P = 3; b– суммарное число сообщений, обработанное в каждом PIR-узле за 10 е.м.в. ≤ 2400 (максимальная интенсивность), P = 3.

Таблица: Результаты прогона модели

Буферна	K_m - глубина буферной памяти, m – номер прогона модели (m=1,2,..10)									
япамять	K_1	K_2	K_3	K_4	K_5	K_6	K_7	K_8	K_9	K_{10}
FifoDS1	1/2	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1
FifoPIR1	1/8	1/7	1/7	1/6	1/7	1/6	1/6	1/7	1/4	1/8
FifoDS2	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1
FifoPIR2	1/9	1/6	1/6	1/6	1/6	1/6	1/6	1/8	1/6	1/8
FifoDS3	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1	1/1
FifoPIR3	1/8	1/7	1/8	1/7	1/8	1/6	1/5	1/6	1/7	1/7

Библиографический список

1. J. Yun and M. Song, "Detecting Direction of Movement Using Pyroelectric Infrared Sensors," in *IEEE Sensors Journal*, vol. 14, no. 5, pp. 1482-1489, May 2014, doi: 10.1109/JSEN.2013.2296601
2. T. Yang, P. Guo, W. Liu, X. Liu and T. Hao, "Enhancing PIR-Based Multi-Person Localization Through Combining Deep Learning With Domain Knowledge," in *IEEE Sensors Journal*, vol. 21, no. 4, pp. 4874-4886, 15 Feb.15, 2021, doi: 10.1109/JSEN.2020.3029810T.
3. V. A. Kokovin, A. A. Evsikov, S. U. Uvaysov, A. S. Uvaysova and V. I. Nefedov, "Scanning Network for Solving Navigation Problems of Autonomous Vehicles," *2020 International Conference on Electrotechnical Complexes and Systems (ICOECS)*, Ufa, Russia, 2020, pp. 1-6, doi: 10.1109/ICOECS50468.2020.9278405.
4. ESA (European Space Agency), standard ECSS-E-50-12A, “Space engineering. SpaceWire – Links, nodes, routers and networks. European cooperation for space standardization”, ESA Publications Division ESTEC, Noordwijk, The Netherlands, 2003.
5. В.А. Коковин, А.А. Евсиков, А.П. Леонов Особенности организации и взаимодействия функциональных сетевых компонентов в распределенных управляющих системах // Информационные технологии. 2021. Т.27, №4. С. 212—224.

УДК 004.415.22

Макаров А.О.

НАСЛЕДОВАНИЕ КЛАССОВ В C#

CLASS INHERITANCE IN C#

Филиал «Протвино» государственного университета «Дубна»
Секция «Информационные технологии»

Автор: Макаров Артем Олегович, студент 1 курса направления «Информатика и вычислительная техника» филиала «Протвино» государственного университета «Дубна».

Научный руководитель: Кульман Татьяна Николаевна, кандидат технических наук, доцент кафедры информационных технологий филиала «Протвино» государственного университета «Дубна».

Author: Makarov Artyom Olegovich, 1t year student of the direction "Informatics and computer engineering" of the branch "Protvino" state University "Dubna".

Scientific adviser: Kulman Tatiana Nikolaevna, candidate of technical sciences, associate professor of the department information technology of the branch "Protvino" state University "Dubna".

Аннотация

Рассматривается механизм наследования классов в C#. Изучается принцип полиморфизма.

Abstract

The class inheritance mechanism in C# is considered. The polymorphism principle is investigated.

Ключевые слова: C#, ООП, класс, объектно-ориентированное программирование, наследование классов, полиморфизм, UML, UML-диаграммы.

Keywords: C#, OOP, object-oriented programming, class inheritance, polymorphism, UML, UML diagrams.

Целью работы является разработка иерархии классов с использованием механизмов наследования C# для описания электронных компонентов.

