**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 №2**

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

Приобретение практических навыков по созданию классов.

# порядок выполнения работы

1. Изучить гл. 6 «Классы и объекты языка C#» учебного пособия В. В. Вдовенко «Разработка приложений на языке C#» (Красноярск, 2010) (см. с. 115–150).

2. Изучить возможности языка UMLдля представления классов.

3. Разработать класс в соответствии с вариантом практического задания и создать консольное приложение.

4. Разработать UML-диаграмму классов для созданного приложения.

5. Ответить на контрольные вопросы.

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 создания классов;
2. выполнить практическое задание по разработке собственных классов в программах на языке С*#*;

Вариант №21: описать класс Комната, содержащий сведения о метраже, высоте потолков и количестве окон. Предусмотреть инициализацию   
с проверкой допустимости значений полей. Описать методы вычисления площади и объема комнаты и свойства для получения состояния объекта.

1. изучить возможности унифицированного языка моделирования Unified Modeling Language (UML) для представления классов;
2. разработать UML-диаграмму классов.

# ХОД РАБОТЫ

1. изучил средства языка С*#* для создания классов;
2. выполнил практическое задание по разработке собственных классов в программах на языке С*#*;

Вариант №21: описал класс Комната, содержащий сведения о метраже, высоте потолков и количестве окон. Предусмотрел инициализацию   
с проверкой допустимости значений полей. Описал методы вычисления площади и объема комнаты и свойства для получения состояния объекта.

Листинг программы:

using System;

namespace Lab2 // 21. Описать класс Комната, содержащий сведения о метраже, высоте потолков и количестве окон. Предусмотреть инициализацию

//с проверкой допустимости значений полей. Описать методы вычисления площади и объема комнаты и свойства для получения состояния объекта.

{

class Room

{

double length;

double width;

double high;

int countWindow;

public Room()

{

length = 0.0;

width = 0.0;

high = 0.0;

countWindow = 0;

}

public Room(double length, double width, double high, int countWindow)

{

this.length = length;

this.width = width;

this.high = high;

this.countWindow = countWindow;

}

public void print()

{

Console.WriteLine($"\nДлина комнаты: {length}, ширина: {width}, высота: {high}, количество окон: {countWindow}.\n");

}

public void Square()

{

Console.WriteLine($"Площадь комнаты: {length\*width}.\n");

}

public void Volume()

{

Console.WriteLine($"Объем комнаты: {length\*width\*high}.\n");

}

public double Length

{

set

{

if (value < 1)

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

else

length = value;

}

get { return length; }

}

public double Width

{

set

{

if (value < 1)

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

else

width = value;

}

get { return width; }

}

public double High

{

set

{

if (value < 1)

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

else

high = value;

}

get { return high; }

}

public int CountWindow

{

set

{

if (value < 1)

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

else

countWindow = value;

}

get { return countWindow; }

}

static void Main(string[] args)

{

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

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

int option = 0;

while (option != 10)

{

Console.Write("\n1. Старт;\n10. ВЫХОД.\nВвод: ");

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

switch (option)

{

case 1:

{

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

int RoomsCount = Convert.ToInt32(Console.ReadLine());

var A = new Room[RoomsCount];

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

{

string l, w, h, c;

do

{

Console.Write("\nВведите длину команты: ");

l = Console.ReadLine()!;

} while (Double.TryParse(l, out double ll) != true);

double length = Convert.ToDouble(l);

do

{

Console.Write("Введите ширину команты: ");

w = Console.ReadLine()!;

} while (Double.TryParse(w, out double ww) != true);

double width = Convert.ToDouble(w);

do

{

Console.Write("Введите высоту комнаты: ");

h = Console.ReadLine()!;

} while (Double.TryParse(h, out double hh) != true);

double high = Convert.ToDouble(h);

do

{

Console.Write("Введите количество окон в комнате: ");

c = Console.ReadLine()!;

} while (Int32.TryParse(c, out int cc) != true);

int countWindow = Convert.ToInt32(c);

A[i] = new Room(length, width, high, countWindow);

A[i].print();

A[i].Square();

A[i].Volume();

Console.WriteLine($"Количество окон: {A[i].CountWindow}");

}

}

break;

case 10:

{

//exit

break;

}

default:

{

Console.WriteLine("\nВведено неверное число.\nВведите число 1 или 10.");

break;

