

}

public void Plus(int pPlus)

{

Velocity += pPlus;

}

public void Minus(int pMinus)

{

Velocity -= pMinus;

}

public void DrawObject()

{

Console.WriteLine

(

" --------------\n" +

"\_\_\_\_\_\_/ \\\_\_\_\_\_\n" +

"| |\n" +

"----(@)-----------------(@)--\n"

);

Console.WriteLine(Name);

}

}

}



Рисунок 9.1 – Результат выполнения учебной задачи

**Индивидуальное задание\_1:**

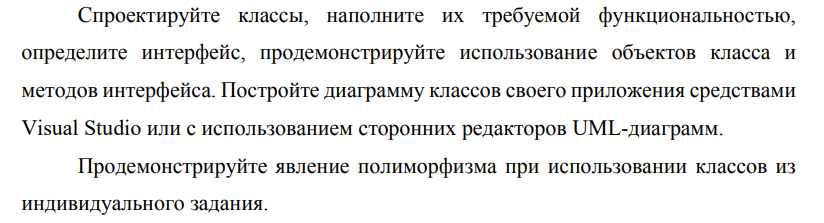
****

Рисунок 9.2 – Условие задания

Код программы:

using System;

namespace L\_9

{

class Program

{

static void Main(string[] args)

{

// 6 question

/\*MainObject ob1 = new MainObject();

ob1.Draw();

Parent ob2 = new Parent();

ob2.Draw();\*/

// Учебная задача

/\* Console.BackgroundColor = ConsoleColor.White;

Console.ForegroundColor = ConsoleColor.Blue;

Console.Clear();

Console.Title = "Лабораторная работа №11";

Human h = new Human("Иванов Иван Иванович", 50);

h.Plus(5);

h.Minus(1);

h.DrawObject();

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

Car car = new Car("Hyndai", "ix35", 120);

car.Plus(25);

car.Minus(11);

car.DrawObject();\*/

// Ind\_task

Трава chamomile = new Трава("Ромашка аптечная", ЖизненнаяФорма.трава, ПораЦветения.лето, Травянистые.однолетние);

Console.WriteLine(">>>>>>>>>>>>> " + chamomile.GetType().Name);

Console.WriteLine(chamomile.inf);

chamomile.DrawObject();

Console.WriteLine("\n");

Кустарник spirea = new Кустарник("Спирея", ЖизненнаяФорма.кустарник, ПораЦветения.весна, Куст.декоративно\_лиственные);

Console.WriteLine(">>>>>>>>>>>>> " + spirea.GetType().Name);

Console.WriteLine(spirea.inf);

spirea.DrawObject();

Console.WriteLine("\n");

Дерево fir = new Дерево("Пихта", ЖизненнаяФорма.дерево, ПораЦветения.лето, Листья.хвойные);

Console.WriteLine(">>>>>>>>>>>>> " + fir.GetType().Name);

Console.WriteLine(fir.inf);

fir.DrawObject();

Console.WriteLine("\n");

}

}

// 5 & 6 questions

/\*public class MainObject : IString, IDraw

{

int data;

public void Draw()

{

Console.WriteLine(data.ToString());

}

public string GetStr()

{

return "Данные: " + data.ToString();

}

}

public interface IDraw

{

void Draw();

}

public interface IString

{

string GetStr();

}

public class Parent

{

protected int a;

}\*/

public interface ICalculate

{

void Plus(int pPlus);

void Minus(int pMinus);

}

public interface IVisual

{

string Name { get; set; }

void DrawObject();

}

public class Human : ICalculate, IVisual

{

public Human(string pFIO, int pAge)

{

FIO = pFIO;

Age = pAge;

}

private string FIO;

private int Age;

public string Name

{

get

{

return FIO + " : " + Age.ToString();

}

set

{

FIO = value;

}

}

public void Plus(int pPlus)

{

Age += pPlus;

}

public void Minus(int pMinus)

{

Age -= pMinus;

}

public void DrawObject()

{

Console.WriteLine

(

" o \n" +

" ----- \n" +

" | \n" +

" / \\ \n" +

" / \\ \n"

);

Console.WriteLine(Name);

