Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Российская академия народного хозяйства и государственной службы при Президенте Российской Федерации»

Нижегородский институт управления

Кафедра Информатики и информационных технологий

ОТЧЕТ

ПО ЗАДАНИЮ К ЛЕКЦИИ №11

«Тонкости работы с объектами»

Выполнил: студент группы: ИБ-321

Борисков Дмитрий Андреевич

Преподаватель: Окулич Виктор Иванович

Нижний Новгород

2023 г.

**Оглавление**

[**Задание 1. Освоение темы «Статические поля», а также приложенной к ней программы.** 3](#_Toc131383405)

[Рисунок 1. Реализация скрипта, описывающего статистические поля. 3](#_Toc131383406)

[Рисунок 2. Результат реализации скрипта статистического поля. 4](#_Toc131383407)

[**Описание действия скрипта:** 4](#_Toc131383408)

[**Задание 2. Освоение темы «Статические методы», а также приложенной к ней программы.** 6](#_Toc131383409)

[Рисунок 3. Реализация скрипта, описывающего статистический метод. 6](#_Toc131383410)

[Рисунок 4. Результат реализации скрипта статистического метода. 6](#_Toc131383411)

[**Описание действия скрипта:** 6](#_Toc131383412)

[**Задание 3. Освоение темы «Создание копии объекта методом класса», а также приложенной к ней программы.** 8](#_Toc131383413)

[Рисунок 5. Реализация скрипта, описывающего создание копии объекта методом класса. 8](#_Toc131383414)

[Рисунок 6. Результат реализации скрипта для создания копии объекта методом класса. 9](#_Toc131383415)

[**Описание действия скрипта:** 9](#_Toc131383416)

# **Задание 1. Освоение темы «Статические поля», а также приложенной к ней программы.**

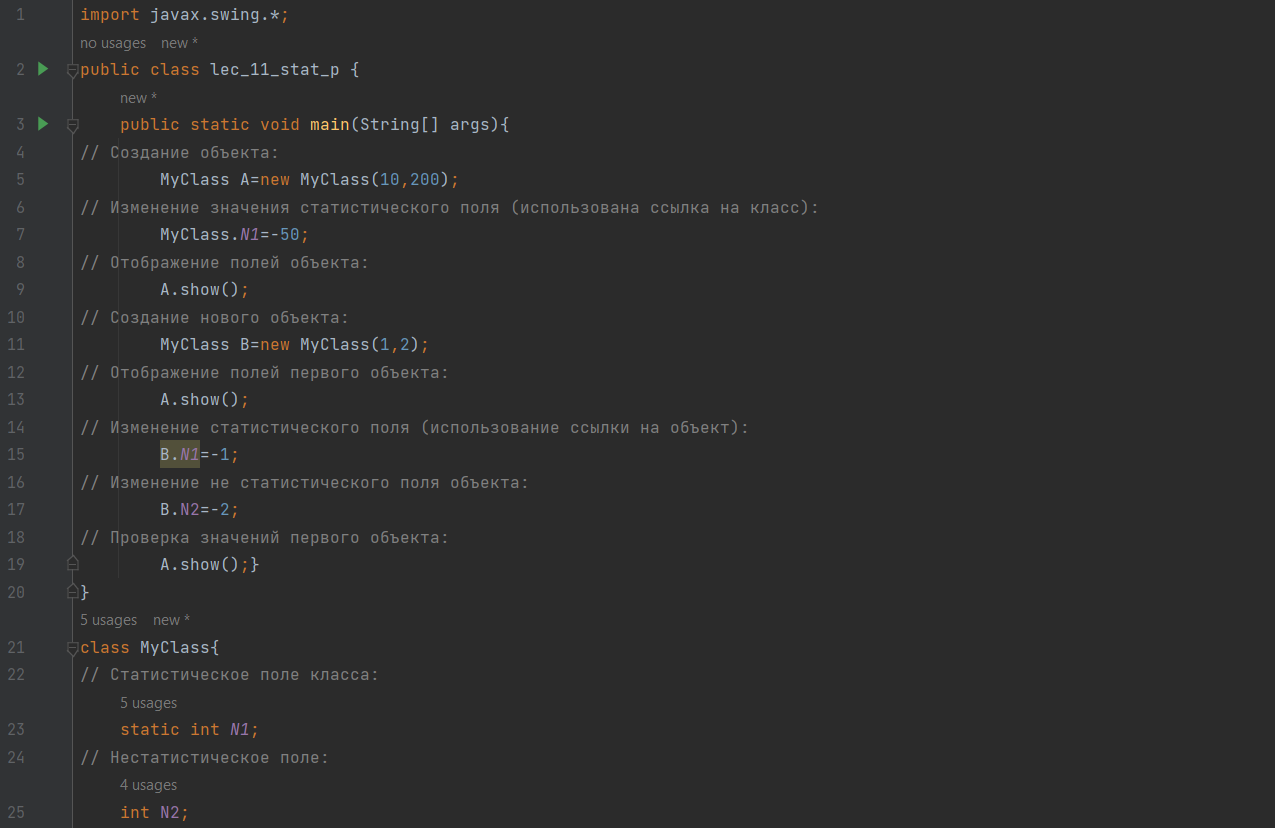
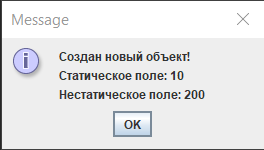
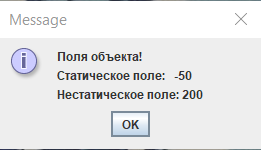
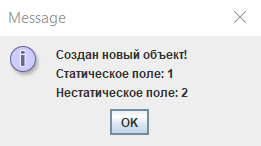
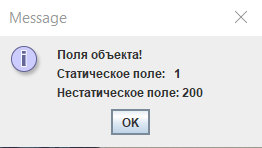
 

