**Практическая работа №3**

**Оглавление**

[Практическая работа № 3. Классы 3](#_Toc161929610)

[Содержание пояснительной записки 12](#_Toc161929611)

[Используемое программное обеспечение 13](#_Toc161929612)

[Список литературы 14](#_Toc161929613)

Практическая работа № 3.  
Классы

**Цель практической работы**

Закрепление теоретических знаний по созданию классов.

**Постановка задачи**

**Объе́ктно-ориенти́рованное программирование (сокр. ООП)** – методология программирования, основанная на представлении программы в виде совокупности взаимодействующих **объектов**, каждый из которых является экземпляром определённого **класса**, а классы образуют **иерархию наследования**.

Идеологически ООП – подход к программированию как к моделированию информационных **объектов**, решающий на новом уровне основную задачу структурного программирования: структурирование информации с точки зрения управляемости, что существенно улучшает управляемость самим процессом моделирования, что, в свою очередь, особенно важно при реализации крупных проектов.

Описанием объекта является **класс**, а объект представляет экземпляр этого класса. Можно еще провести следующую аналогию. У нас у всех есть некоторое представление о человеке, у которого есть имя, возраст, какие-то другие характеристики. То есть некоторый шаблон - этот шаблон можно назвать классом. Конкретное воплощение этого шаблона может отличаться, например, одни люди имеют одно имя, другие - другое имя. И реально существующий человек (фактически экземпляр данного класса) будет представлять объект этого класса.

class название\_класса

{

// содержимое класса

}

class Person

{

}

**Создание объекта класса**

После определения класса мы можем создавать его объекты. Для создания объекта применяются **конструкторы**. По сути конструкторы представляют специальные методы, которые называются так же как и класс, и которые вызываются при создании нового объекта класса и выполняют инициализацию объекта.

Person tom = new Person(); // создание объекта класса Person

class Person // определение класса Person

{

}

Person tom = new Person(); // создание объекта класса Person

class Person // определение класса Person

{

public Person() //Конструктор

{

//Если конструктор пустой, то создавать его не обязательно

}

}

В строке Person tom = new Person(); Круглые скобки означают вызов конструктора.

Заметим, что для создания объекта всегда используется специальное слово new.

Чтобы конструктор был доступен вне класса Person используется модификатор public.

**Поля и методы класса**

Класс может хранить некоторые данные. Для хранения данных в классе применяются **поля**. По сути **поля класса** - это переменные, определенные на уровне класса.

Кроме того, класс может определять некоторое поведение или выполняемые действия. Для определения поведения в классе применяются методы.

class Person

{

public string name = "Undefined"; // имя

public int age; // возраст

public void Print()

{

Console.WriteLine($"Имя: {name} Возраст: {age}");

}

}

В отличие от переменных, определенных в методах, поля класса могут иметь **модификаторы**, которые указываются перед полем.

**Обращение к функциональности класса**

Для обращения к функциональности класса - полям, методам (а также другим элементам класса) применяется точечная нотация точки - после объекта класса ставится точка, а затем элемент класса:

Person tom = new Person(); // создание объекта класса Person

string personName = tom.name; // Получаем значение полей в переменные

int personAge = tom.age;

Console.WriteLine($"Имя: {personName} Возраст {personAge}");

// Имя: Undefined Возраст: 0

tom.name = "Tom"; // устанавливаем новые значения полей

tom.age = 37;

// обращаемся к методу Print

tom.Print(); // Имя: Tom Возраст: 37

class Person **//Конструкторов может быть несколько**

{

public string name;

public int age;

public Person()

{

name = "Неизвестно"; age = 18;

}

public Person(string n)

{

name = n; age = 18;

}

public Person(string n, int a)

{

name = n; age = a;

}

public void Print() =>

Console.WriteLine($"Имя: {name} Возраст: {age}");

}

**Цепочка вызова конструкторов**

В примере выше определены три конструктора. Все три конструктора выполняют однотипные действия - устанавливают значения полей name и age. Но этих повторяющихся действий могло быть больше. И мы можем не дублировать функциональность конструкторов, а просто обращаться из одного конструктора к другому также через ключевое слово **this**, передавая нужные значения для параметров:

class Person

{

public string name;

public int age;

public Person() : this("Неизвестно")

{

}

public Person(string name) : this(name, 18)

{

}

public Person(string name, int age)

{

this.name = name;

this.age = age;

}

public void Print() =>

Console.WriteLine($"Имя: {name} Возраст: {age}");

}

**Инициализаторы объектов**

Для инициализации объектов классов можно применять **инициализаторы**. Инициализаторы представляют передачу в фигурных скобках значений доступным полям и свойствам объекта:

Person tom = new Person { name = "Tom", age = 31 };

// или так

// Person tom = new() { name = "Tom", age = 31 };

tom.Print(); // Имя: Tom Возраст: 31

С помощью инициализатора мы можем установить значения **только доступных из вне** класса полей и свойств объекта. Например, в примере выше поля **name** и **age** имеют модификатор доступа public, поэтому они доступны из любой части программы.