}

}

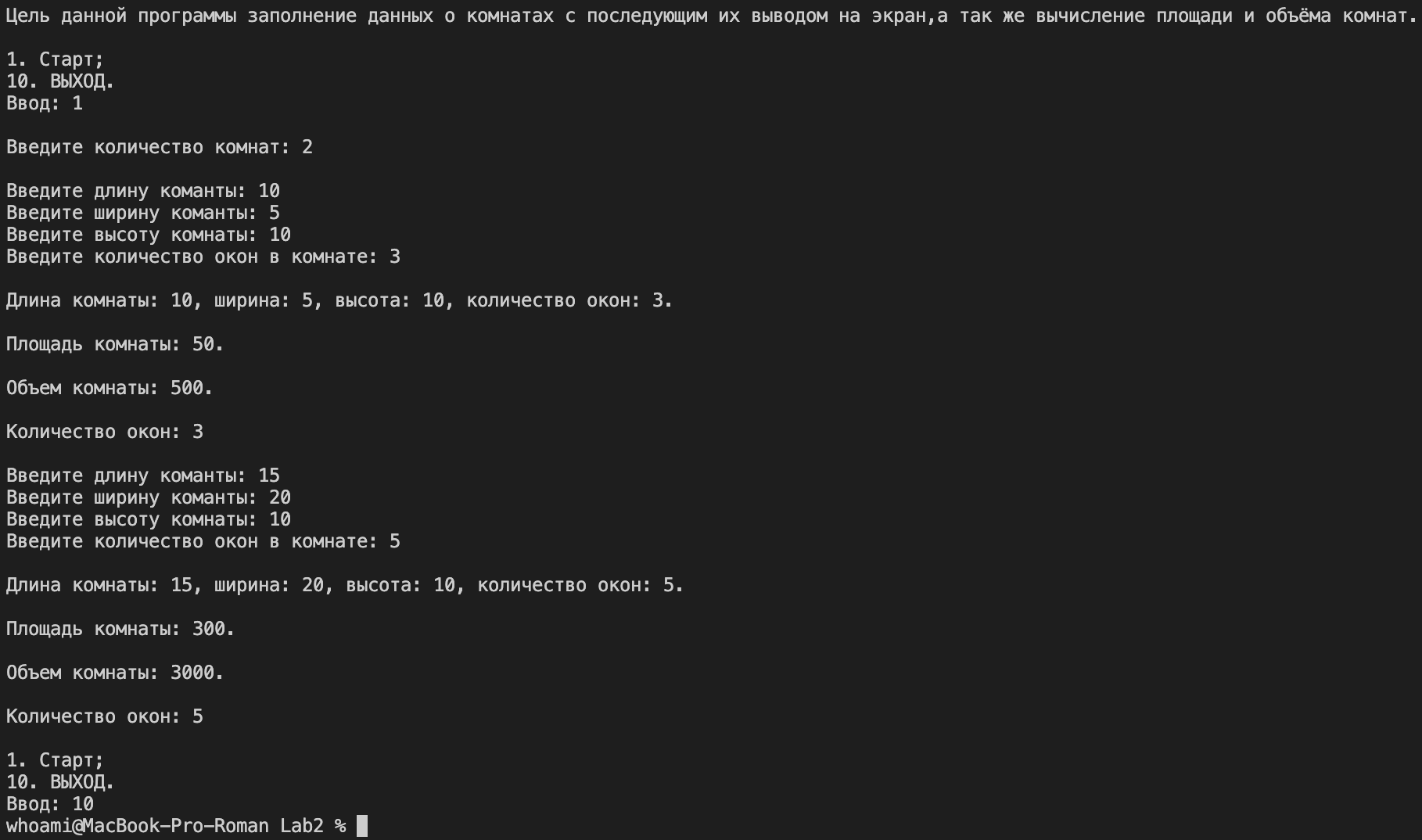
}

}

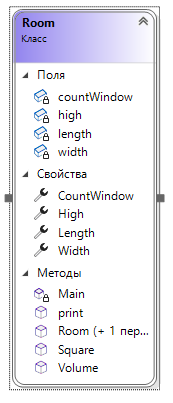
}

}

Результаты работы:



1. изучил возможности унифицированного языка моделирования Unified Modeling Language (UML) для представления классов;
2. разработал UML-диаграмму классов.



**ОТВЕТЫ НА КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. Дайте определение класса.

Класс − это пользовательский тип данных.

Класс − описание структуры объекта и методов работы с ним.

1. Охарактеризуйте объект. Приведите примеры объектов.

Объект – структура данных, содержащая описание свойств внешнего объекта программирования. Пример: комната (длина, ширина, высота, кол-во окон).

1. Какие роли играет класс в объектно-ориентированной программе?

