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 №12

«Тонкости работы с объектами 2»

Выполнил: студент группы: ИБ-321

Борисков Дмитрий Андреевич

Преподаватель: Окулич Виктор Иванович

Нижний Новгород

2023 г.

Оглавление

[**Задание 12.1** 3](#_Toc131904417)

[Рисунок 1. Реализация программы из рисунка 12.2, с ответами на вопросы (в комментариях). 3](#_Toc131904418)

[Рисунок 2. Результат реализации программы 12.2. 3](#_Toc131904419)

[Ответы на вопросы: 3](#_Toc131904420)

[Объяснение работы кода: 4](#_Toc131904421)

[**Задание 12.2** 4](#_Toc131904422)

[1. Реализовать программу рисунка 12.3 и объяснить его работу; 4](#_Toc131904423)

[Рисунок 3. Реализация программы 12.3. 4](#_Toc131904424)

[Рисунок 4. Результат реализации программы из рисунка 12.3. 5](#_Toc131904425)

[Объяснение работы кода: 5](#_Toc131904426)

[2. Сравнить код рисунка 12.3 с кодом рисунка 12.4 и найти ошибку, и исправить её. 6](#_Toc131904427)

[Рисунок 5. Реализация скрипта из рисунка 12.4, с внесёнными правками. 6](#_Toc131904428)

[Рисунок 6. Результат реализации скрипта рисунка 12.4. 6](#_Toc131904429)

[Ошибка: 6](#_Toc131904430)

[**Задание 12.3** 7](#_Toc131904431)

[Рисунок 7. Реализация программы из 12.5. 7](#_Toc131904432)

[Рисунок 8. Результат работы скрипта из рисунка 12.5. 7](#_Toc131904433)

[Объяснение работы кода: 7](#_Toc131904434)

[Класс lec\_12\_3: 7](#_Toc131904435)

[Класс First: 8](#_Toc131904436)

[Класс Second: 8](#_Toc131904437)

[**Задание 12.4.1** 9](#_Toc131904438)

[Рисунок 9. Рисунок к заданию 12.4.1. 9](#_Toc131904439)

[Ответьте на следующие вопросы: 9](#_Toc131904440)

[Ответы на вопросы: 9](#_Toc131904441)

[**Задание 12.4.2** 10](#_Toc131904442)

[Рисунок 10. Рисунок к заданию 12.4.2. 10](#_Toc131904443)

[Ответьте на следующие вопросы: 10](#_Toc131904444)

[Ответы на вопросы: 10](#_Toc131904445)

[**Задание 12.4.3** 11](#_Toc131904446)

[Рисунок 11. Рисунок к заданию 12.4.3. 11](#_Toc131904447)

[Ответьте на следующие вопросы: 11](#_Toc131904448)

[Ответы на вопросы: 11](#_Toc131904449)

[**Задание 12.5** 12](#_Toc131904450)

[Рисунок 12. Реализация программы к заданию 12.5. 12](#_Toc131904451)

[Рисунок 13. Результат реализации программы к заданию 12.5. 13](#_Toc131904452)

[Объяснение кода: 13](#_Toc131904453)

# **Задание 12.1**

Реализовать код рисунка 12.2, ответить на имеющиеся в нём вопросы и объяснить работу кода