}

}

public class Car : ICalculate, IVisual

{

public Car(string pManuFacturer, string pModel, int pVelocity)

{

Manufacturer = pManuFacturer;

Model = pModel;

Velocity = pVelocity;

}

private string Manufacturer;

private string Model;

private int Velocity;

public string Name

{

get

{

return Manufacturer + " - " + Model + " : " + Velocity.ToString() + "km/h";

}

set

{

Model = value;

}

}

public void Plus(int pPlus)

{

Velocity += pPlus;

}

public void Minus(int pMinus)

{

Velocity -= pMinus;

}

public void DrawObject()

{

Console.WriteLine

(

" --------------\n" +

" / \\ \_\n" +

"| |\n" +

"----(@)-----------------(@)--\n"

);

Console.WriteLine(Name);

}

}

// Ind\_task

public interface IDrawImg

{

string Name { get; set; }

void DrawObject();

}

class Растение: IDrawImg

{

public string Name

{

get

{

return name + ": " + vital\_form.ToString();

}

set

{

name = value;

}

}

public void DrawObject()

{

Console.WriteLine(@"РАСТЕНИЕ");

Console.WriteLine(Name);

}

public string name;

public ЖизненнаяФорма vital\_form;

public ПораЦветения flowering\_time;

// Пустой конструктор

public Растение()

{

name = "нет данных";

vital\_form = ЖизненнаяФорма.информация\_отсутствует;

flowering\_time = ПораЦветения.информация\_отсутствует;

}

// Конструктор с параметрами

public Растение(string n, ЖизненнаяФорма v, ПораЦветения f)

{

name = n;

vital\_form = v;

flowering\_time = f;

}

// Переопределяемое свойство inf

public virtual string inf

{

get { return "Название: " + name + " Жизненная форма: " + vital\_form + " Время цветения: " + flowering\_time; }

}

}

class Трава : Растение

{

public new string Name

{

get

{

return name + ": " + vital\_form.ToString();

}

set

{

name = value;

}

}

public new void DrawObject()

{

Console.WriteLine(@"

$$$$$$

$$$$$$$$

$$$$$$$ $$$$$$ $$$$$$

$$$$$$$$$ $$$$$ $$$$$$$$

$$$$$$$$$ $$$$$ $$$$$$$$

$$$$$$$$ $$$$ $$$$$$$

$$$$$ $$$ $$$$$

$$$$$$$$$ $ $ $$$$$$$$$

$$$$$$$$$$ $ $ $$$$$$$$$$

$$$$$$$$$$$ $$$$$$$$$$$

$$$$$$$$$ $ $ $$$$$$$$$

$$$$$ $$$ $$$$$

$$$$$$$$ $$$$ $$$$$$$

$$$$$$$$$ $$$$$ $$$$$$$$

$$$$$$$$$ $$$$$ $$$$$$$$

$$$$$$$ $$$$$$ $$$$$$

$$$$$$$$ $$

$$$$$$ $$

$$$$$ $$

$$$$$$$ $$

$$$$$$$$$$$ $$

$$$$$$$$$ $$

$$$$$$$ $$

$$$$$ $$

$$$$$$

$$$$

$$

$$ $$$$$$$

$$ $$$$$$$$$$

$$ $$$$$$$$

$$ $$$$$$

$$ $$$$$

$$$$$$

$$$$

$$

");

Console.WriteLine(Name);

}

public Травянистые life\_expectancy;

// Пустой конструктор

public Трава()

: base()

{

life\_expectancy = Травянистые.информация\_отсутствует;

}

// Конструктор с параметрами

public Трава(string n, ЖизненнаяФорма v, ПораЦветения f, Травянистые t)

{

name = n;

vital\_form = v;

flowering\_time = f;

life\_expectancy = t;

}

public override string inf

{

get { return base.inf + " Продолжительность жизни: " + life\_expectancy.ToString(); }

}

}

class Кустарник : Растение

{

public new string Name

{

get

{

return name + ": " + vital\_form.ToString();

}

set

{

name = value;

}

}

public new void DrawObject()

{

Console.WriteLine(@"

.-(¯`:´¯)-

(¯` ● \ ● / ●´¯)

(¯´●\_0()()()0\_●´¯)

(¯`● ()()()()()() ●´¯)

(\_. ●¯0()()()0¯●.\_)

(\_.●´/ ● \`●.\_)

) '-(\_.:.\_)

(¯ `·.\|/.·´¯) █

(¯ `·.(█).·´¯)█ (¯`:´¯)

(\_.·´/|\`·.\_)█ (·.(█)`.)