Задачи, которые необходимо выполнить для достижения цели:

- Исследование предметной области
- Изучение механизма наследования классов в C#
- Изучение диаграммы классов языка моделирования UML
- Графическое представление иерархии классов на UML
- Практическое применение механизма наследования
- Реализация класса запросов, обращающегося к иерархии

Актуальность: Наследование является мощным инструментом объектно-ориентированного программирования (ООП). Классы потомков получают свойства классов-предков и могут дополнять их или изменять. Таким образом, наследование обеспечивает важную возможность многократного использования кода. Кроме того, наследование является единственной возможностью использовать объекты, исходный код которых недоступен, но в которые требуется внести изменения.

Исследование предметной области

Любое электронное устройство состоит из таких компонентов, как резисторы, конденсаторы, интегральные схемы и т.д.

Представим некоторые из этих компонентов в виде классов. Каждый такой класс будет иметь соответствующие описываемому компоненту поля, свойства-методы для доступа к ним [1, 2] (только чтение), а также метод Passport() для вывода информации о компоненте. На основе этих классов будет построена иерархия. Всего будет создано 6 классов: ElectronicComponent, Resistor, Capacitor, IntegratedCircuit, LogicGate, MicroController.

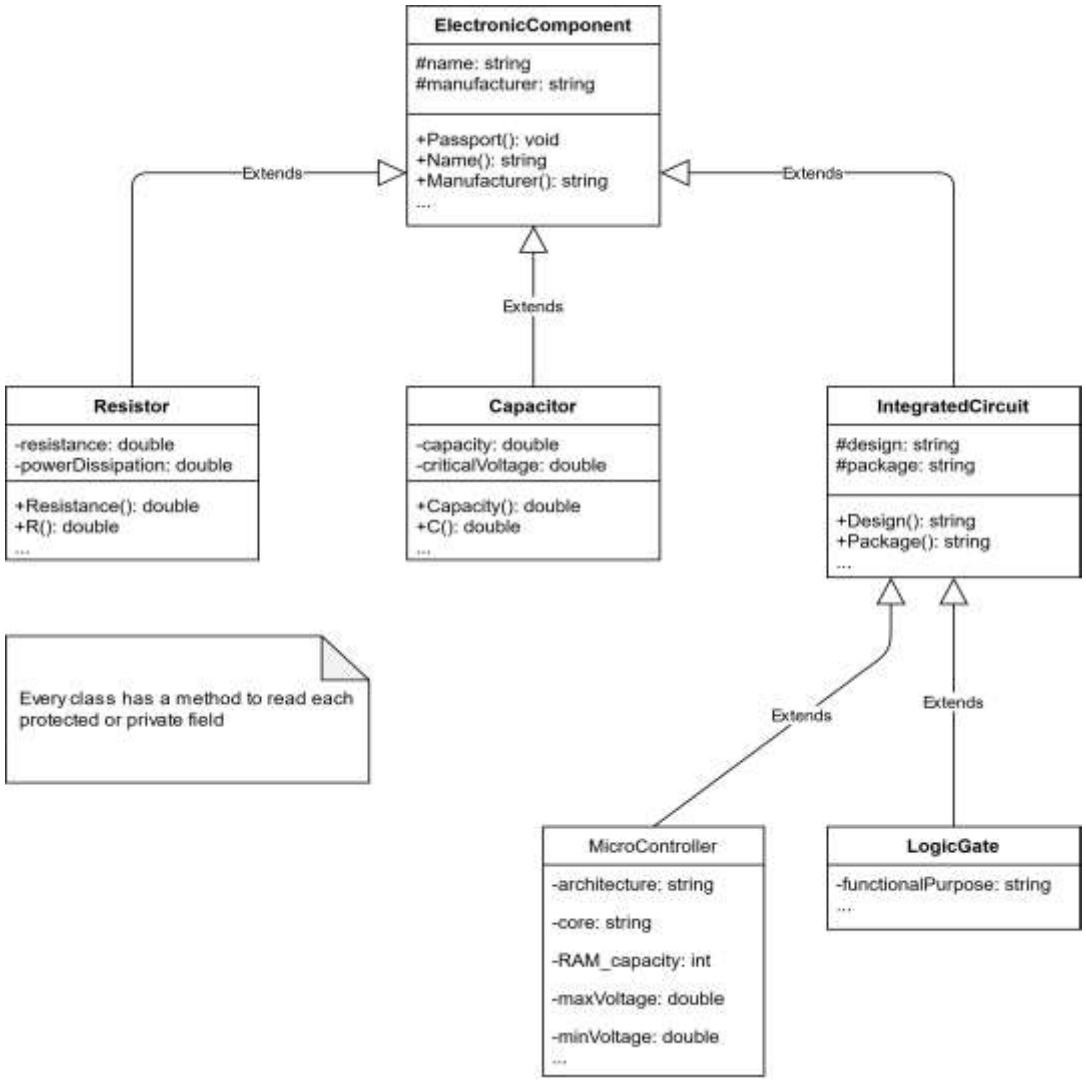

Рис. 1. Иерархия классов электронных компонентов

Далее приводится исходный код некоторых классов на C#.

```
class ElectronicComponent
{
  protected string name;
  protected string manufacturer;
  public ElectronicComponent(string name, string manufacturer = "Unknown manufacturer")
  {
    this.name = name;
    this.manufacturer = manufacturer;
  }
  public string Name
  {
    get
    {
      return this.name;
    }
  }
  public string Manufacturer
  {
    get
```

```
            {
                return this.manufacturer;
            }
        }
        public virtual void Passport()
        {
            Console.WriteLine("Name: {0}\nManufacturer: {1}", name, manufacturer);
        }
    }
class Resistor: ElectronicComponent
    {
        private double resistance;
        private double powerDissipation;
        public Resistor(double resistance, double powerDissipation, string name, string manufacturer) :
base(name, manufacturer)
        {
            this.resistance = resistance;