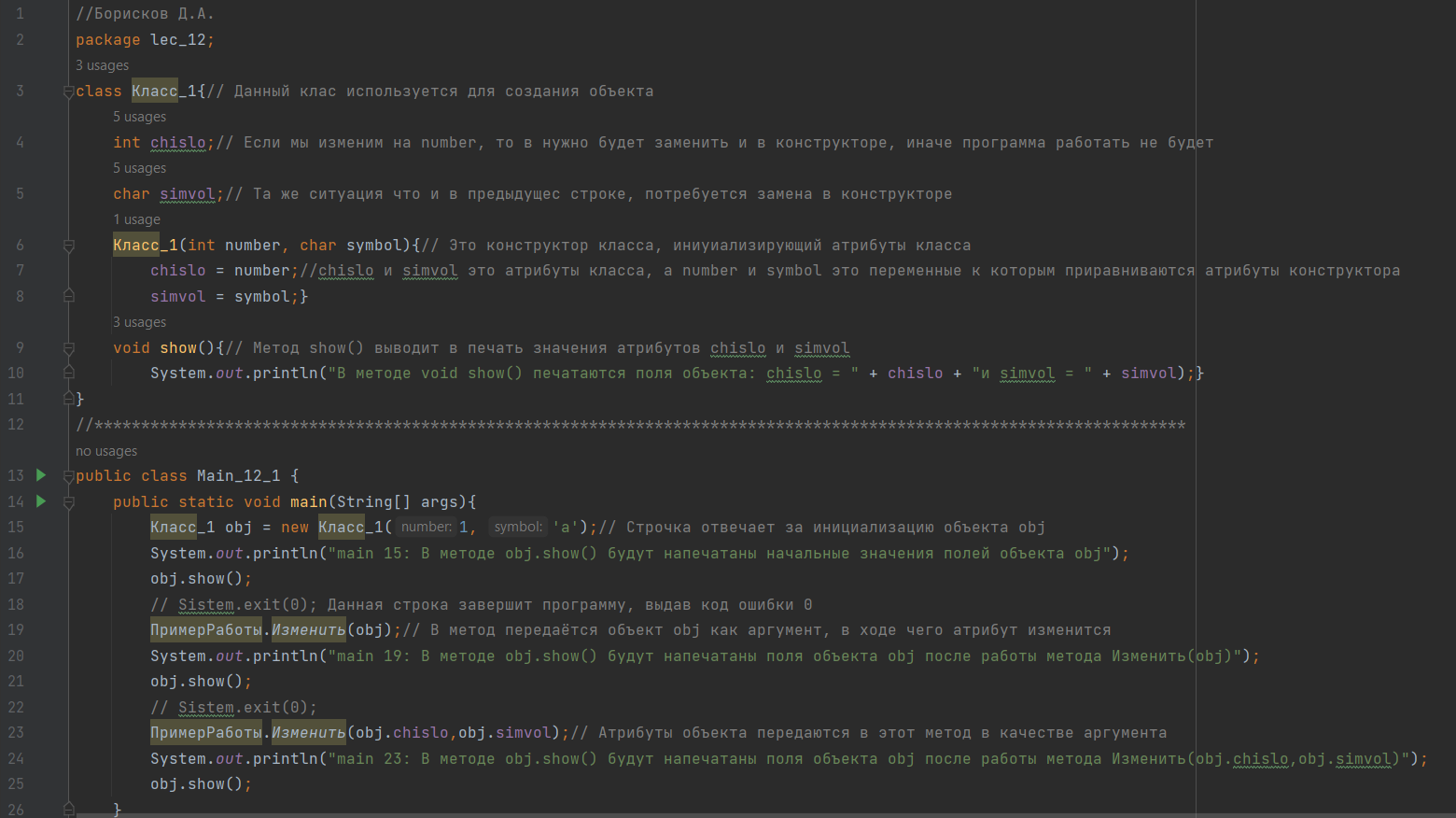
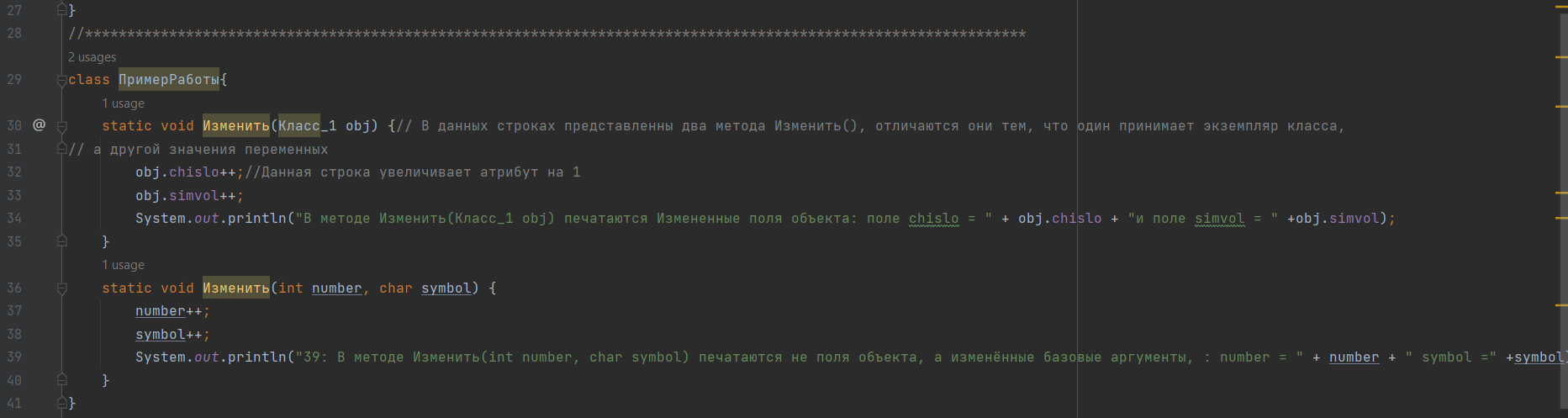
 

Рисунок 1. Реализация программы из рисунка 12.2, с ответами на вопросы (в комментариях).

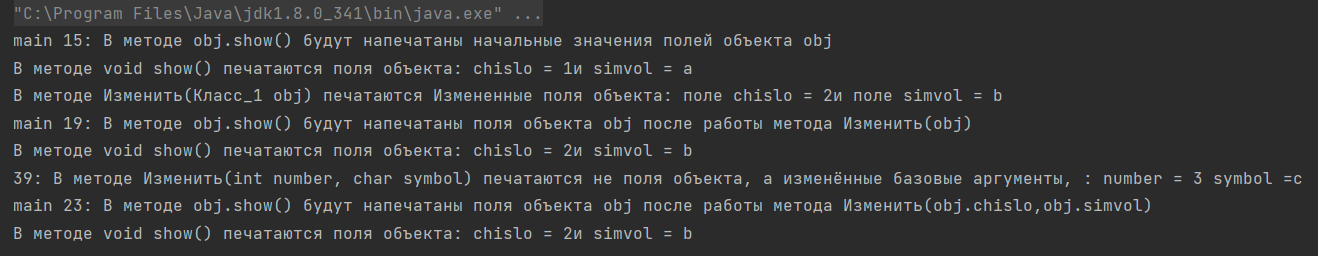


Рисунок 2. Результат реализации программы 12.2.