(\_.:.\_) █ (\_.:.\_)

██████████

$$$$$$$$$$$$$$$$$$$$

$$$$$$$$$$$$$$$$$$$$

$$$$$$$$$$$$$$$$$$

$$$$$$$$$$$$$$$$

$$$$$$$$$$$$$$

██████████");

Console.WriteLine(Name);

}

public Куст type;

// Пустой конструктор

public Кустарник()

: base()

{

type = Куст.информация\_отсутствует;

}

// Конструктор с параметрами

public Кустарник(string n, ЖизненнаяФорма v, ПораЦветения f, Куст k)

{

name = n;

vital\_form = v;

flowering\_time = f;

type = k;

}

public override string inf

{

get { return base.inf + " Тип кустарника: " + type.ToString(); }

}

}

class Дерево : Растение

{

public new string Name

{

get

{

return name + ": " + vital\_form.ToString();

}

set

{

name = value;

}

}

public new void DrawObject()

{

Console.WriteLine(@"

¶¶

¶¶¶¶

¶¶ 11 ¶1

¶¶¶¶¶¶¶¶¶¶¶

1¶¶11¶111111¶¶1

¶¶¶111111111111¶¶

¶¶¶111111111111111¶¶¶

¶¶¶¶¶1111111111111111¶¶¶

¶¶¶¶1 1¶¶11111111111¶¶1¶¶¶¶

¶¶ 1¶¶11111111111¶¶¶

¶¶¶¶¶111111111111111¶¶

¶¶11¶¶1111111111111111¶¶¶

¶¶¶11111111111111111111111¶¶¶

¶¶¶11111111111111111111¶11111¶¶1

¶¶¶1111111111111111111111¶¶¶1111¶¶¶

¶¶¶¶¶1¶¶111111111111111¶¶¶1\_1¶¶1¶¶¶¶¶1

¶¶¶1111111111111111¶¶ 1¶¶¶

1¶¶111¶¶¶1111111111111¶¶¶¶¶1¶¶¶

¶¶¶11¶¶¶111¶¶1111111111¶¶111111¶¶1

¶¶¶1111¶¶1 1¶11111111111111111111¶¶¶

¶¶¶11111111¶¶¶¶¶1111111111111111111111¶¶¶

1¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶111111111¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶

¶¶¶¶¶¶¶¶¶¶¶¶¶1¶¶¶1¶¶¶¶¶¶¶¶¶1¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶¶

¶¶¶¶¶¶¶1

¶¶¶¶¶¶¶¶

¶¶¶¶¶¶¶¶

¶¶¶¶¶¶¶¶

¶¶¶¶¶¶¶¶

");

Console.WriteLine(Name);

}

public Листья type;

// Пустой конструктор

public Дерево()

: base()

{

type = Листья.информация\_отсутствует;

}

// Конструктор с параметрами

public Дерево(string n, ЖизненнаяФорма v, ПораЦветения f, Листья l)

{

name = n;

vital\_form = v;

flowering\_time = f;

type = l;

