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

Задания лекции №11

Выполнила: студент группы:

ИБ-321

Кодоева Нели Тамазиевна

Нижний Новгород

2023 г.

Оглавление

[**Программа из темы «Статические поля» и ее описание** 3](#_Toc131372948)

[**Текст программы** 3](#_Toc131372949)

[**Программа** 3](#_Toc131372950)

[**Реализация программы** 4](#_Toc131372951)

[**Описание** 6](#_Toc131372952)

[**Программа из темы «Статические методы» и ее описание** 7](#_Toc131372953)

[**Текст программы** 7](#_Toc131372954)

[**Программа** 7](#_Toc131372955)

[**Реализация программы** 8](#_Toc131372956)

[**Описание программы** 8](#_Toc131372957)

[**Программа из темы «Создание копии объекта методом класса» и ее описание** 9](#_Toc131372958)

[**Текст программы** 9](#_Toc131372959)

[**Программа** 10](#_Toc131372960)

[**Реализация программы** 11](#_Toc131372961)

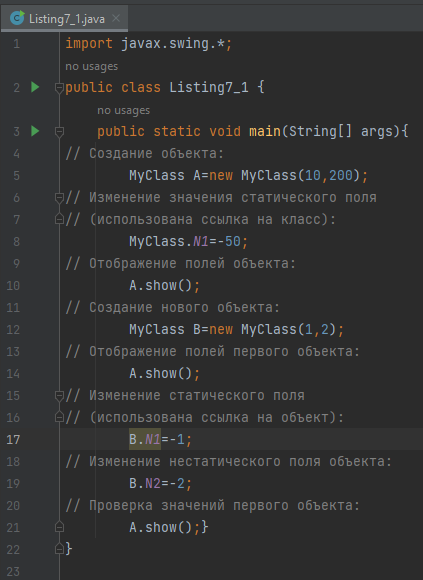
[**Описание программы** 12](#_Toc131372962)

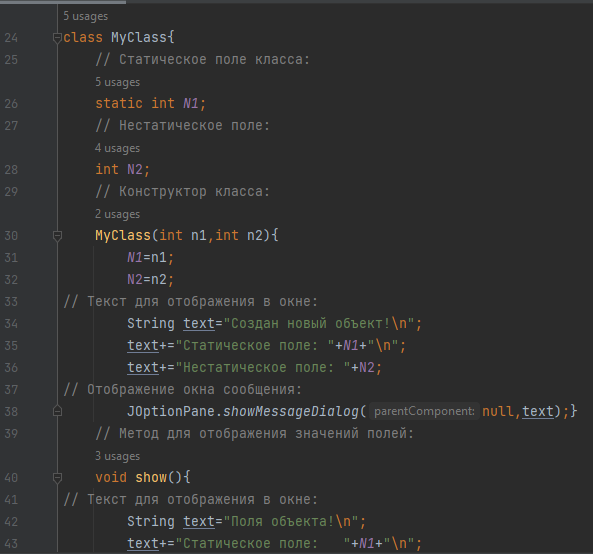
# **Программа из темы «Статические поля» и ее описание**

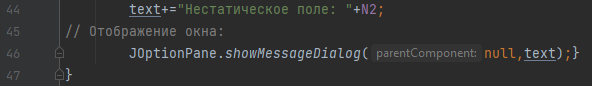
## **Текст программы**

import javax.swing.\*;  
public class Listing7\_1 {  
 public static void main(String[] args){  
 MyClass A=new MyClass(10,200);  
 MyClass.*N1*=-50;  
 A.show();  
 MyClass B=new MyClass(1,2);  
 A.show();  
 B.*N1*=-1;  
 B.N2=-2;  
 A.show();}  
}  
class MyClass{  
 static int *N1*;  
 int N2;  
 MyClass(int n1,int n2){  
 *N1*=n1;  
 N2=n2;  
 String text="Создан новый объект!\n";  
 text+="Статическое поле: "+*N1*+"\n";  
 text+="Нестатическое поле: "+N2;  
 JOptionPane.*showMessageDialog*(null,text);}  
 void show(){  
 String text="Поля объекта!\n";  
 text+="Статическое поле: "+*N1*+"\n";  
 text+="Нестатическое поле: "+N2;  
 JOptionPane.*showMessageDialog*(null,text);}  
}