Рисунок 1. Реализация скрипта, описывающего статистические поля.









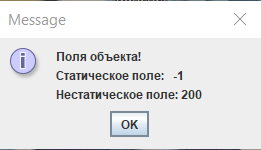


Рисунок 2. Результат реализации скрипта статистического поля.

## **Описание действия скрипта:**

В программе описан класс MyClass, содержащий два числовых поля: статическое поле N1 и нестатическое поле N2. Конструктору класса передается два аргумента: первый аргумент определяет значение статического поля N1, а второй аргумент определяет значение статического поля N2.

Кроме присваивания значения полям, в конструкторе формируется текстовое сообщение (локальная переменная текстового типа) с информацией о полях вновь созданного объекта (включая значение статического поля) и отображается соответствующее диалоговое окно.

Присваивание значения статическому полю в конструкторе является идеей не очень хорошей. Дело в том, что в такой схеме каждый раз при создании нового объекта будет изменяться значение статического поля, которое, как известно, одно на все объекты. Целесообразность использования статического поля в таком контексте представляется не очень целесообразной. Но в данном случае пример иллюстративный, поэтому проблемы никакой нет

Что касается инициализации статического поля, то ее можно выполнить прямо в описании класса, как для обычной переменной. Например, если объявить статическое поле в классе MyClass как static int N1=10, то каждый раз при запуске программы статическое поле будет получать значение 10.

Также в классе MyClass описан метод show(), с помощью которого отображаются сведения о полях уже существующего объекта. В главном методе программы командой MyClass A=new MyClass(10, 200) создается объект А. Статическое поле получает значение 10, а нестатическое поле – значение 200. При создании объекта автоматически отображается окно сообщения.