}

public override string inf

{

get { return base.inf + " Тип листьев: " + type.ToString(); }

}

}

// Данные типов-перечислений

public enum ЖизненнаяФорма

{

трава,

кустарник,

дерево,

информация\_отсутствует

}

public enum ПораЦветения

{

зима,

весна,

лето,

осень,

информация\_отсутствует

}

public enum Травянистые

{

однолетние,

двулетние,

многолетние,

информация\_отсутствует

}

public enum Куст

{

цветущие,

декоративно\_лиственные,

плодовые,

информация\_отсутствует

}

public enum Листья

{

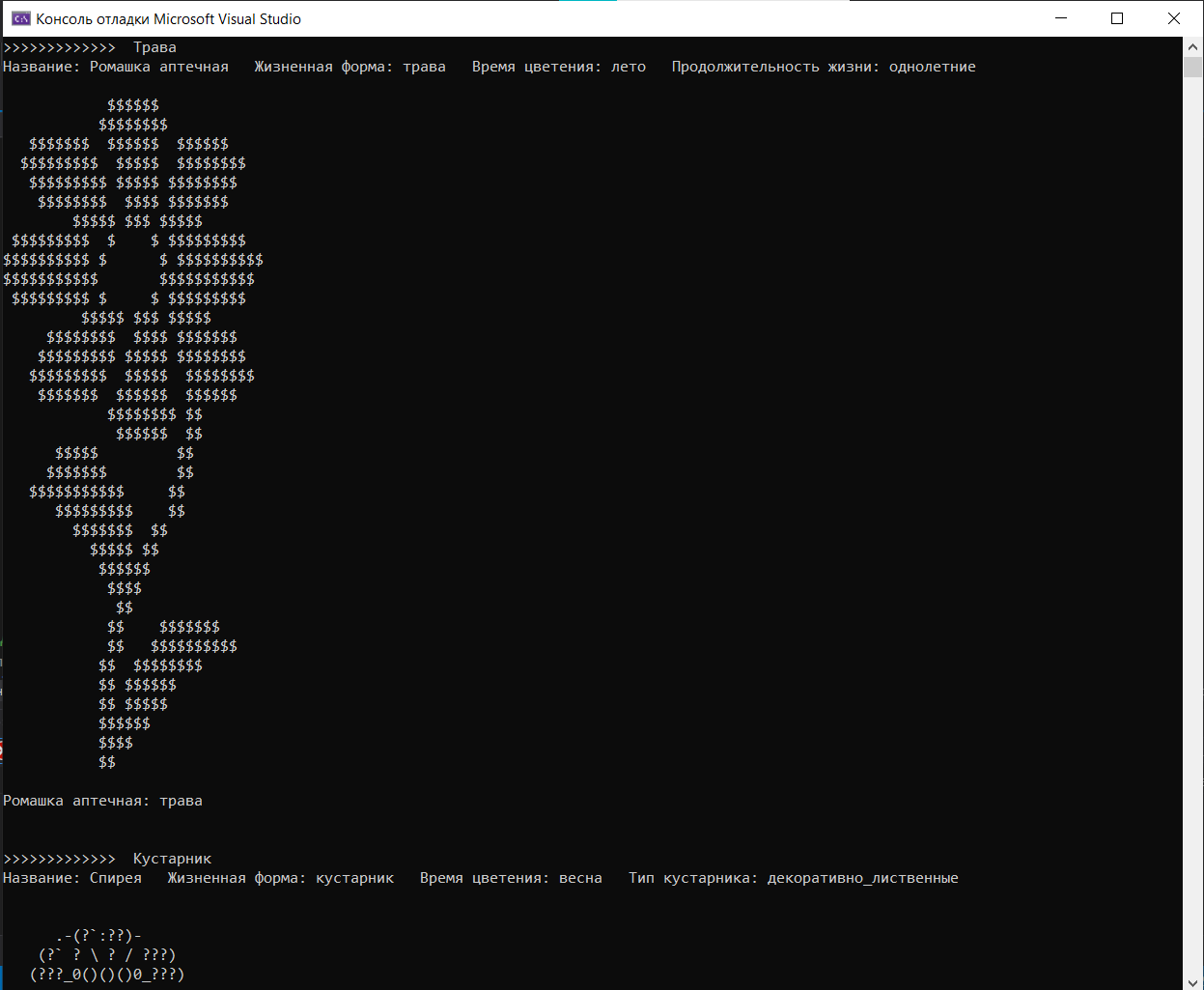
хвойные,

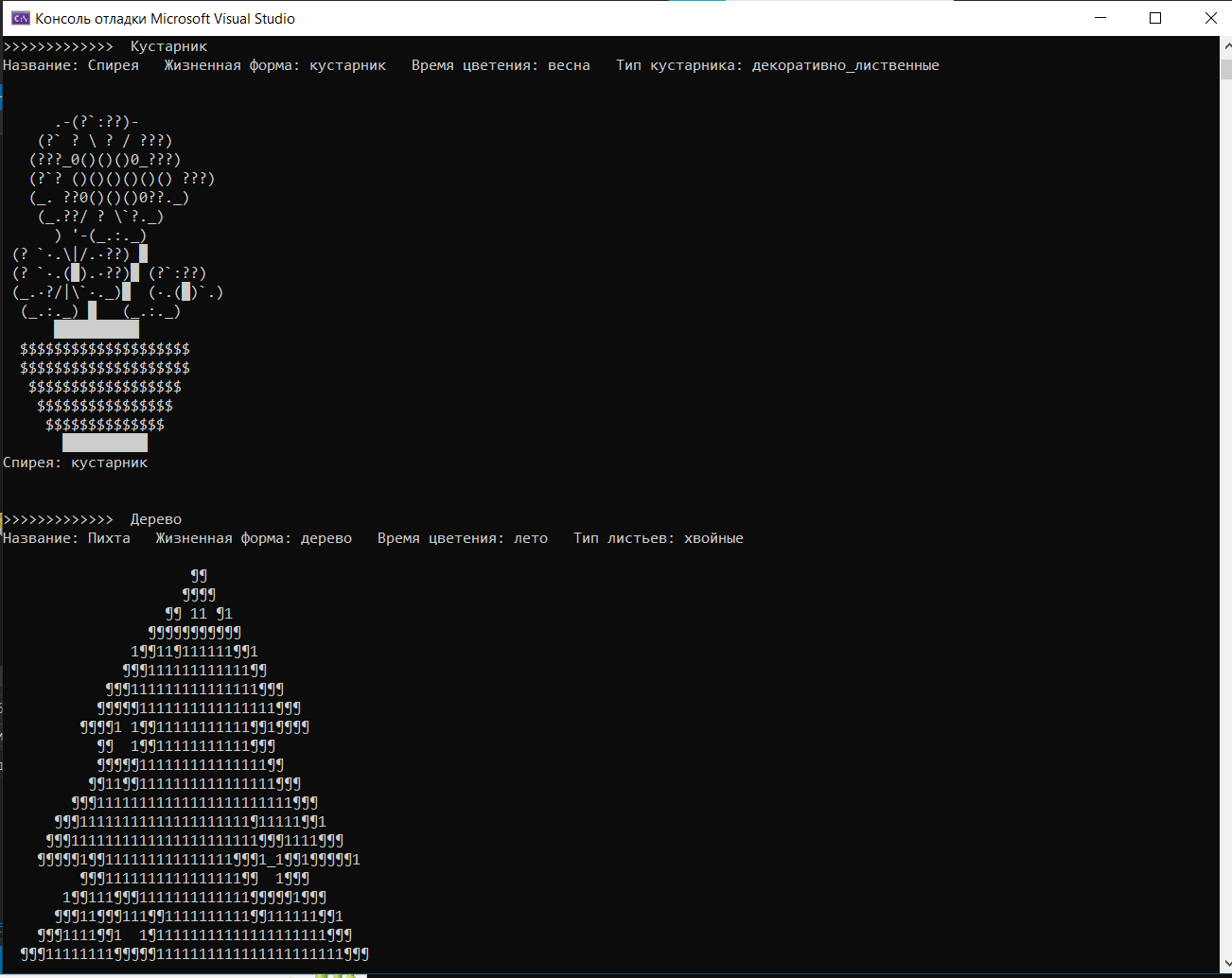
лиственные,

информация\_отсутствует

}

}





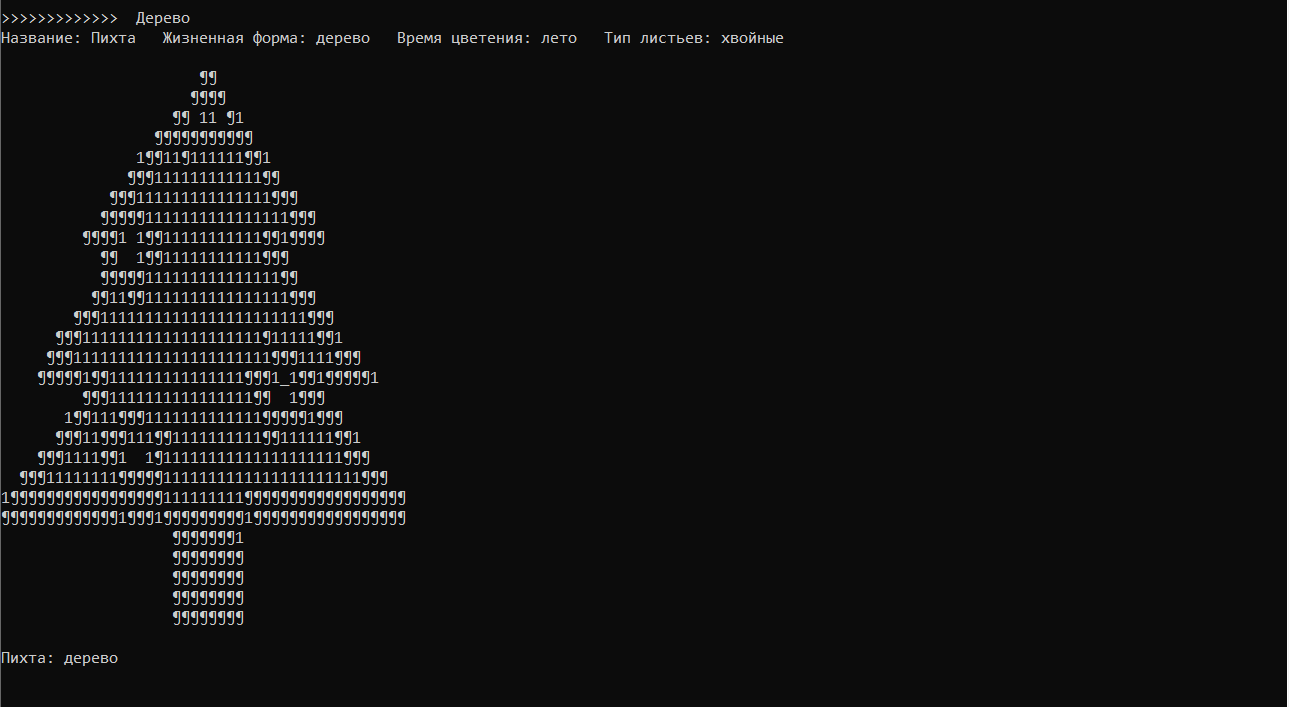


Рисунок 9.3 – Результат выполнения программы