## Ответы на вопросы:

1. Если мы изменим на number, то в нужно будет заменить и в конструкторе, иначе программа работать не будет;
2. Та же ситуация что и в предыдущес строке, потребуется замена в конструкторе;
3. Это конструктор класса, инициализирующий атрибуты класса;
4. Метод show() выводит в печать значения атрибутов chislo и simvol;
5. Строчка отвечает за инициализацию объекта obj;
6. Данная строка завершит программу, выдав код ошибки 0;
7. В метод передаётся объект obj как аргумент, в ходе чего атрибут изменится;
8. Атрибуты объекта передаются в этот метод в качестве аргумента;
9. В данных строках представлены два метода Изменить(), отличаются они тем, что один принимает экземпляр класса, а другой значения переменных;
10. Данная строка увеличивает атрибут на 1.

## Объяснение работы кода:

В коде присутствуют 2 класса *Класс\_1* и *ПримерРаботы*. Объекты класса *Класс\_1* имеют 2 атрибута: *chislo* и *simvol*.

При помощи класса *ПримерРаботы* мы изменяем атрибуты класса *Класс\_1:* число увеличивается на 1, а символ заменяется на следующий по алфавиту.

С помощью метода *ПримерРаботы.Изменить(obj)* изменяется объект полностью, а с помощью метода *ПримерРаботы.Изменить(obj.chislo,obj.simvol)* изменяются атрибуты, не не сам объект.

# **Задание 12.2**

## 1. Реализовать программу рисунка 12.3 и объяснить его работу;

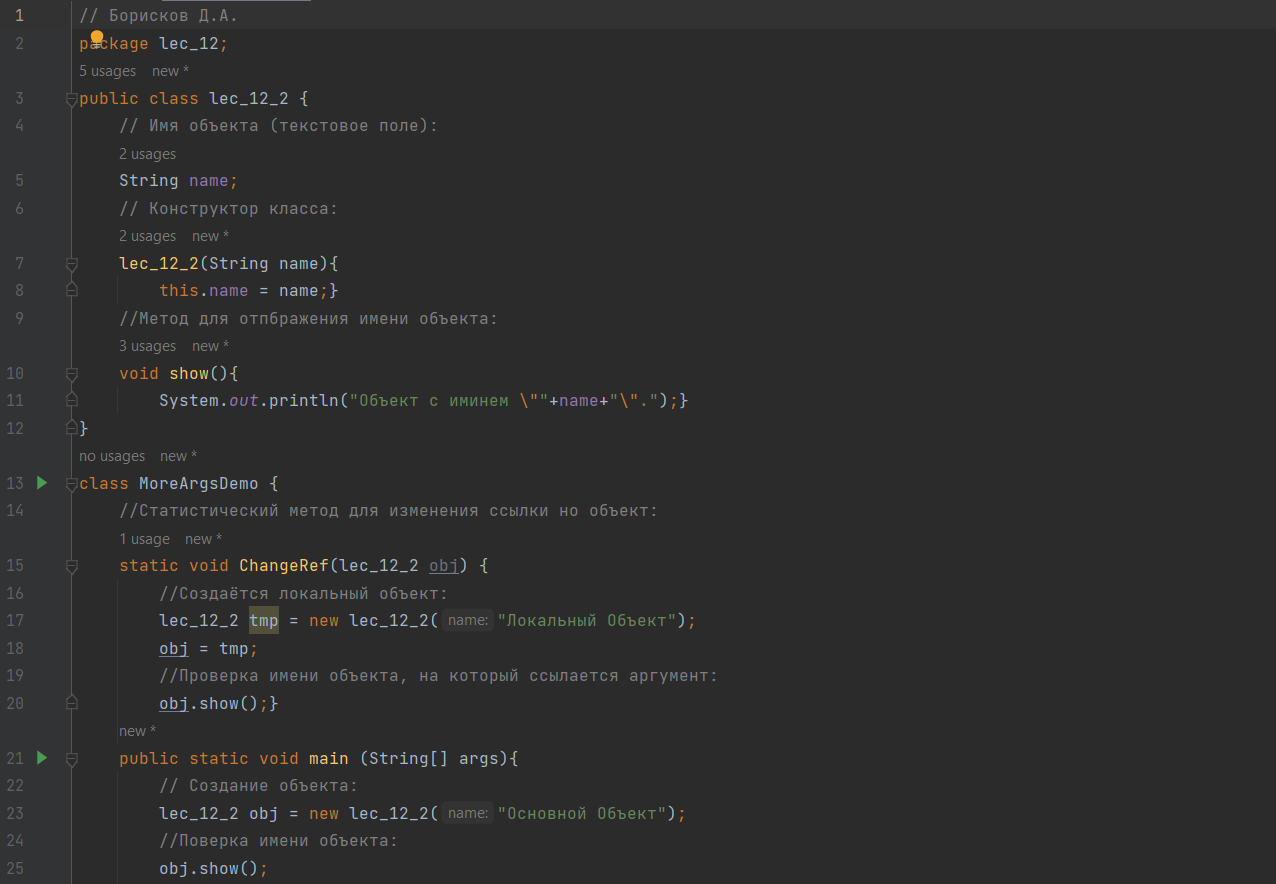
 

Рисунок 3. Реализация программы 12.3.