Инициализатор выполняется после конструктора, поэтому если и в конструкторе, и в инициализаторе устанавливаются значения одних и тех же полей и свойств, то значения, устанавливаемые в конструкторе, заменяются значениями из инициализатора.

Person tom = new Person { name = "Tom", company = { title = "Microsoft" } };

tom.Print(); // Имя: Tom Компания: Microsoft

class Person

{

public string name;

public Company company;

public Person()

{

name = "Undefined";

company = new Company();

}

public void Print()

=> Console.WriteLine($"Имя: {name} Компания: {company.title}");

}

**Задание на практическую работу**

1. В соответствии с вариантом разработать консольное приложение для расчета угла между векторами.
2. Предусмотреть задание произвольных координат векторов.
3. Защита работы включает демонстрацию работы программы и сравнение ответа с письменным решением студента.

**Варианты заданий на практическую работу**

1. Создайте класс "Person" с полями для имени и возраста. Добавьте метод "PrintInfo()", который будет выводить информацию о человеке.
2. Напишите класс "Animal" с полями для названия и вида. Добавьте метод "MakeSound()", который будет воспроизводить звук, характерный для данного животного.
3. Реализуйте класс "Rectangle" с полями для ширины и высоты. Добавьте метод "CalculateArea()", который будет вычислять площадь прямоугольника.
4. Создайте класс "Student" с полями для имени, возраста и среднего балла. Добавьте метод "GetGrade()", который будет возвращать оценку студента на основе среднего балла.
5. Напишите класс "Car" с полями для марки и цвета. Добавьте метод "StartEngine()", который будет запускать двигатель автомобиля.
6. Реализуйте класс "Circle" с полем для радиуса. Добавьте метод "CalculateCircumference()", который будет вычислять длину окружности.
7. Создайте класс "Book" с полями для названия и автора. Добавьте метод "DisplayInfo()", который будет выводить информацию о книге.
8. Напишите класс "Employee" с полями для имени и должности. Добавьте метод "Promote()", который будет повышать сотрудника на более высокую должность.
9. Реализуйте класс "Triangle" с полями для длин сторон. Добавьте метод "CheckValidity()", который будет проверять, является ли треугольник допустимым.
10. Создайте класс "Product" с полями для названия и цены. Добавьте метод "ApplyDiscount()", который будет применять скидку к товару.
11. Создайте класс "House" с полями для адреса и количества комнат. Добавьте метод "PrintAddress()", который будет выводить адрес дома.
12. Напишите класс "Dog" с полями для клички и породы. Добавьте метод "Bark()", который будет заставлять собаку лаять.
13. Реализуйте класс "Square" с полем для длины стороны. Добавьте метод "CalculatePerimeter()", который будет вычислять периметр квадрата.
14. Создайте класс "Teacher" с полями для имени и предмета. Добавьте метод "Teach()", который будет учить студентов.
15. Напишите класс "Computer" с полями для модели и процессора. Добавьте метод "TurnOn()", который будет включать компьютер.
16. Реализуйте класс "SquareRoot" с полем для числа. Добавьте метод "CalculateSquareRoot()", который будет вычислять квадратный корень числа.
17. Создайте класс "Movie" с полями для названия и жанра. Добавьте метод "Play()", который будет воспроизводить фильм.
18. Напишите класс "Manager" с полями для имени и подчиненных. Добавьте метод "OrganizeMeeting()", который будет организовывать совещание.
19. Реализуйте класс "Cube" с полем для длины ребра. Добавьте метод "CalculateVolume()", который будет вычислять объем куба.
20. Создайте класс "MobilePhone" с полями для модели и операционной системы. Добавьте метод "Call()", который будет осуществлять звонок.
21. Напишите класс "BankBranch" с полями для названия и адреса. Добавьте метод "OpenAccount()", который будет открывать новый банковский счет.
22. Реализуйте класс "Email" с полями для отправителя, получателя и текста сообщения. Добавьте метод "Send()", который будет отправлять электронное письмо.
23. Создайте класс "Fruit" с полями для названия и вкуса. Добавьте метод "Eat()", который будет описывать процесс поедания фрукта.
24. Напишите класс "Hospital" с полями для названия и специализации. Добавьте метод "TreatPatient()", который будет лечить пациента.
25. Реализуйте класс "SquareMatrix" с полем для размерности. Добавьте метод "Transpose()", который будет транспонировать матрицу.
26. Создайте класс "Airplane" с полями для модели и авиакомпании. Добавьте метод "TakeOff()", который будет осуществлять взлет самолета.
27. Напишите класс "Restaurant" с полями для названия и типа кухни. Добавьте метод "ServeCustomer()", который будет обслуживать клиента.
28. Реализуйте класс "TemperatureConverter" с полем для температуры. Добавьте метод "ConvertToFahrenheit()", который будет конвертировать температуру в градусы Фаренгейта.
29. Создайте класс "Train" с полями для номера поезда и маршрута. Добавьте метод "Departure()", который будет отображать информацию о отправлении поезда.
30. Напишите класс "Game" с полями для названия и жанра. Добавьте метод "PlayGame()", который будет запускать игру.

