**Дисциплина «Надежность программного обеспечения»**

**Практическое занятие № 3**

**Тема:** Оценка характеристик программ на основе объектно-ориентированных метрик Мартина.

**Время выполнения практического занятия (аудиторные часы)** –   
4 часа.

**Цель работы:** изучить методику оценки характеристик программ с использованием объектно-ориентированных метрик Мартина.

**Оборудование и программное обеспечение:** работа выполняется на ПЭВМ типа IBM PC с использованием стандартных функций ОС.

# **1. Теоретические сведения**

Класс представляет собой абстракцию, которая может быть размещена на различных уровнях детализации, причем разными способами (например, в виде списка операций, последовательности состояний или последовательности взаимодействий). В связи с этим для объектно-ориентированных метрик более целесообразно представление абстракций в терминологии измерений класса. К примерам такого представления можно отнести количество экземпляров класса в приложении, количество родовых классов в приложении, отношение количества родовых классов к количеству неродовых классов.

В 1995 г. американский консультант по программному обеспечению Роберт Сесил Мартин (Robert Cecil Martin) предложил пять основных принципов дизайна классов в объектно-ориентированном проектировании (так называемая методология ***SOLID***) и разработал комплекс метрик, получивших наименование метрик Мартина.

В программах класс редко может быть повторно использован изолированно от других классов. Почти всегда класс имеет группу классов, с которыми он работает во взаимодействии и от которых его достаточно трудно отделить. Для повторного использования таких классов необходимо заново использовать всю группу классов. Связность такой группы классов (она называется *категорией классов*) достаточно высока, и для ее существования должны быть соблюдены **следующие условия**:

• *классы в пределах категории закрыты от любых попыток изменения отдельных экземпляров.* Это означает, что если одному классу необходимо измениться, то весьма вероятно изменение всех классов в этой категории. Если любой из классов открыт для некоторого вида изменений, они все открыты для этого вида изменений;

• *классы в категории повторно используются только вместе.* Они настолько взаимозависимы, что не могут быть отделены друг от друга. Поэтому если предпринимается попытка повторного использования одного класса в категории, то все другие классы этой категории также повторно используются вместе с таким классом;

• *классы в категории обеспечивают некоторую общую функцию или достигают некоторую общую цель*.

Ответственность, независимость и стабильность категории, по мнению Мартина, могут быть измерены путем подсчета некоторых зависимостей, взаимодействующих с этой категорией.

Мартин предложил следующие метрики для оценки характеристик программы при объектно-ориентированном программировании:

• **(центростремительное сцепление)** - метрика, определяющая количество классов вне конкретной категории, которые зависят от классов внутри её;

• **(центробежное сцепление)** - метрика, оценивающая количество классов внутри конкретной категории, которые зависят от классов вне её;

• ***I* (нестабильность)** - расчетная метрика, определяемая в соответствии с выражением

Данная метрика имеет диапазон значений [0, 1]: значение метрики  
означает **максимально стабильную категорию**, а указывает **максимально нестабильную категорию**.

Дополнительно к указанным метрикам можно определять меру ***А*** (абстрактность), которая позволяет оценить абстрактность категории (если категория абстрактна, то она является достаточно гибкой и может быть легко расширена):

где - количество абстрактных классов в категории; - общее количество классов в категории.

Значения метрики абстрактности расположены в диапазоне [0, 1]: при **нулевом значении** *категория полностью конкретна*, а при **единичном значении** является *полностью абстрактной*.

На основе приведенных метрик Мартина можно построить график, отражающий зависимость между абстрактностью и нестабильностью. Если построить прямую, задаваемую формулой ***I + А = 1***, то категории, расположенные на этой прямой, будут иметь наилучшую сбалансированность между абстрактностью и нестабильностью. Эта прямая называется **главной последовательностью**.

Получив главную последовательность, можно ввести еще две метрики:

• расстояние до главной последовательности:

• нормализированное расстояние до главной последовательности:

Практически для любых категорий классов справедливым является следующее утверждение: ***чем ближе они находятся к главной последовательности, тем лучше для обеспечения качества, и как следствие надежности, программного средства.***

## Практическая реализация оценки характеристик программы на основе объектно-ориентированных метрик Мартина

### Задача «Платеж за электроэнергию»

**Задача:** Необходимо определить класс, описывающий платеж за электроэнергию. В рамках класса следует предусмотреть следующие поля:

• фамилия плательщика;

• потребление электроэнергии за оплачиваемый месяц;

• нормативное среднемесячное потребление;

• тариф (стоимость одного киловатт-часа).