У класса две различные роли: модуля и типа данных. Класс − это модуль, архитектурная единица построения программной системы. Модульность построения - основное свойство программных систем. В ООП программная система, строящаяся по модульному принципу, состоит из классов, являющихся основным видом модуля. Модуль может не представлять собой содержательную единицу - его размер и содержание определяется архитектурными соображениями, а не семантическими. Ничто не мешает построить монолитную систему, состоящую из одного модуля - она может решать ту же задачу, что и система, состоящая из многих модулей.

Вторая роль класса не менее важна. Класс − это тип данных, задающий реализацию некоторой абстракции данных, характерной для задачи, в интересах которой создается программная система. С этих позиций классы - не просто кирпичики, из которых строится система. Каждый кирпичик теперь имеет важную содержательную начинку. Представьте себе современный дом, построенный из кирпичей, и дом будущего, где каждый кирпич выполняет определенную функцию: один следит за температурой, другой - за составом воздуха в доме. ОО-программная система напоминает дом будущего.

1. Каким образом соотносятся понятия «класс» и «объект»?

Класс **–**тип данных, определяемый программистом, объект **–** переменная класса.

1. Перечислите принципы объектно-ориентированного программирования.

* Абстракция — отделение концепции от ее экземпляра;
* Полиморфизм — реализация задач одной и той же идеи разными способами;
* Наследование — способность объекта или класса базироваться на другом объекте или классе. Это главный механизм для повторного использования кода. Наследственное отношение классов четко определяет их иерархию;
* Инкапсуляция — размещение одного объекта или класса внутри другого для разграничения доступа к ним.

1. Дайте определение объектно-ориентированного программирования.

Объектно-ориентированное программирование − методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования.

1. В чем состоят преимущества объектно-ориентированного программирования?

* Классы позволяют проводить конструирование из полезных компонент, обладающих простыми инструментами, что дает возможность абстрагироваться от деталей реализации.
* Данные и операции вместе образуют определенную сущность, и они не «размазываются» по всей программе, как это нередко бывает в случае процедурного программирования.
* Локализация кода и данных улучшает наглядность и удобство сопровождения программного обеспечения.
* Инкапсуляция информации защищает наиболее критичные данные от несанкционированного доступа.
* Полиморфизм.

1. Дайте определение метода класса.

Метод – функция, работающая с объектом.

1. В чем состоит принцип инкапсуляции? За счет чего обеспечивается инкапсуляция?

Принцип инкапсуляции состоит в сокрытии определенных переменных или методов. Одни переменные и методы доступны для чтения, изменения и вызова, а другие служат исключительно нуждам самого объекта и недоступны извне, они называются приватными.

1. В чем заключается полиморфизм объектов?

В возможности объектов с одинаковой спецификацией иметь различную реализацию.

1. Какие члены класса доступны другим классам?

Если модификатор private, то никакие, а если модификатор public − все.

1. Какие варианты доступа к членам класса могут быть реализованы в программе на языке C#?

Когда член класса обозначается спецификатором public, он становится доступным из любого другого кода в программе, включая и методы, определенные в других классах. Когда же член класса обозначается спецификатором private, он может быть доступен только другим членам этого класса. С помощью модификатора доступа protected обеспечивается создание защищенного члена класса, доступ к которому открыт в пределах иерархии классов. А модификатор internal служит в основном для сборок.

1. Дайте определение термина «конструктор». Для чего предназначен конструктор?

Конструктор − специальная компонентная функция, автоматически вызываемая при создании объекта класса для его инициализации. Предназначен для инициализации объекта класса.

1. Приведите синтаксис конструктора класса в общем виде. Проиллюстрируйте его фрагментом программы на языке C#.

Название\_класса() {}

using System;

namespace Lab2

{

class Room

{

double length;

double width;

double high;

int countWindow;

public Room()

{

length = 0.0;

width = 0.0;

high = 0.0;

countWindow = 0;

}

}

static void Main(string[] args)

{

var A = new Room();

}

1. Сколько конструкторов может содержать класс языка C#?

Класс может иметь любое число конструкторов, которые должны отличаться один от другого числом параметров и/или типами параметров.

1. Какие умолчания для конструкторов приняты в языке C#?   
   В каком случае конструктор по умолчанию не используется?

По умолчанию в C# предусмотрено наличие конструктора, который присваивает нулевые значения всем переменным экземпляра и значения null. Если конструктор явно определёнен в классе, то конструктор по умолчанию не используется.

1. Приведите синтаксис описания класса в общем виде. Проиллюстрируйте его фрагментом программы на языке C#.

class имя\_класса {

// Объявление переменных экземпляра.

доступ тип переменная1; //...

// Объявление методов.