# Содержание пояснительной записки

1. Постановка задачи. Приводится теоретический материал, использованный при написании приложения.

2. Формулировка задания и вариант. Приводится задание на лабораторную работу и вариант этого задания.

3. Описание выполняемых действий. Необходимо привести описание последовательности разработки программы, реализации используемых методов, алгоритмов, блок-схем.

4. Анализ результатов. Привести анализ входных и выходных данных. Показать результаты выполнения программного кода. Предоставить скриншоты обработки тестовых примеров. Сделать выводы.

5. Листинг программы. Привести листинг разработанного программного кода, содержание файлов входных и выходных данных.

# Используемое программное обеспечение

1. Среда программирования MS Visual Studio Community 2022 (Свободно распространяемое программное обеспечение (в учебных целях));
2. Microsoft Office Standard 2007 (Open License: 42267924);
3. Open Office (Свободно распространяемое программное обеспечение).
4. Браузер (Свободно распространяемое программное обеспечение).

# Список литературы

* + - 1. Мейер Б. Объектно-ориентированное программирование и программная инженерия [Электронный ресурс]/ Мейер Б. – Электрон. текстовые данные. – М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. – 285 c.
      2. Биллиг, В. A. Основы объектного программирования на С# (C# 3.0, Visual Studio 2008) [Электронный ресурс] : учебное пособие / В. A. Биллиг. — Электрон. текстовые данные. — Москва, Саратов : Интернет-Университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017. — 583 c. — 978-5-4487-0145-0. — Режим доступа: http://www.iprbookshop.ru/72339.html
      3. Павловская, Т. А. Программирование на языке высокого уровня C# [Электронный ресурс] / Т. А. Павловская. — 2-е изд. — Электрон. текстовые данные. — М. : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 245 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/73713.html
      4. Агапов, В. П. Основы программирования на языке С# [Электронный ресурс] : учебное пособие / В. П. Агапов. — Электрон. текстовые данные. — М. : Московский государственный строительный университет, ЭБС АСВ, 2012. — 128 c. — 978-5-7264-0576-6. — Режим доступа: http://www.iprbookshop.ru/16366.html
      5. Медведев, М. А. Программирование на СИ# [Электронный ресурс] : учебное пособие / М. А. Медведев, А. Н. Медведев ; под ред. А. В. Присяжный. — Электрон. текстовые данные. — Екатеринбург : Уральский федеральный университет, ЭБС АСВ, 2015. — 64 c. — 978-5-7996-1561-1. — Режим доступа: http://www.iprbookshop.ru/69667.html
      6. Казанский А.А. Объектно-ориентированное программирование на языке Microsoft Visual С# в среде разработки Microsoft Visual Studio 2008 и .NET Framework. 4.3 [Электронный ресурс]: учебное пособие и практикум/ Казанский А.А.— Электрон. текстовые данные.— М.: Московский государственный строительный университет, ЭБС АСВ, 2011.— 180 c
      7. Уйманова Н.А. Основы объектно-ориентированного программирования [Электронный ресурс]: практикум/ Уйманова Н.А., Таспаева М.Г.— Электрон. текстовые данные.— Оренбург: Оренбургский государственный университет, ЭБС АСВ, 2017.— 156 c.— Режим доступа: http://www.iprbookshop.ru/78808.html.— ЭБС «IPRbooks»
      8. Новиков П.В. Объектно-ориентированное программирование [Электронный ресурс]: учебно-методическое пособие к лабораторным работам/ Новиков П.В.— Электрон. текстовые данные.— Саратов: Вузовское образование, 2017.— 124 c.— Режим доступа: http://www.iprbookshop.ru/64650.html.— ЭБС «IPRbooks»