*Рекомендуется применять следующие методы:*

• вычисление суммы оплаты;

• формирование сводной информации по одному платежу (вид платежа, фамилия, сумма, потребление).

Платеж может выполняться по показаниям счетчика или по нормативно установленному среднемесячному потреблению. В процессе эксплуатации программы тариф и нормативно установленное среднемесячное потребление не изменяются. Для создания конкретного платежа предусмотреть соответствующий конструктор.

Все платежи должны сохраняться в архиве. Запросы по ведению архива выполняются статическими методами класса «Запрос»:

• занесение платежей в архив. Данные платежа вводятся с клавиатуры. Ввод отрицательного показания счетчика означает оплату по нормативно установленному среднемесячному потреблению;

• вывод сводной информации из архива.

Архив моделируется массивом объектов.

В основном классе «Платежи» следует сформировать архив платежей. По данным архива предусмотреть выдачу сводной информации о платежах.

**Требуется:** Разработать исходный код программы, а также определить оценки характеристик программы на основе объектно-ориентированных метрик Мартина.

**Реализация программы**

Текст программы для реализации возможного решения поставленной задачи, разработанной с использованием языка программирования С#, приведен на Таблице 3.1.

*Таблица 3.1 Текст программы «Платеж за электроэнергию»*

|  |  |
| --- | --- |
| **Номера**  **строк** | **Строки программы** |
| 1 | using System; |
| 2 | using System.Collections.Generic; |
| 3 | using System.Text; |
| 4  5 | namespace EX1  { |
| 6 | class Электро |
| 7 | { |
| 8 | private static double тариф = 1.84; |
| 9 | private static int потреблениеСреднее = 300; |
| 10 | private string фамилия; |
| 11 | private int потребление; |
| 12 | public Электро(string фамилия) |
| 13 | { |
| 14 | this.фамилия = фамилия; |
| 15 | потребление = потреблениеСреднее; |
| 16 | } |
| 17 | public Электро(string фамилия, int текущее, int предыдущее) |
| 18 | { |
| 19 | this.фамилия = фамилия; |
| 20 | потребление = текущее - предыдущее; |
| 21 | } |
| 22 | public double Сумма() |
| 23 | { |
| 24 | return потребление \* тариф; |
| 25 | } |
| 26 | public string Инфо() |
| 27 | { |
| 28 | return string.Format(" {0,-20} {1 ,-20} {2,10:f2} {3,10:d6}", |
| 29 | "Электроэнергия", фамилия, Сумма(), потребление); |
| 30 | } |
| 31 | } |
| 32 | class Запрос |
| 33 | { |
| 34 | public static void Заполнить(Электро[] платеж) |
| 35 | { |
| 36 | string фам; |
| 37 | int счПред=0, счТек=0; |
| 38 | Console.Clear(); |
| 39 | for (int i = 0; i < плaтcж.Length; i++) |
| 40 | { |
| 41 | Console.Write("Платеж "+ i +": Фамилия -> "); |
| 42 | фам = Console.ReadLine(); , |
| 43 | Console.Write ("Платеж " + i +": Текущее значение счетчика -> "); |
| 44 | счТек = int.Parse(Console.ReadLine()); |
| 45 | if (счТек > 0) |
| 46 | { |
| 47 | Console.Write ("Платеж" + i +": Предыдущее значение счетчика ->"); |
| 48 | счПред = int.Parse(Console.ReadLine()); |
| 49 | } |
| 50 | if(cчTeк <= 0) |
| 51 | платеж[i] = new Электро(фам); |
| 52 | else |
| 53 | платеж[i] = new Электро(фам, счТек, счПред); |
| 54 | } |
| 55 | } |
| 56 | public static void Вывести(Электро[] платеж) |
| 57 | { |
| 58 | for (int i = 0; i < платеж.Length; i++) |
| 59 | Соnsо1е.WriteLine(платеж[i].Инфо()); |
| 60 | } |
| 61 | } |
| 62 | class Платежи |
| 63 | { |
| 64 | static void Main(string[] args) |
| 65 | { |
| 66 | ConsoleKeyInfo rep; |
| 67 | Электро[] плэ; |
| 68 | int кпэ; |
| 69 | do |
| 70 | { |
| 71 | Console.Clear(); |
| 72 | Console.Write("Количество платежей за электроэнергию: "); |
| 73 | кпэ = int.Parse(Console.ReadLine()); |
| 74 | плэ = new Электро[кпэ]; |
| 75 | Запрос.Заполнить(плэ); |
| 76 | Запрос.Вывести(плэ); |
| 77 | Console.WriteLine("Для выхода нажмите ESC"); |
| 78 | rep = Console.ReadKey(true); |
| 79 | }while(rep.Key!= ConsoleKey.Escape); |
| 80 | } |
| 81 | } |