## **Программа**

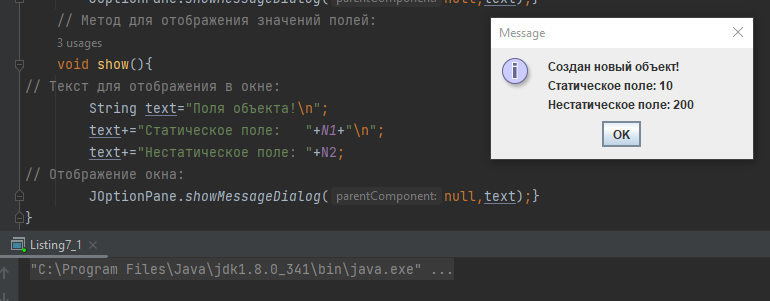


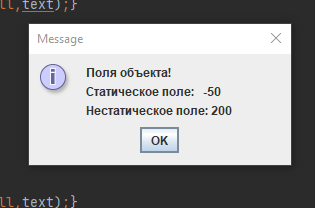


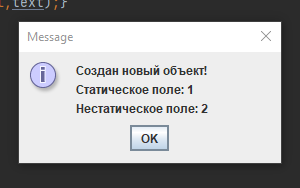


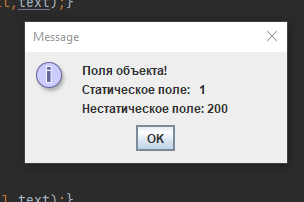
*Рис.1-3(программа для описания статических полей)*

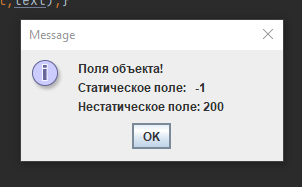
## **Реализация программы**

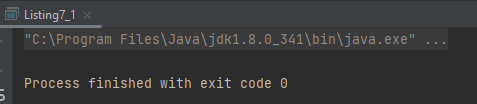












*Рис.4-9(результат программы)*

## **Описание**

В классе MyClass содержатся два числовых поля: статическое поле N1 и нестатическое поле N2. Конструктору этого класса передается два аргумента: первый, определяющий значение N1 и второй, определяющий значение N2.

Кроме присваивания значения полям, в конструкторе формируется текстовое сообщение с информацией о полях вновь созданного объекта и отображается соответствующее диалоговое окно.

Если мы объявим статическое поле static int N1=10 в классе MyClass, то каждый раз при запуске программы статическое поле будет получать значение 10, но этого не сделано, поэтому значение 10 каждый раз при запуске не объявляется.

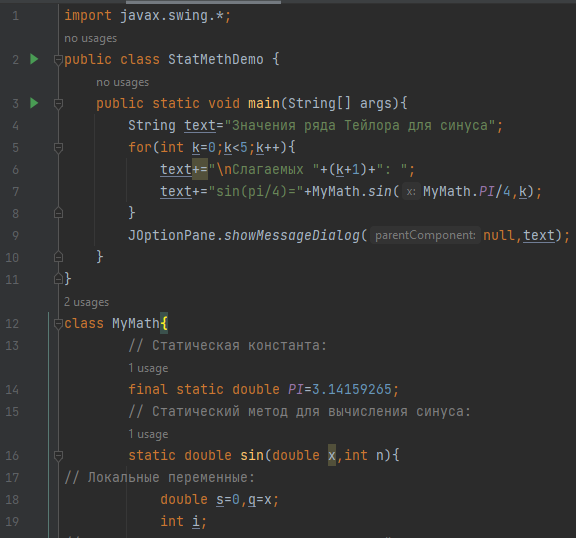
Также в классе MyClass описан метод show(), с помощью которого отображаются сведения о полях уже существующего объекта. В главном методе программы командой MyClass A=new MyClass(10, 200) создается объект А. Статическое поле получает значение 10, а нестатическое поле – значение 200.