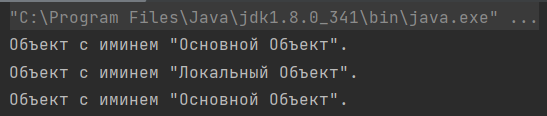


Рисунок 4. Результат реализации программы из рисунка 12.3.

### Объяснение работы кода:

В программе объявлены два класса. Класс *lec\_12\_2* достаточно прост. У него всего одно текстовое поле *name* (отождествляем значение этого поля с именем объекта), конструктор с одним аргументом (значение, которое присваивается полю *name*) и метод *show ()* для отображения значения текстового поля (имени объекта).

В методе *show ()* при выводе консольного сообщения в тексте отображаются двойные кавычки. Для отображения двойных кавычек в тексте перед ними указываем косую черту, то есть для вставки двойных кавычек в текст используем инструкцию \".

Класс *MoreArgsDemo* содержит главный метод программы и статический метод *ChangeRef()*, аргументом которому передается объект *obj* класса *lec\_12\_2* (точнее, объектная переменная этого класса). В методе создается локальный объект *tmp* класса *lec\_12\_2* (с именем "Локальный Объект") и ссылка на него присваивается аргументу метода (команда *obj=tmp*). После этого командой *obj.show()* проверяем, на какой объект ссылается переменная *obj*.

В главном методе программы создаем объект *obj* с передачей конструктору класса *lec\_12\_2* текстового аргумента "Основной Объект". Проверку имени объекта выполняем с помощью команды *obj.show()*. Затем пытаемся изменить значение объектной переменной, вызвав команду *ChangeRef(obj)* (предвосхищая результат, отметим, что попытка будет неудачной). Затем снова проверяем имя объекта, на который ссылается переменная *obj*, для чего используем команду *obj.show()*.

## 2. Сравнить код рисунка 12.3 с кодом рисунка 12.4 и найти ошибку, и исправить её.

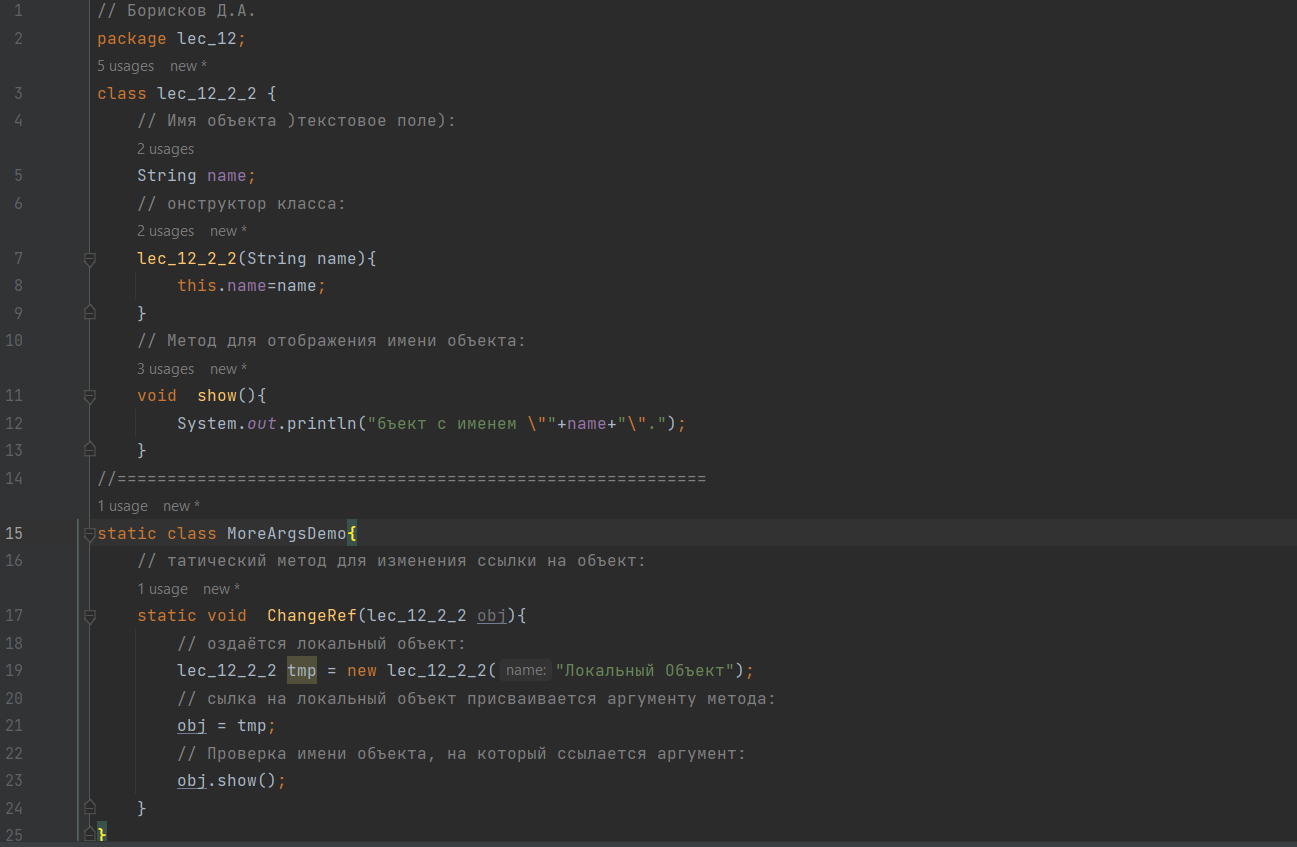
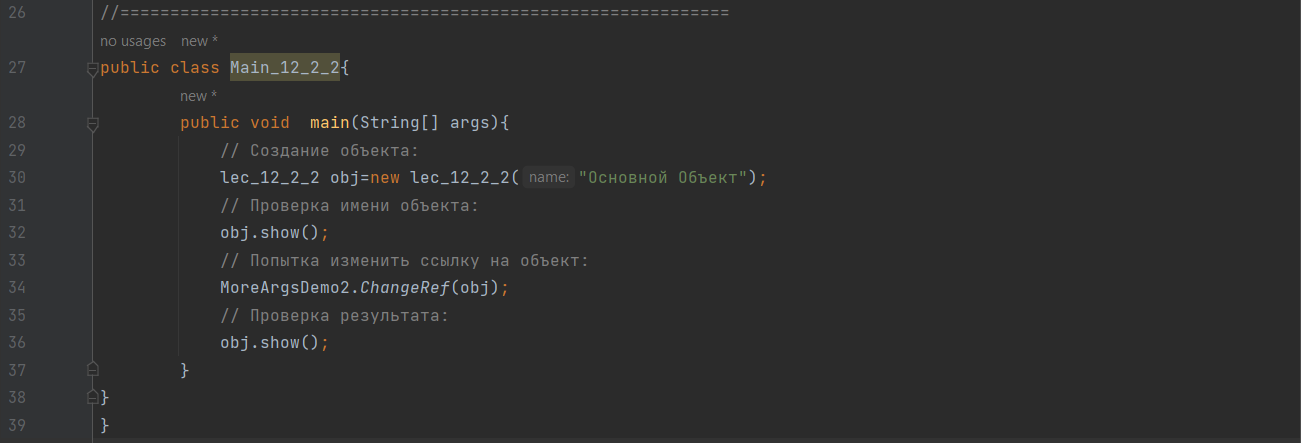
 