**Оценка характеристик программы**

Проанализируем текст программы для оценки ее качества с помощью метрик Мартина, которые позволяют оценить меру сложности объектно-ориентированной программы на основе анализа детализации организационной структуры классов программы и их связности, так как все классы работают во взаимодействии.

Исходный код программы включает три класса, которые можно отнести к одной категории, предназначенной для решения задачи, целью которой является расчет сумм платежей за электроэнергию.

К ним относятся следующие классы (см. табл. 3.1):

• ***class Электро*** (строка 6) - класс, определяющий экземпляр объекта платежа за электроэнергию;

• ***class Запрос*** (строка 32) - класс, содержащий методы обработки информации об объектах типа *Электро*;

• ***class Платежи*** (строка 62) - класс, в котором содержится основной метод обработки информации массивов объектов типа *Электро* с помощью методов класса *Запрос*.

Кроме указанных классов в программе используются обращения к библиотеке встроенных классов среды программирования, которые не входят в рассматриваемую категорию классов. Необходимо отметить, что классы, не входящие в рассматриваемую категорию, никак не зависят от ее классов. Следовательно, центростремительное сцепление классов по теории Мартина   
***Са = 0***. *Нулевое значение метрики определяет полную независимость классов других категорий, не связанных с расчетом платежей за электроэнергию.*

В классе *Электро* используется метод класса *string* (строка 25), в классах *Запрос* и *Платежи* используются методы класса *Console* (строки 35, 38, 39, 40, 42, 68, 69, 70). Таким образом, все три класса рассматриваемой категории имеют зависимость от встроенных классов, которые не являются ее составными частями. Поэтому центробежное сцепление классов по теории Мартина ***Се = 3.***

Расчетная метрика нестабильности ***I*** определяется следующим образом:

Исходя из полученного значения (***I = 1***) следует, что категория классов анализируемой программы является максимально нестабильной, так как зависимость от классов других категорий носит преобладающий характер.

Определим меру абстрактности ***А*** рассматриваемой категории классов. Код программы не содержит ни одного абстрактного класса, следовательно, ***nA = 0***. Общее количество классов в рассматриваемой категории составляет 3, следовательно, ***nAll = 3***.

Исходя из значения метрики ***А=0*** можно сделать вывод, что рассматриваемая категория классов является полностью конкретной.

Расстояние до главной последовательности определяется уравнением   
***I+А=1***. В соответствии с теорией Мартина расстояние до главной последовательности определятся формулой

Категория расположена на прямой главной последовательности, что свидетельствует о наилучшей сбалансированности между абстрактностью и нестабильностью. Нормализованное расстояние до главной последовательности равно:

**Вывод:** В целом качество разработанной, программы можно считать достаточно высоким, так как рассматриваемая категория классов исходного кода находится в пределах нормализованного расстояния до главной последовательности.

### Задача «Геометрия окружности и прямоугольника»

**Задача:** В предметной области «Геометрия» определены информационные объекты, описывающие понятия «Точка», «Окружность» «Прямоугольник» в виде классов. Классы «Окружность» и «Прямоугольник» являются наследниками класса «Точка». Каждый из классов имеет метод вывода параметров объекта. *Следует иметь в виду, что метод вывода в базовом классе не виртуальный и не абстрактный.* Методы вывода в наследниках переопределены. Вызываемый метод будет определяться типом ссылки.