доступ возращаемый\_тип метод1 (параметры)

{ // тело метода }

class Room

{

double length;

double width;

double high;

int countWindow;

public Room()

{

length = 0.0;

width = 0.0;

high = 0.0;

countWindow = 0;

}

public Room(double length, double width, double high, int countWindow)

{

this.length = length;

this.width = width;

this.high = high;

this.countWindow = countWindow;

}

public void print()

{

Console.WriteLine($"\nДлина комнаты: {length}, ширина: {width}, высота: {high}, количество окон: {countWindow}.\n");

}

public void Square()

{

Console.WriteLine($"Площадь комнаты: {length\*width}.\n");

}

public void Volume()

{

Console.WriteLine($"Объем комнаты: {length\*width\*high}.\n");

}

public double getLength()

{

return length;

}

public double getWidth()

{

return width;

}

public double getHigh()

{

return high;

}

public int getCW()

{

return countWindow;

}

}

1. Какие модификаторы типа доступа вам известны?

* private: закрытый или приватный компонент класса или структуры. Приватный компонент доступен только в рамках своего класса или структуры.
* privateprotected: компонент класса доступен из любого места в своем классе или в производных классах, которые определены в той же сборке.
* protected: такой компонент класса доступен из любого места в своем классе или в производных классах. При этом производные классы могут располагаться в других сборках.
* internal: компоненты класса или структуры доступен из любого места кода в той же сборке, однако он недоступен для других программ и сборок.
* protectedinternal: совмещает функционал двух модификаторов protected и internal. Такой компонент класса доступен из любого места в текущей сборке и из производных классов, которые могут располагаться в других сборках.
* public: публичный, общедоступный компонент класса или структуры. Такой компонент доступен из любого места в коде, а также из других программ и сборок.

1. Какие особенности имеет использование полей класса,   
   в объявлении которых использовано служебное слово static?

Модификатор static используется для объявления статического члена, принадлежащего собственно типу, а не конкретному объекту. Модификатор static можно использовать для объявления классов static. В классах, интерфейсах и структурах вы можете добавить модификатор static к полям, методам, свойствам, операторам, событиям и конструкторам. Вызвать можно только через имя класса.

1. Приведите синтаксис описания метода класса в общем виде.

доступ тип имя()

{

Тело\_метода;

}

1. Каким образом можно вернуть параметры из метода?

С помощью return, кортежа, структуры, класса или модификатором out.

1. Приведите синтаксис создания объекта в общем виде. Проиллюстрируйте его фрагментом программы на языке C#.

имя\_класса имя\_обьекта = new имя\_класса();

var A = new Room();

1. Какой скрытый параметр автоматически передается в любой нестатический метод класса?

Для этого в любой нестатический метод автоматически передается скрытый параметр this, в котором хранится ссылка на вызвавший функцию экземпляр.

1. Могут ли методы класса быть закрытыми?

Да, могут быть.

1. Какое значение инициализирует поля ссылочного типа по умолчанию?

Null.

1. Каким образом в определении класса реализуется такой принцип ООП, как инкапсуляция?

Инкапсуляция − свойство языка программирования, позволяющее пользователю не задумываться о сложности реализации используемого программного компонента (то, что у него внутри), взаимодействовать с ним посредством предоставляемого интерфейса, а также объединить и защитить жизненно важные для компонента данные. При этом пользователю предоставляется только интерфейс — спецификация объекта.

Private или protected.

Пользователь не может использовать закрытые данные и методы.

1. Приведите синтаксис метода-свойства класса.

доступ тип Имя

{

set

{

тело

}

get { тело }

}

1. Объясните, что представляет собой индексатор. В каких случаях при программировании следует использовать индексаторы?

Индексаторы позволяют индексировать объекты и обращаться к данным по индексу. Фактически с помощью индексаторов мы можем работать с объектами как с массивами. По форме они напоминают свойства со стандартными блоками get и set, которые возвращают и присваивают значение.

1. Какой класс называют вложенным? В каком случае использование вложенного класса может иметь смысл?

Вложенные классы определяются внутри области определения другого класса. Вложенные классы обладают некоторыми специальными возможностями, которые удобны, когда нужен вспомогательный класс, работающий внутри содержащего его класса.

Например, контейнерный класс может содержать коллекцию объектов. Предположим, что требуется некоторое средство для выполнения итерации по всем содержащимся объектам, чтобы позволить внешним пользователям, выполняющим итерацию, поддерживать маркер, или некую разновидность курсора, который запоминает свое текущее место во время итерации.

1. Назовите элементы **UML**-диаграммы классов.

Диаграммы, сущности, связи.

1. Каким образом можно создать **UML**-диаграмму классов?

С помощью конструктора классов в Visual Studio.

# ВЫВОДЫ

Были приобретены практические навыки по созданию классов.