Рисунок 5. Реализация скрипта из рисунка 12.4, с внесёнными правками.

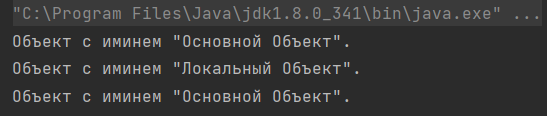


Рисунок 6. Результат реализации скрипта рисунка 12.4.

### Ошибка:

Класс *MoreArgsDemo* нужно сделать статическим, в этом случае программа будет работать.

# **Задание 12.3**

Разобрать работу программы на рисунке 12.5, определив особенности работы с массивами и объектами.

Рисунок 7. Реализация программы из 12.5.

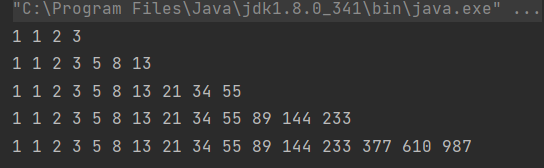


Рисунок 8. Результат работы скрипта из рисунка 12.5.

## Объяснение работы кода:

### Класс lec\_12\_3:

Размер массива на этапе создания задается целочисленной переменной count (в данном случае значение равно 5). Далее с помощью оператора цикла для каждой из объектных переменных в массиве ObjAr создается объект класса Second и ссылка на этот объект записывается в соответствующую переменную. Для индексной переменной i, которая в операторе цикла пробегает значения от 0 до ObjAr. length-1 включительно, значение элементу массива ObjAr[i] присваивается командой ObjAr[i]=new Second(3\*i+4).

В результате в объектах класса Second, на которые ссылаются объектные переменные в массиве ObjAr, поле obj ссылается на объекты класса First с массивами размера 4, 7, 10 и так далее.

### Класс First:

Класс First содержит поле nums, представляющее собой переменную массива, который будет состоять из целых чисел. На первый взгляд кажется, что nums – это поле-массив, но это не так. Поле-массив создаётся в конструкторе класса First и связывается с переменной nums. Размер массива определяется целочисленным аргументом, который передается конструктору. То есть массив появляется только на этом этапе. До этого была только переменная массива. После этого можно спокойно использовать имя переменной nums для манипулирования с массивом.

Значения первому и второму элементам массива присваивается командами nums[0]=1 и nums[1]=1. Каждый следующий элемент массива определяется как сумма двух предыдущих элементов. В результате получаем последовательность Фибоначчи.

Кроме конструктора, в классе First описан еще и метод show(), с помощью которого на консоль выводится последовательность элементов массива – поля класса.