Базовый класс «Точка» описан следующим образом:

* закрытые поля:
* координата по оси X (вещественная);
* координата по оси Y (вещественная);
* метод: вывод координат точки.

Класс «Окружность» задан следующими элементами:

* закрытые поля:
* координаты центра - унаследованные поля;
* радиус (вещественный);
* метод: вывод координат центра и длины окружности.

Класс «Прямоугольник» задается так:

* закрытые поля:
* координаты левого верхнего угла - унаследованные поля;
* основание (вещественное);
* высота (вещественная);
* метод: вывод координат левого верхнего угла и площади.

**Требуется:** В основном классе необходимо сформировать список геометрических объектов разных типов в виде массива и вывести параметры объектов на экран. Разработать исходный код программы, по которому определить оценки характеристик программы на основе объектно-ориентированных метрик Мартина.

**Реализация программы**

Текст программы на языке C# для реализации возможного алгоритма решения поставленной задачи представлен в табл. 3.2.

*Таблица 3.2 Текст программы «Геометрия окружности и прямоугольника»*

|  |  |
| --- | --- |
| **Номера**  **строк** | **Строки программы** |
| 1 | namespace EX1\_1 |
| 2 | { |
| 3 | class Точка |
| 4 | { |
| 5 | protected double x, у; |
| 6 | public Точка (double x, double y) |
| 7 | { |
| 8 | this.x = x; this.y = y; |
| 9 | } |
| 10 | public void Показать() |
| 11 | { |
| 12 | Console.WriteLine("Toчкa ({0:f2}:{1:f2})", x, y); |
| 13 | } |
| 14 | } |
| 15 | class Окружность: Точка |
| 16 | { |
| 17 | private double r; |
| 18 | public Окружность(double x, double y, double r): base(x, y) |
| 19 | { |
| 20 | this.r = r; |
| 21 | } |
| 22 | public new void Показать() |
| 23 | { |
| 24 | Console. WriteLine("Центр({0:f2}: {1 :f2}) Длина={2:f2}",x,y,2\*Math.PI\*r); |
| 25 | } |
| 26 | } |
| 27 | class Прямоугольник: Точка |
| 28 | { |
| 29 | private double a, h; |
| 30 | public Прямоугольник(double х, double у, double a, double h): base(x, у) |
| 31 | { |
| 32 | this.a = a; this.h = h; |
| 33 | } . |
| 34 | public new void Показать() |
| 35 | { |
| 36 | Console.WriteLine("Угол({0:f2}:{l:f2}) Площадь={2:f2}",х, у, a\*h); |
| 37 | } |
| 38 | } |
| 39 | class Program |
| 40 | { |
| 41 | static void Main(string[] args) |
| 42 | { |
| 43 | Точка[] список = new Точка[6]; |
| 44 | список[0] = new Точка(-1.0, -1.0); |
| 45 | список[1] = new Окружность(1.0, 1.0, 1.0); |
| 46 | список[2] = new Прямоугольник(10.0, 10.0, 10.0, 10.0); |
| 47 | список[3] = new Точка(-2.0, -2.0); |
| 48 | список[4] = new Окружность(3.0, 3.0, 3.0); |
| 49 | список[5] = new Прямоугольник(20.0, 20.0, 0.5, 0.5); |
| 50 | for (int i = 0; i < список.Length; i++) список[i].Показать(); |
| 51 | Console. WriteLine(); |
| 52 | for (int i = 0; i < список.Length; i++) |
| 53 | { |
| 54 | if(список[i] is Точка) |
| 55 | ((Точка)список[i]).Показать(); |
| 56 | else |
| 57 | if (список[i] is Окружность) |
| 58 | ((Окружность)список[i]).Показать(); |
| 59 | else |
| 60 | if (список[i] is Прямоугольник) |
| 61 | ((Прямоугольник)список[i]).Показать(); |
| 62 | } |
| 63 | Console.WriteLine(); |
| 64 | for (int i = 0; i < список.Length; i++) |
| 65 | switch (список[i].GetType().Name) |
| 66 | { |
| 67 | case "Точка":((Точка)список[i]).Показать(); break; |
| 68 | case "Окружность": ((Окружность)список[i]).Показать(); break; |
| 69 | case "Прямоугольник": ((Прямоугольник)список[i]).Показать(); break; |
| 70 | } |
| 71 | } |
| 72 | } |
| 73 | } |

**Оценка характеристик программы**

Проанализируем текст программы для оценки ее качества с помощью метрик Мартина, которые позволяют оценить меру сложности объектно-ориентированной программы на основе анализа организационной структуры классов программы и их связности, так как все классы работают во взаимодействии.