            this.powerDissipation = powerDissipation;
        }
        public double Resistance
        {
            get
            {
                return this.resistance;
            }
        }
        public double R
        {
            get
            {
                return this.resistance;
            }
        }
        public double PowerDissipation
        {
            get
            {
                return this.powerDissipation;
            }
        }
        public double P
        {
            get
            {
                return this.powerDissipation;
            }
        }
        public override void Passport()
        {
            base.Passport();
            Console.WriteLine("Resistance: {0} Ohms\nPower dissipation: {1} Watts", resistance,
powerDissipation);
        }
```

```
    }
class IntegratedCircuit: ElectronicComponent
    {
        protected string design;
        protected string package;
        public IntegratedCircuit(string design, string package, string name, string manufacturer =
"Unknown manufacturer"): base(name, manufacturer)
        {
            this.design = design;
            this.package = package;
        }
        public string Design
        {
            get
            {
                return this.design;
            }
        }
        public string Package
        {
            get
            {
                return this.package;
            }
        }
        public override void Passport()
        {
            base.Passport();
            Console.WriteLine("Design: {0}\nPackage: {1}", design, package);
        }
    }
```

Изучение наследования в C#

C# является полностью объектно-ориентированным языком. Это значит, что каждая программа на этом языке состоит из классов. Концепция ООП строится на трёх принципах: инкапсуляция, полиморфизм и наследование. Инкапсуляция есть не только сокрытие данных, но и предоставление методов для обработки этих данных. Полиморфизм [3] переводится с греческого как «много форм». В контексте ООП это означает много форм одного метода. Из каждого объекта вызывается метод, соответствующий типу объекта. Принцип наследования заключается в создании новых классов на основе уже существующих. Все свойства и методы базового класса включаются в класс-наследник. С помощью наследования строятся иерархии классов. Иерархии, в свою очередь, можно описать с помощью языка UML.