**II. Ответы на контрольные вопросы:**

1. Чем отличается наследование интерфейсов от наследования реализации?

Наследование реализации (implementation inheritance) означает, что тип происходит от базового типа, получая от него все поля-члены и функции члены.

При наследовании реализации производный класс наследует реализацию каждой функции базового типа, если только в его определении не указано, что реализация функции должна быть переопределена.

Кроме наследования реализации (implementation inheritance) в языке C# реализован механизм наследования интерфейсов (interface inheritance).

Отличие от механизма наследования реализации состоит в том, что:

- класс обязательно должен реализовать методы и свойства, продекларированные в интерфейсе (никакой реализации в интерфейсе не существует);

- класс может реализовывать (наследовать) несколько интерфейсов, тогда как базовый класс может быть только один.

2. Поясните, каким образом проявляется полиморфное поведение объектов при реализации классами интерфейсов?

Полиморфизм (от греч. poly — много и morphe — форма) — один из главных столпов объектно-ориентированного программирования. Его суть заключается в том, что один фрагмент кода может работать с разными типами данных.

В интерфейсе ни у одного из методов не должно быть тела. Это означает, что в интерфейсе вообще не предоставляется никакой реализации. В нем указывается только, что именно следует делать, но не как это делать. Как только интерфейс будет определен, он может быть реализован в любом количестве классов. Кроме того, в одном классе может быть реализовано любое количество интерфейсов.