Исходный код программы включает четыре класса, которые можно отнести к одной категории, предназначенной для решения задачи, целью которой является определение геометрических фигур на основании исходных координат точек, определяющих местоположение этих фигур. К ним относятся следующие классы (см. табл. 3.2):

* class Точка - базовый класс, определяющий понятие точка, расположенная на координатной плоскости;
* class Окружность - производный класс от class Точка, определяющий понятие окружность с центром в точке, расположенной на координатной плоскости;
* class Прямоугольник - производный класс от class Точка, определяющий понятие прямоугольник, расположение которого на плоскости определяется координатами точки верхнего левого его угла;
* class Program - класс, в котором содержится основной метод обработки массивов объектов типа Точка, Окружность и Прямоугольник.

Кроме указанных классов в программе используются обращения к библиотеке встроенных классов среды программирования, которые не входят в рассматриваемую категорию классов.

Необходимо отметить, что классы, не входящие в рассматриваемую категорию, никак не зависят от ее классов. Следовательно, центростремительное сцепление классов по теории Мартина **Са=*0***. Нулевое значение метрики определяет полную независимость классов других категорий, не связанных с обработкой информации по определению геометрических фигур.

В классе Точка осуществляется вызов метода из класса Console (строка 12). В классе Окружность осуществляется вызов методов из классов Console и Math (строка 24). В классе Прямоугольник осуществляется вызов метода из класса Console (строка 36). Класс Program содержит обращения к методам класса Console (строки 51, 63). Таким образом, все четыре класса программы имеют связь с классами, не входящими в рассматриваемую категорию, и зависят от них. Следовательно, центробежное сцепление классов по теории Мартина **Се = *4***.

Расчетная метрика нестабильности **I** определяется следующим образом:

Исходя из полученного значения (***I = 1***) следует, что категория классов анализируемой программы является максимально нестабильной, так как зависимость от классов других категорий носит преобладающий характер.

Определим меру абстрактности **А** рассматриваемой категории классов. Код программы не содержит ни одного абстрактного класса, следовательно,   
**nА *= 0***. Общее количество классов в рассматриваемой категории составляет 4, следовательно, ***nAll = 4***.

Исходя из значения метрики **А *= 0*** можно сделать вывод, что рассматриваемая категория классов является полностью конкретной.

Расстояние до главной последовательности определяется уравнением   
***I +* А = *1***. В соответствии с теорией Мартина расстояние до главной последовательности определятся формулой:

Категория расположена на прямой главной последовательности, что свидетельствует о наилучшей сбалансированности между абстрактностью и нестабильностью. Нормализованное расстояние до главной последовательности равно:

**Вывод:** В целом качество разработанной программы можно считать достаточно высоким, так как рассматриваемая категория классов исходного кода находится в пределах нормализованного расстояния до главной последовательности.

# 2. Задание

При оценке характеристик программ на основе объектно-ориентированных метрик Мартина необходимо выполнить следующее:

* 1. разработать программу, реализующую заданный, в соответствии с вариантом, алгоритм (рекомендуется использовать язык программирования С#);
  2. оценить характеристики разработанной программы на основе применения объектно-ориентированных метрик Мартина.

Практическая работа выполняется в соответствии с вариантом (номер варианта соответствует номеру студента в журнале группы).

**Варианты заданий**

**Вариант 1.** Определить понятие «Радиостанция». Состояние объекта определяется следующими полями:

* наименование радиостанции (строка до 60 символов);
* частота вещания (вещественное число).

Наименование радиостанции может иметь несколько слов, разделенных пробелами. Вычислить количество радиостанций, вещающих в заданном диапазоне частот.

**Вариант 2.** Определить понятие «Радиостанция». Состояние объекта определяется следующими полями:

* наименование радиостанции (строка до 60 символов);
* частота вещания (длинное целое число).