# **Задание 2. Освоение темы «Статические методы», а также приложенной к ней программы.**

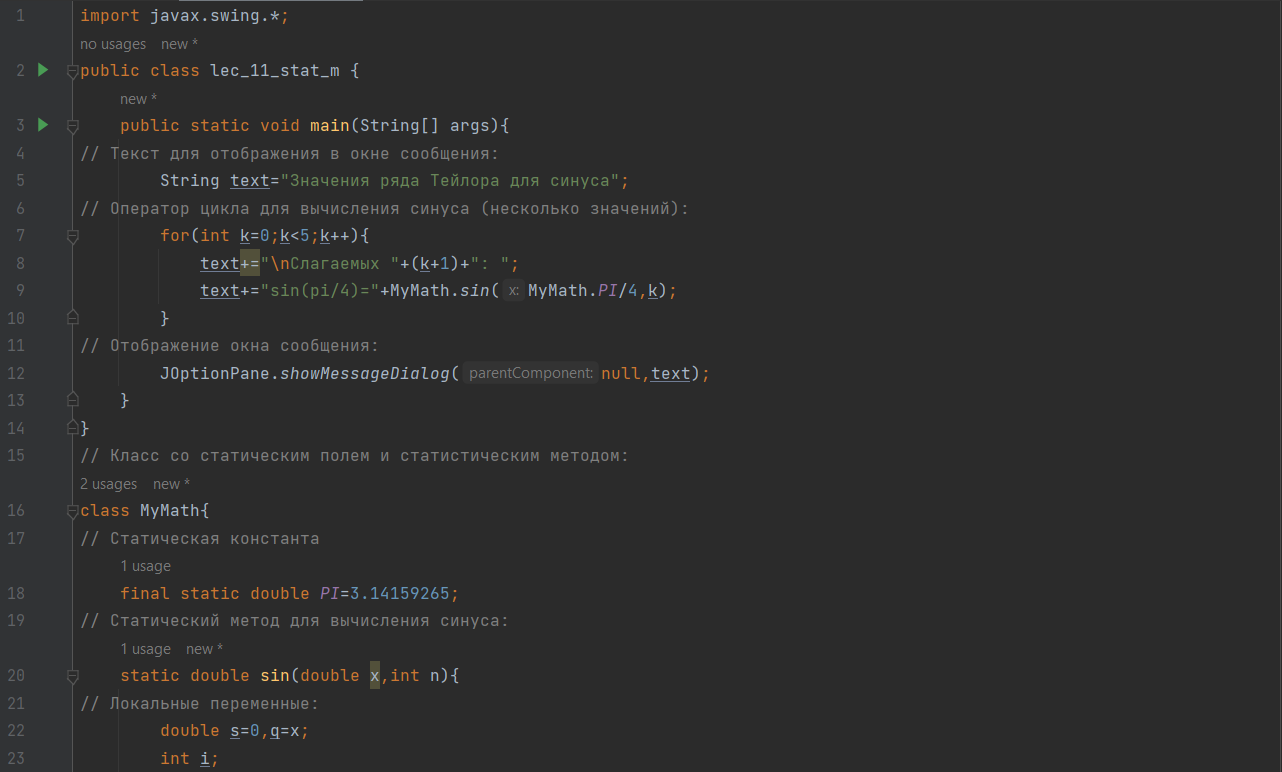
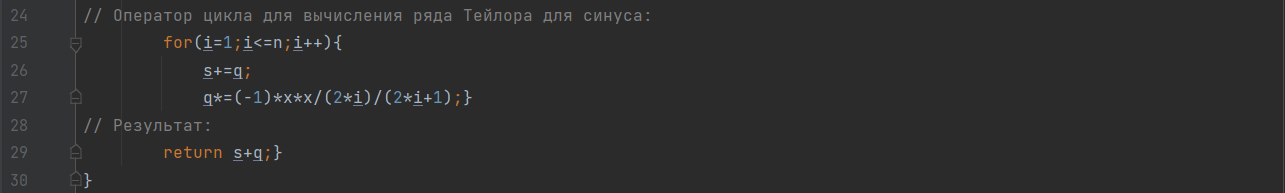
 

Рисунок 3. Реализация скрипта, описывающего статистический метод.