### Класс Second:

В классе Second объявляется объектная переменная obj класса First, которая является полем класса Second. Как и в случае с полем – переменной массива, в данном случае поле obj является всего лишь ссылкой на объект, которого пока нет, и который сам по себе не появляется. Его необходимо создать. Создается объект класса First для поля obj в конструкторе класса Second. У конструктора один целочисленный аргумент, который определяет размер поля-массива в объекте, на который ссылается поле obj. Сам объект создается командой obj=new First(n). Командой obj.show() содержимое поля-массива объекта obj выводится на консоль. Все это происходит при вызове конструктора класса Second. Собственно, на конструкторе код класса Second заканчивается.

# **Задание 12.4.1**

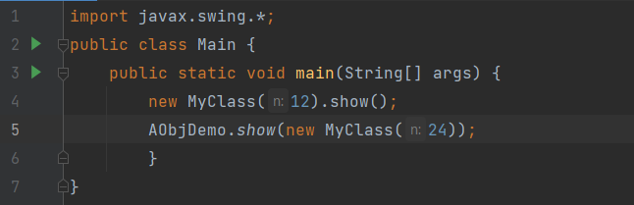


Рисунок 9. Рисунок к заданию 12.4.1.

## Ответьте на следующие вопросы:

1. В какой строчке создаётся анонимный объект?
2. Какие ещё классы присутствуют в программе и что они делают?
3. В каком виде будут выводится результаты работы всей программы?
4. Что делает строка 4?
5. Что делает строка 5 и что является аргументом в методе, используемым в этой строке?

## Ответы на вопросы:

1. Анонимный объект создаётся в строке 4;
2. В программе присутствуют класс *MyClass* который выводит массив и класс *AobjDemo* выводящий элементы класса *MyClass*;
3. Результаты программы будут выводится в отдельном окне;
4. Строка 4 создаёт анонимный объект класса, а также применяет метод *show(),* который выводит элементы класса в отдельное окно;
5. Строчка 5 применяет метод *show()*, выводящий элементы объекта в отдельном окне. Аргументом является объект класса *MyClass*.

# **Задание 12.4.2**

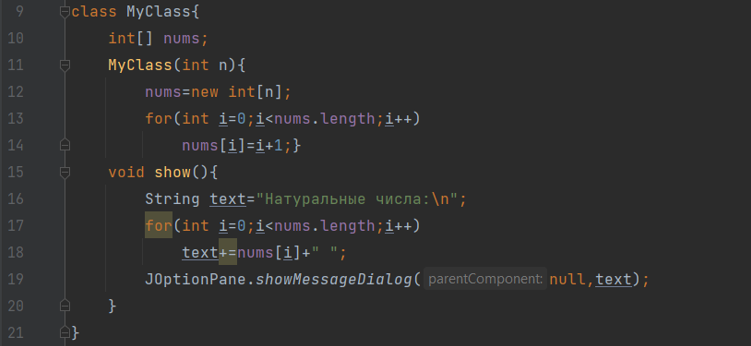


Рисунок 10. Рисунок к заданию 12.4.2.

## Ответьте на следующие вопросы:

1. Зачем используется класс MyClass?
2. Что создаётся в строке 12 кода на рисунке 12.11?
3. Что создаётся в конструкторе класса MyClass?

## Ответы на вопросы:

1. Класс *MyClass* используется для создания массива, содержащего натуральные числа, которые увеличиваются на 1 порядок, с каждым значением;
2. В строке 12 происходит инициализация массивом;
3. В конструкторе *MyClass* создаётся массив, заполненный натуральными числами.

# **Задание 12.4.3**

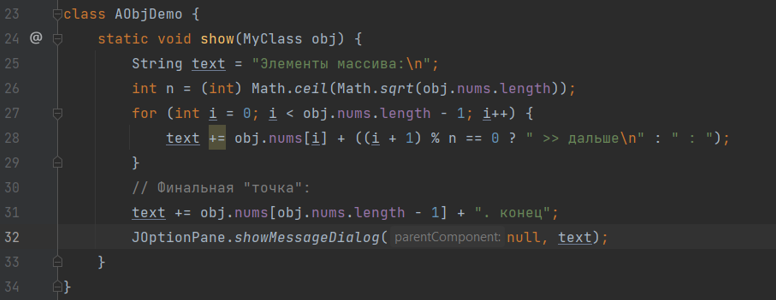


Рисунок 11. Рисунок к заданию 12.4.3.

## Ответьте на следующие вопросы:

1. Как характеризовать метод в строке 24 рисунка 12.12? Отличается ли он от класса в строке 15 рисунка 12.11 и, если отличается, то чем?
2. Что передаётся аргументом этого методу?
3. Каковы особенности статического метода в Java?
4. Как работает строка 26?
5. Как работает строка 28?
6. Сколько выводов будет выведено в результате работы программы в целом?