Язык моделирования UML

UML – унифицированный язык моделирования (Unified Modeling Language) – это система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования. UML состоит из диаграмм, сущностей и связей [4].

В языке UML представлено несколько видов диаграмм, одна из которых — диаграмма классов. Далее будет обсуждаться именно она.

Диаграммы определяют классы и различного рода связи между ними. На таких диаграммах также изображаются атрибуты классов, их методы, ограничения, накладываемые на связи между классами. Диаграммы классов позволяют наглядно отобразить логическое представление системы. На основе диаграмм создаётся исходный код описанных классов.

Практическое применение механизма наследования

Правильно применённое на практике, наследование классов может значительно облегчить разработку программы на этапах её проектирования и написания кода программы.

Следующий простой пример демонстрирует непосредственное использование наследования на C#.

Класс запросов к базе электронных компонентов

Класс Request будет иметь четыре открытых статических метода: AllComponents(), OnlyResistors(), OnlyICs() и OnlyResistorsWithCondition(). Каждый из этих методов принимает в качестве аргумента массив элементов типа ElectronicComponent. Рассмотрим исходный код некоторых из этих методов.

Метод AllComponents():

```
public static void AllComponents(ElectronicComponent[] components)
    {
      Console.WriteLine("All electronic components\n");
      foreach (ElectronicComponent ec in components)
      {
        ec.Passport();
        Console.WriteLine();
      }
    }
```

В цикле по коллекции выводится информация о каждом элементе массива с помощью метода Passport(). Однако в массиве могут присутствовать элементы разных типов из иерархии электронных компонентов, потому метод Passport() в классе ElectronicComponent объявлен с атрибутом *virtual.* В каждом классе ниже по иерархии этот метод перегружен, поэтому для каждого объекта класса из данной иерархии метод Passport() вызывается из соответствующего класса, а не из класса ElectronicComponent, в котором он изначально объявлен. Именно так и работает принцип полиморфизма, заключающийся в многообразии форм одного и того же метода в каждом классе иерархии.

Метод OnlyResistors():

```
public static void OnlyResistors(ElectronicComponent[] components)
    {
      Console.WriteLine("All resistors\n");
      Resistor res;
      for (int i = 0; i < components.Length; i++)
      {
        res = components[i] as Resistor;
        if (res != null)
        {
          res.Passport();
          Console.WriteLine();
        }
      }
    }
```

В этом методе применён оператор *as,* возвращающий ссылку на объект, если возможно приведение к типу, указанного справа от оператора или значение *null* в противном случае. Дальше в методе, соответственно, проводится проверка объекта *res* на наличие в нём действительной ссылки на объект, и если эта проверка успешна, то на экран выводится информация об объекте через метод Passport().

Демонстрация работы методов класса Request

В функции Main() создаётся массив из элементов типа ElectronicComponent. Затем туда заносятся сами элементы, имеющие типы из разных уровней иерархии. После чего последовательно вызываются методы класса Request. Метод OnlyResistorsWithCondition() выводит информацию о тех резисторах, сопротивление которых более 5000 Ом.

```
static void Main(string[] args)

{
  ElectronicComponent[] components = new ElectronicComponent[7];
```

```
        components[0] = new Resistor(4700, 0.25, "Resistor", "China");
        components[1] = new Capacitor(2200 * 1e-6, 25.0, "Сглаживающий конденсатор");
        components[2] = new IntegratedCircuit("NMOS", "DIP-40", "580ВК91А", "Квазар");
        components[3] = new MicroController("8-bit", "8051", 2048, 4.5, 5.5, "CMOS", "DIP-40",
"AT89C51", "Atmel");
        components[4] = new LogicGate("6-NOT", "TTL", "DIP-14", "К155ЛН1", "Интеграл");
        components[5] = new ElectronicComponent("6П3С", "Фотон");
        components[6] = new Resistor(10000.0, 0.25, "Подтягивающий резистор", "China");
        Request.AllComponents(components);
        Request.OnlyICs(components);
        Request.OnlyResistors(components);
        //ищутся резисторы с сопротивлением больше 5000 Ом
        Request.OnlyResistorsWithCondition(components, delegate (Resistor res) { return
res.Resistance > 5000.0; });
        Console.ReadKey();
    }
```