Наименование радиостанции может иметь несколько слов, разделенных пробелами. В таблице радиостанций изменить название радиостанции, вещающей на заданной частоте.

**Вариант 3.** Определить понятие «Радиостанция». Состояние объекта определяется следующими полями:

* наименование радиостанции (строка до 60 символов);
* частота вещания (длинное целое число).

Наименование радиостанции может иметь несколько слов, разделенных пробелами. В таблице радиостанций изменить частоту вещания радиостанции с заданным названием.

**Вариант 4.** Определить понятие «Радиостанция». Состояние объекта определяется следующими полями:

* наименование радиостанции (строка до 60 символов);
* частота вещания (вещественное число).

Наименование радиостанции может иметь несколько слов, разделенных пробелами. Вычислить количество радиостанций, имеющих в своем названии сочетание букв «музыка».

**Вариант 5**. Определить понятие «Радиостанция». Состояние объекта определяется следующими полями:

* наименование радиостанции (строка до 60 символов);
* частота вещания (вещественное число).

Наименование радиостанции может иметь несколько слов, разделенных пробелами. Вычислить количество радиостанций, имеющих в своем названии от 2 до 4 слов.

**Вариант 6**. Определить понятие «Радиостанция». Состояние объекта определяется следующими полями:

* наименование радиостанции (строка до 60 символов);
* частота вещания (длинное целое число).

Наименование радиостанции может иметь несколько слов, разделенных пробелами. Составляется заявка на распределение частот вещания. Проверить таблицу радиостанций на корректность. Сформировать и вывести на экран наименования радиостанций, подавших заявки на вещание на совпадающих частотах. При отсутствии конфликтов выдать сообщение «Норма».

**Вариант 7**. Определить понятие «Книга». Состояние объекта определяется следующими полями:

* единый регистрационный номер (длинное целое число);
* автор (строка до 20 символов);
* год издания (целое число);
* количество экземпляров (целое число).

Вычислить количество экземпляров книг заданного автора, выпущенных в период с 2007 по 2016 г.

**Вариант 8**. Определить понятие «Книга». Состояние объекта определяется следующими полями:

* единый регистрационный номер (длинное целое число);
* автор (строка до 20 символов);
* год издания (целое число);
* количество страниц (целое число).

Вычислить средний объем книги в страницах среди книг, выпущенных за заданный период времени.

# 3. Требования к отчету

Отчёт выполняется каждым студентом индивидуально. Работа должна быть оформлена в электронном виде в формате .doc и распечатана на листах формата А4. На титульном листе указываются: наименование учебного учреждения, наименование дисциплины, название и номер работы, вариант, выполнил: фамилия, имя, отчество, группа, проверил: преподаватель ФИО (образец титульного листа представлен в приложении 1).

Отчет должен содержать:

* титульный лист;
* цель работы;
* краткие теоретические сведения, ответы на контрольные вопросы;
* задания в соответствии с вариантом, решение поставленной задачи;
* выводы.

# 4. Контрольные вопросы

1. В чем заключается суть исследования программы при объектно-ориентированном программировании?
2. Какие условия должны быть соблюдены для существования категории классов?
3. Какие метрики используются для оценки характеристик программы при объектно-ориентированном программировании?
4. Что такое «мера абстрактности» в метриках Мартина?
5. Что такое главная последовательность программы в метриках Мартина?

# 5. Литература

1. Керниган Б., Ритчи Д. Язык программирования С: Пер. с англ. М.: Вильямс, 2009.

2. Майерс Г. Надежность программного обеспечения. М.: Мир, 1980.

3. Холстед М.Х. Начала науки о программах. М.: Финансы и статистика, 1981.

4. Черников Б.В. Управление качеством программного обеспечения: учебник. М.: ИД «ФОРУМ»: ИНФРА-М, 2012.

**Приложение 1**



МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет Информатика и вычислительная техника

Кафедра Кибербезопасность информационных систем

**Практическое занятие № \_\_\_\_**

на тему «\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_»

Выполнил обучающийся гр.\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия, Имя, Отчество)

Проверил:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(должность, Фамилия, Имя, Отчество)

Ростов-на-Дону

2018