## Ответы на вопросы:

1. Метод *show()* в троке 24 является статическим и может быть вызван без создания экземпляра класса. Метод *show()* отличается от метода из строки 15 дополнительными математическими формулами и вычислениями;
2. Этому методу передаётся объект класса *MyClass*;
3. Особенностью статического метода в Java является то, что они могут вызываться без создания экземпляра класса;
4. Строка 26 объявляет переменную n, ей присваивается метод *ceil()*, котором, в качестве значения возвращается ближайшее наименьшее целое число.
5. Строка 28 отвечает за составление строки, которая будет выводится в окне.
6. В результате работы программы в целом будет выедено 2 окна.

# **Задание 12.5**

Разобрать работу программы на рисунке 12.6

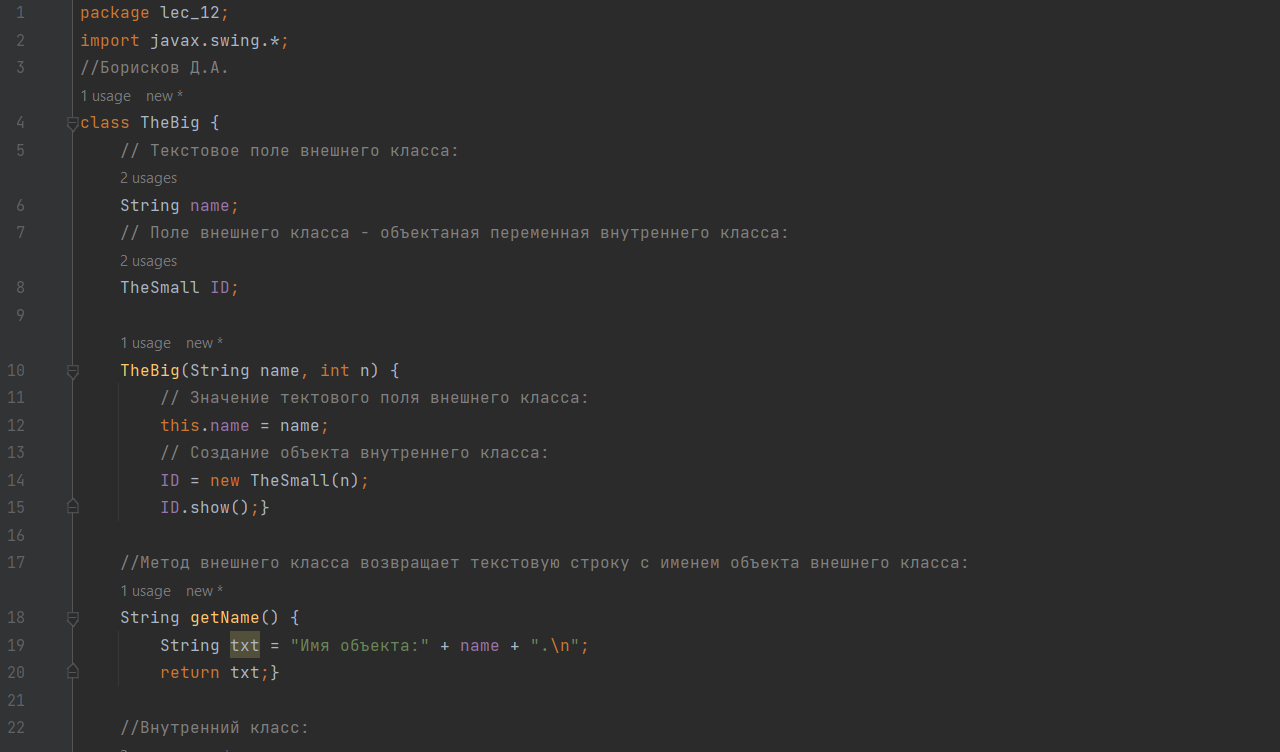
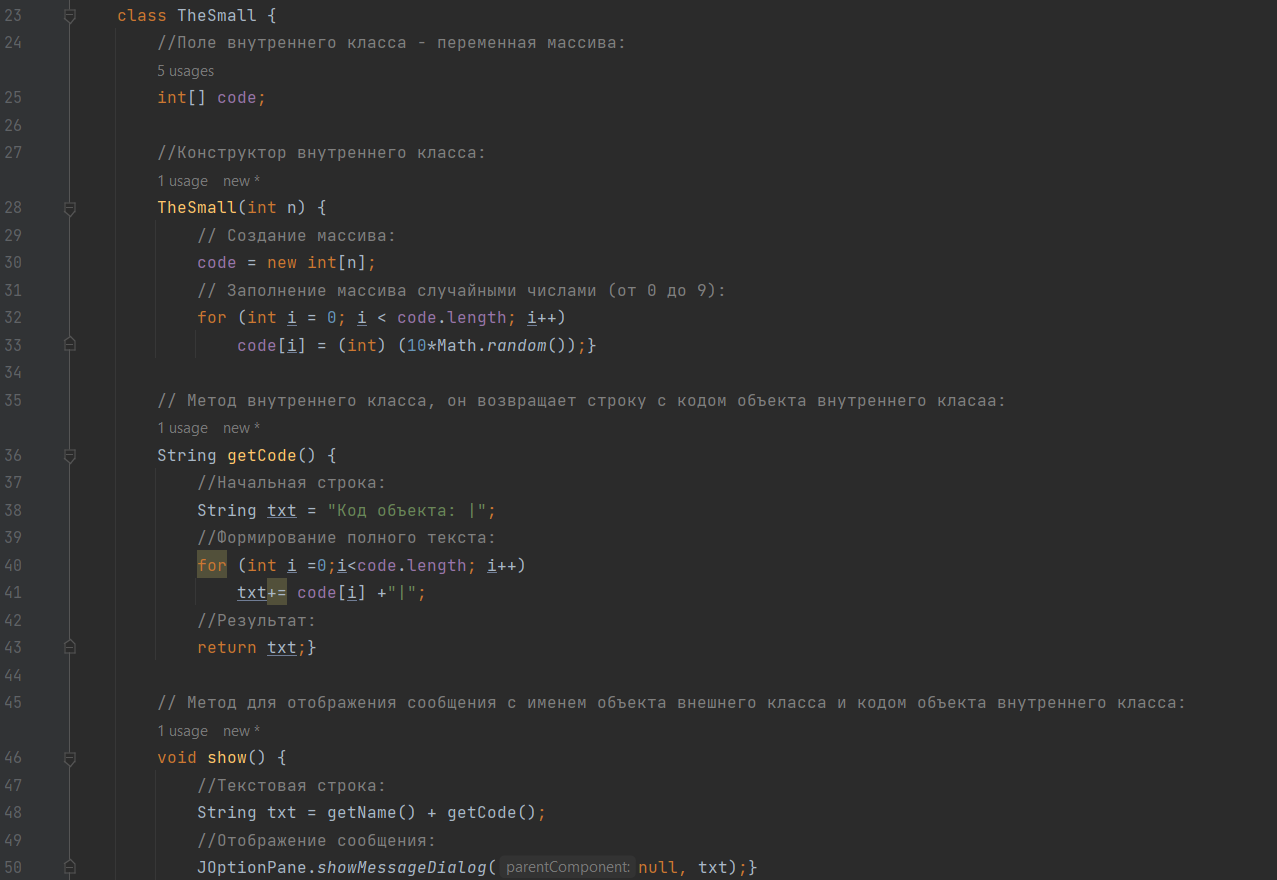
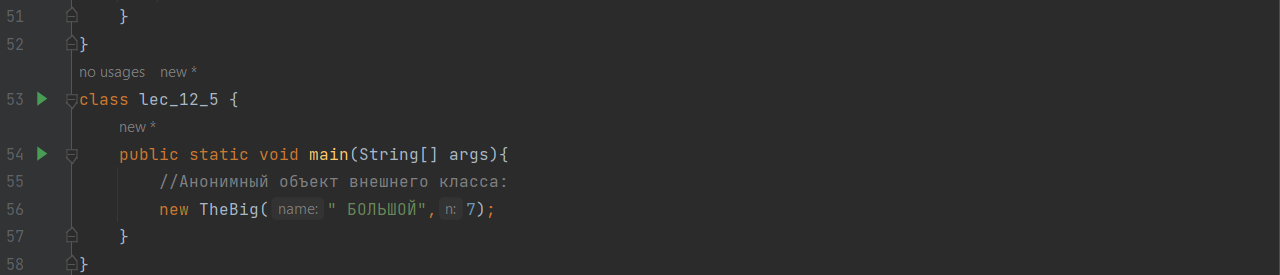
  