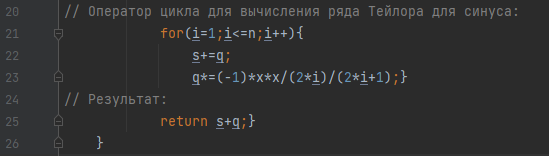
# **Программа из темы «Статические методы» и ее описание**

## **Текст программы**

import javax.swing.\*;  
public class StatMethDemo {  
 public static void main(String[] args){  
 String text="Значения ряда Тейлора для синуса";  
 for(int k=0;k<5;k++){  
 text+="\nСлагаемых "+(k+1)+": ";  
 text+="sin(pi/4)="+MyMath.*sin*(MyMath.*PI*/4,k);  
 }  
 JOptionPane.*showMessageDialog*(null,text);  
 }  
}  
class MyMath{  
 final static double *PI*=3.14159265;  
 static double sin(double x,int n){  
 double s=0,q=x;  
 int i;  
 for(i=1;i<=n;i++){  
 s+=q;  
 q\*=(-1)\*x\*x/(2\*i)/(2\*i+1);}  
 return s+q;}  
 }

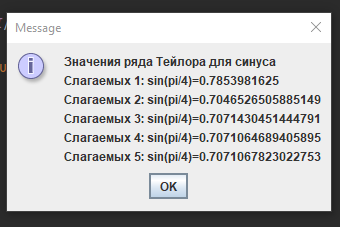
## **Программа**





*Рис.10-11(программа для описания статических методов)*

## **Реализация программы**



*Рис.12(результат программы)*

## **Описание программы**

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. Об этом 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. 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 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). Что касается самого оператора цикла, то за каждый цикл выполняется 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. Последняя вычисленная в рамках оператора цикла добавка не была использована на последнем этапе цикла.

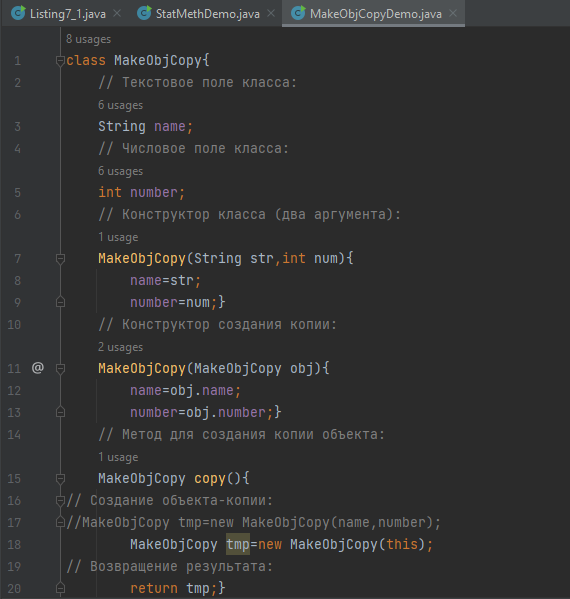
В главном методе программы в классе StatMethDemo для разного количества слагаемых в ряде Тейлора для синуса вычисляется значение синуса для аргумента π/4. При вычислениях используется статическое поле PI и статический метод sin() класса MyMath. Результат для каждого из вычислений заносится в текстовую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

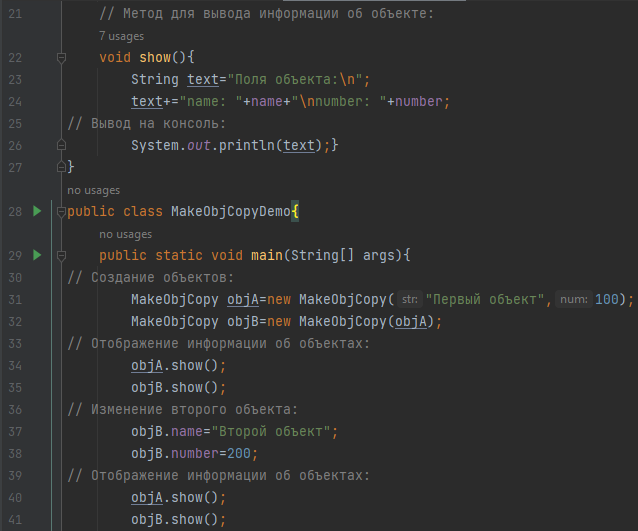
# **Программа из темы «Создание копии объекта методом класса» и ее описание**