Рис. 2. Фрагмент результата вызова метода AllComponents()

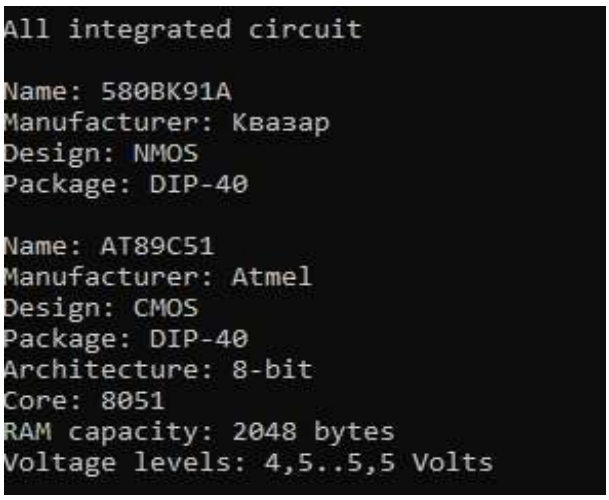

Рис. 3. Фрагмент результата вызова метода OnlyICs()

Рис. 4. Результат вызова метода OnlyResistors()

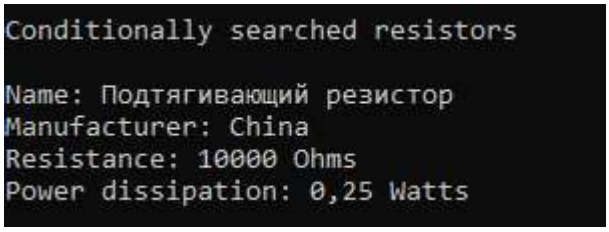

Рис. 5. Результат вызова метода OnlyResistorsWithCondition()

Результатом проделанной работы является создание иерархии классов электронных компонентов. Кроме того, выполнены все поставленные задачи, а именно:

1. Исследована предметная область
2. Изучен механизм наследования классов в C#
3. Изучена диаграмма классов языка моделирования UML
4. Графически представлена иерархия классов на UML
5. Механизм наследования применён на практике
6. Реализован класс запросов, обращающийся к иерархии классов

Библиографический список

1. Руководство по программированию на C#. Классы | Microsoft Docs
2. Методы. Руководство по программированию на C# | Microsoft Docs
3. Руководство по программированию на C#. Полиморфизм | Microsoft Docs
4. UML-диаграммы классов — https://prog-cpp.ru/uml-classes/

УДК 37.015.3

Макаров В.В.

ПСИХОЛОГИЧЕСКИЕ И МЕТОДИЧЕСКИЕ ФАКТОРЫ, ВЛИЯЮЩИЕ НА УСПЕШНОЕ ИЗУЧЕНИЕ ИНОСТРАННОГО ЯЗЫКА

PSYCHOLOGICAL AND METHODOLOGICAL FACTORS AFFECTING THE SUCCESSFUL STUDY OF A FOREIGN LANGUAGE

Филиал «Протвино» государственного университета «Дубна»
Секция «Социальные и гуманитарные науки»

Авторы: Макаров Василий Владимиррович, студент 1 курса направления «Информатика и вычислительная техника» филиала «Протвино» государственного университета «Дубна».

Научный руководитель: Ерицян Сусанна Михайловна, старший преподаватель кафедры общеобразовательных дисциплин филиала «Протвино» государственного университета «Дубна».