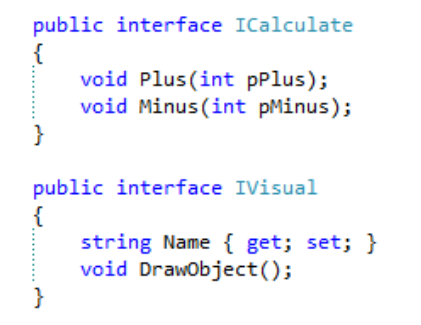
Для реализации интерфейса в классе должны быть предоставлены тела (т.е. конкретные реализации) методов, описанных в этом интерфейсе. Каждому классу предоставляется полная свобода для определения деталей своей собственной реализации интерфейса. Следовательно, один и тот же интерфейс может быть реализован в двух классах по-разному.

3. Сколько интерфейсов может реализовывать класс?

Неограниченное кол-во.

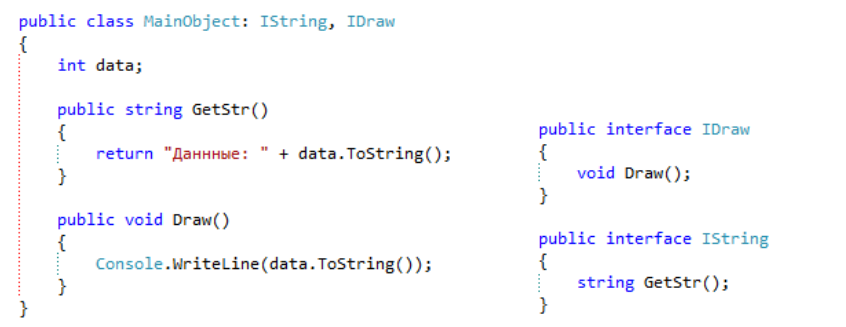
4. Как объявляются интерфейсные типы? Для чего используются интерфейсы?

Для объявления типа интерфейса используется ключевое слово interface:

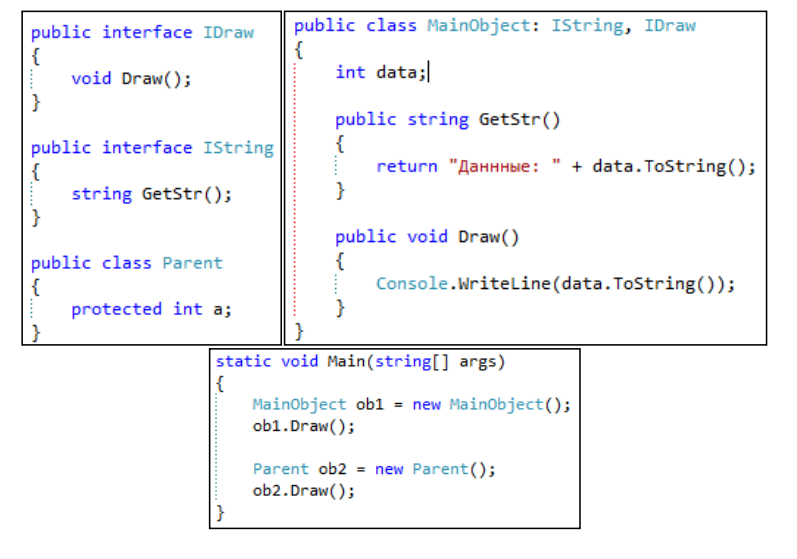
****

Интерфейсы используются для того, чтобы указать классам, что именно нужно реализовать в этих классах. Реализовывать нужно методы (свойства, индексаторы, события). Таким образом, интерфейс описывает функциональные возможности без конкретной реализации. Иными словами интерфейс определяет спецификацию но не реализацию.

5. Даны определения нескольких типов. Укажите ошибки (если есть) в представленном фрагменте:

****

6. Даны определения нескольких типов. Укажите ошибки (если есть) в представленном фрагменте:



Объекты класса Parent не содержат реализации интерфейса Draw.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Серьезность | Код | Описание | Проект | Файл | Строка | Состояние подавления |
| Ошибка | CS1061 | "Parent" не содержит определения "Draw", и не удалось найти доступный метод расширения "Draw", принимающий тип "Parent" в качестве первого аргумента (возможно, пропущена директива using или ссылка на сборку). | L\_9 | C:\Users\kuvsh\Desktop\СКФУ\2\_3\_семестр\Технологии программирования\Programming\_technologies\L\_9\Program.cs | 14 | Активные |

**Дата: 04.12.2022**