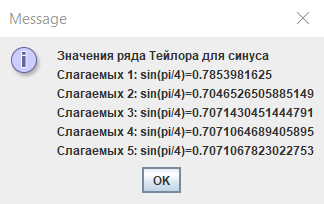


Рисунок 4. Результат реализации скрипта статистического метода.

## **Описание действия скрипта:**

В классе MyMath описано статическое поле и статический метод. Статическое действительное поле PI в качестве значения содержит приближение для числа π. Поле является не только статическим, но еще и постоянным, то есть константой, значение которой в программном коде изменить нельзя. О том, что поле является константой, свидетельствует ключевое слово final в описании поля. Кроме статического поля, в классе есть статический метод sin(). У метода два аргумента, и в качестве результата методом возвращается значение синуса. Синус вычисляется для значения первого аргумента – это переменная типа double. При вычислении используется ряд Тейлора.

Чем больше слагаемых в сумме взять, тем точнее результат (в идеале точное выражение для синуса – это бесконечная сумма). Второй аргумент метода sin() – целое число, которое определяет количество слагаемых при вычислении ряда для синуса. В теле метода реализуется алгоритм, который по известным значениям аргументов метода x и n позволяет вычислить (приближенное) значение для синуса. В частности, в теле метода объявляются локальные double переменные: s с начальным значением 0 (переменная, в которую записывается сумма для ряда) и переменная q с начальным значением x (в переменную q записывается очередная добавка к сумме). Кроме этого, объявляется целочисленная индексная переменная i для оператора цикла.

Сам оператора цикла выполняет две команды: инструкцией s+=q текущее значение суммы изменяется на величину добавки, а инструкцией q\*=(-1)\*x\*x/(2\*i)/(2\*i+1) вычисляется значение добавки для следующего итерационного шага. После выполнения оператора цикла в качестве значения возвращается величина s+q. Здесь принято во внимание, что последняя вычисленная в рамках оператора цикла добавка не была использована на последнем этапе цикла. На этом описание метода sin() и всего класса MyMath заканчивается.

В главном методе программы в классе StatMethDemo для разного количества слагаемых в ряде Тейлора для синуса (от 1 до 5 включительно) вычисляется значение синуса для аргумента π/4. При вычислениях используется статическое поле PI и статический метод sin() класса MyMath. Результат для каждого из вычислений заносится в текстовую переменную и потом все это отображается в окне сообщения.