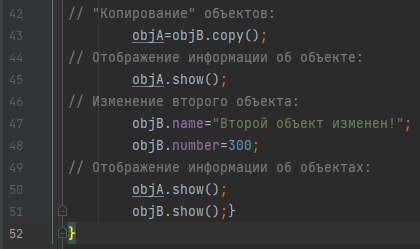
## **Текст программы**

class MakeObjCopy{  
 String name;  
 int number;  
 MakeObjCopy(String str,int num){  
 name=str;  
 number=num;}  
 MakeObjCopy(MakeObjCopy obj){  
 name=obj.name;  
 number=obj.number;}  
 MakeObjCopy copy(){  
//MakeObjCopy tmp=new MakeObjCopy(name,number);  
 MakeObjCopy tmp=new MakeObjCopy(this);  
 return tmp;}  
 void show(){  
 String text="Поля объекта:\n";  
 text+="name: "+name+"\nnumber: "+number;  
 System.*out*.println(text);}  
}  
public class MakeObjCopyDemo{  
 public static void main(String[] args){  
 MakeObjCopy objA=new MakeObjCopy("Первый объект",100);  
 MakeObjCopy objB=new MakeObjCopy(objA);  
 objA.show();  
 objB.show();  
 objB.name="Второй объект";  
 objB.number=200;  
 objA.show();  
 objB.show();  
 objA=objB.copy();  
 objA.show();  
 objB.name="Второй объект изменен!";  
 objB.number=300;  
 objA.show();  
 objB.show();}  
}

## **Программа**

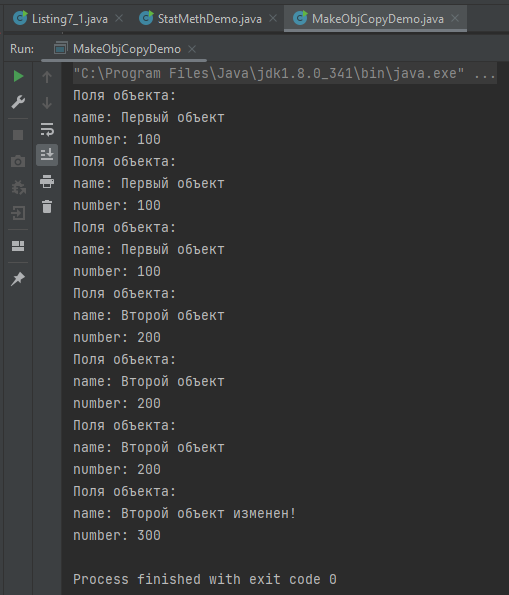






*Рис.13-15(программа для описания Создания копии объекта методом класса)*

## **Реализация программы**



*Рис.4-9(результат программы)*

## **Описание программы**

У класса два поля: текстовое поле name и целочисленное поле number. Перегруженный конструктор позволяет создавать объекты класса, передавая два аргумента (текст и целое число – значения полей объекта) или уже существующий объект класса MakeObjCopy. 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. При выполнении метода создается еще один, новый объект. Значения полей этого нового объекта совпадают со значениями полей исходного объекта (из которого вызывается метод copy()). В качестве результата метод возвращает ссылку на созданный объект.

В теле метода командой MakeObjCopy tmp=new MakeObjCopy(name, number) создается локальный объект tmp класса MakeObjCopy. Для создания объекта используется конструктор с двумя аргументами – значениями полей исходного объекта (из которого вызывается метод). Это фактически и есть та копия объекта, о которой шла речь.

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

Поскольку в данном случае выводится довольно много однотипного текста, вместо создания окон информацию выводим на консоль. Для этих целей в классе MakeObjCopy 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 MakeObjCopy. В главном методе программы создаются два объекта objA и objB. Для этого используем команды MakeObjCopy objA=new MakeObjCopy ("Первый объект",100) и MakeObjCopy objB=new MakeObjCopy (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. 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.