# **Задание 3. Освоение темы «Создание копии объекта методом класса», а также приложенной к ней программы.**

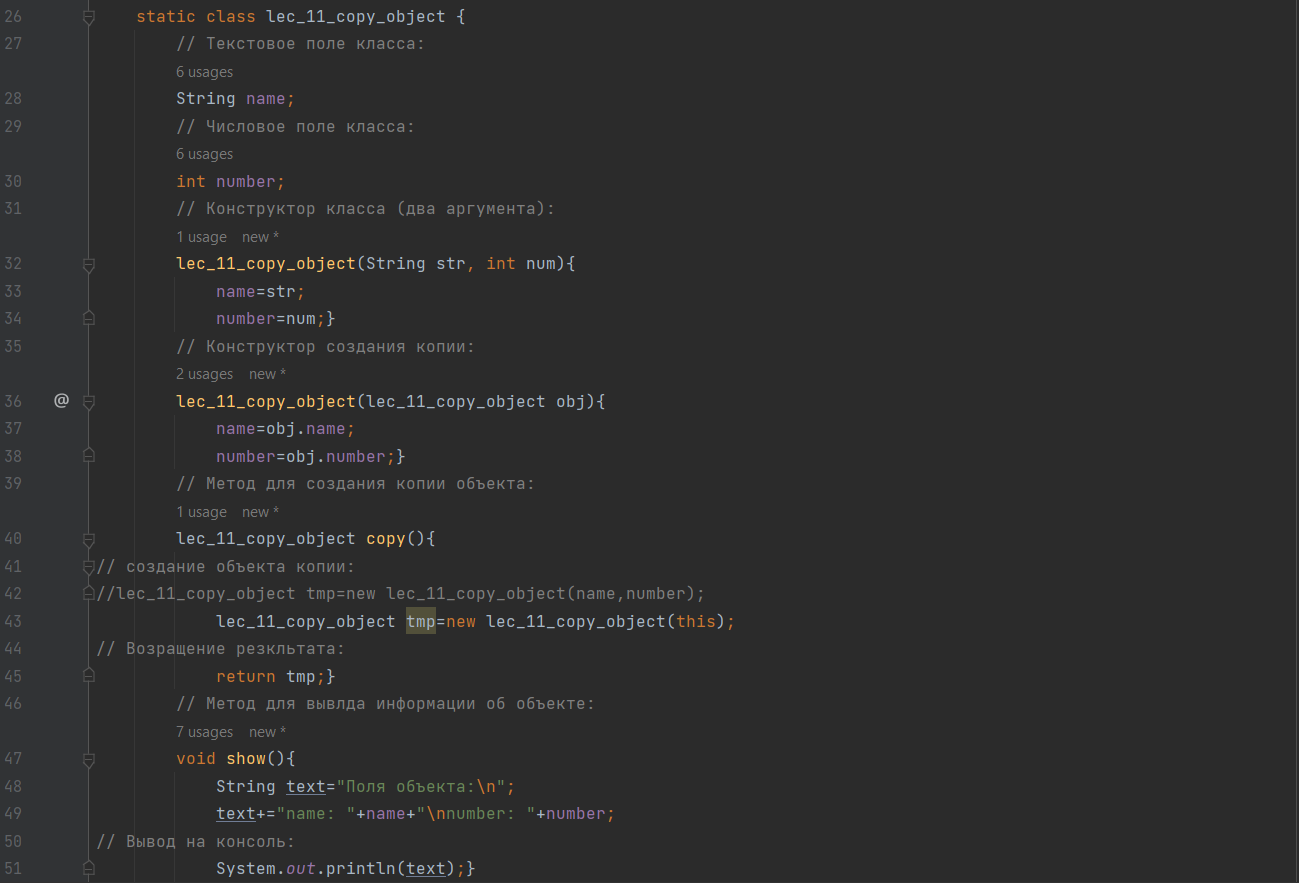
  

Рисунок 5. Реализация скрипта, описывающего создание копии объекта методом класса.

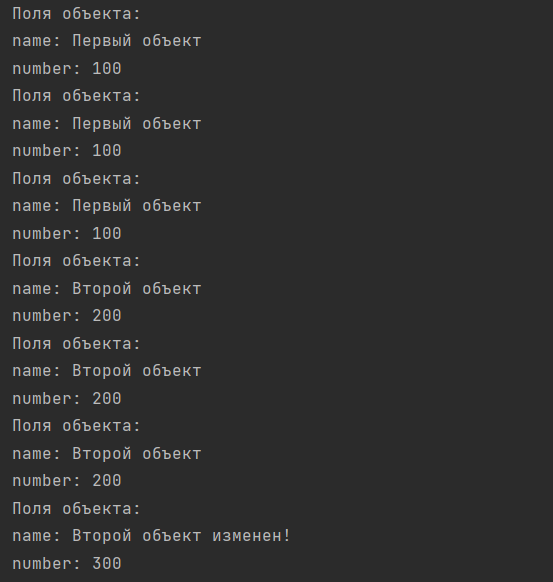


Рисунок 6. Результат реализации скрипта для создания копии объекта методом класса.

## **Описание действия скрипта:**

У класса два поля: текстовое поле name и целочисленное поле number. Перегруженный конструктор позволяет создавать объекты класса, передавая два аргумента (или уже существующий объект класса lec\_11\_copy\_object. В последнем случае поля создаваемого объекта заполняются в соответствии со значениями полей объекта, переданного аргументом конструктору.

Метод copy() предназначен для создания копии объекта. У метода нет аргументов. Он "копирует" тот объект, из которого вызывается.

В теле метода командой lec\_11\_copy\_object tmp=new lec\_11\_copy\_object (name, number) создается локальный объект tmp класса lec\_11\_copy\_object. Для создания объекта используется конструктор с двумя аргументами – значениями полей исходного объекта (из которого вызывается метод).

Вместо команды lec\_11\_copy\_object tmp=new lec\_11\_copy\_object (name, number) при создании локального объекта tmp можно было использовать команду lec\_11\_copy\_object tmp=new lec\_11\_copy\_object (this). Особенность команды состоит в том, что в ней использована инструкция this. Данная инструкция является ссылкой на объект, из которого вызывается метод. Другими словами, ключевое слово this, когда оно используется в программном коде метода, означает тот объект, из которого этот метод вызывается. В данном случае ссылка this передается в качестве аргумента конструктору создания копии.

Следующей командой return tmp ссылка на созданный объект возвращается в качестве результата метода.

Все локальные переменные метода существуют до тех пор, пока выполняется код метода, и доступны только в пределах программного кода этого метода. Но поскольку в данном случае объектная ссылка tmp возвращается в качестве значения, то по завершении работы метода объект, на который ссылается переменная tmp, из памяти не удаляется.

Информацию выводим на консоль. Для этих целей в классе lec\_11\_copy\_object описан метод show(). У метода нет аргументов, и он не возвращает результат. В теле метода объявляется локальная текстовая переменная text, в которую записывается текстовое сообщение, содержащее информацию о значении полей объекта, из которого вызывается метод. Соответствующий текст выводится в консольное окно с помощью команды System.out.println(text), в которой из поля-объекта out класса System вызывается метод println(), который выводит на консоль значение переданного ему аргумента. Это структура класса lec\_11\_copy\_object. В главном методе программы создаются два объекта objA и objB. Для этого используем команды lec\_11\_copy\_object objA= lec\_11\_copy\_object ("Первый объект",100) и lec\_11\_copy\_object objB=new lec\_11\_copy\_object (objA). Объект objA создается путем явной передачи конструктору значений полей, а объект objB создается как копия объекта objA. У объекта objB значения полей такие же, как и у объекта objA, но при этом объекты физически разные (то есть это не один и тот же объект). Для проверки командами objA.show() и objB.show() выводим на экран значения полей каждого из объектов (напомним, значения полей совпадают). Чтобы удостовериться, что объекты при этом разные, командами objB. name="Второй объект" и objB.number=200 меняем значения полей объекта objB. После командами objA.show() и objB.show() еще раз проверяем значения полей объектов. Результат для первого объекта objA не изменился, а у второго объекта objB поля изменили свои значения в соответствии с тем, что мы им предварительно присвоили. Так и должно быть, поскольку переменные objA и objB ссылаются на разные объекты.

Фразу вида "объект objA" следует понимать так: "объект, на который ссылается объектная переменная objA".

Теперь проверим работу метода copy(). Для этого командой objA=objB. copy() выполняем "копирование" объектов. В результате выполнения данной команды на основе объекта objB создается еще один объект с такими же значениями полей, и ссылка на него записывается в переменную objA. Поэтому после выполнения команды objA.show() увидим то же сообщение, что и при выводе на экран значений полей объекта objB. Но, как и в предыдущем случае, objA и objB – разные объекты. Убедиться в последнем легко. Командами objB.name="Второй объект изменен!" и objB.number=300 изменяем значения полей второго объекта, после чего командами objA.show() и objB.show() проверяем значения полей объектов. Несложно убедиться, что изменения коснулись полей лишь одного объекта – объекта objB. Поля объекта objA значений не изменили.