Рисунок 12. Реализация программы к заданию 12.5.

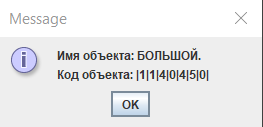


Рисунок 13. Результат реализации программы к заданию 12.5.

## Объяснение кода:

Класс TheBig содержит внутренний класс TheSmall.

У класса TheBig есть конструктор и метод getName(). Метод getName() в качестве результата возвращает текстовое значение, которое состоит из фразы «Имя объекта» и значения текстового поля name. В теле метода командой String txt="Имя объекта: "+name+".\n" создается текстовая переменная txt, которая командой return txt возвращается в качестве результата метода.

Конструктору класса TheBig передаются два аргумента: текстовый и целочисленный. Текстовый в качестве значения присваивается полю name. Целочисленный нужен для передачи в качестве аргумента конструктору внутреннего класса.

Командой this.name=name полю name в качестве значения присваивается ссылка на объект, переданный аргументом конструктору. Поскольку имя поля и имя аргумента в данном случае совпадают, при ссылке на поле используется ключевое слово this.

Командой ID=new TheSmall(n) создается объект внутреннего класса, и ссылка на него записывается в поле ID. Командой ID.show() из вновь созданного объекта вызывается метод show().

У внутреннего класса TheSmall есть поле code. Это переменная массива, она ссылается на массив из целых чисел. Конструктору в качестве аргумента передается целое число, определяющее, сколько элементов будет в массиве. Создание массива и присваивание ссылки на него полю code внутреннего класса выполняется командой code=new int[n].

Далее в операторе цикла массив заполняется случайными числами в диапазоне значений от 0 до 9 включительно. Для генерирования случайного числа используем метод random() из встроенного класса Math (инструкция Math.random()). Проблема, однако, в том, что методом генерируется случайное действительное число (тип double) в диапазоне от 0 (включительно) до 1 (не включая). Чтобы из такого числа получить целое число, умножаем результат вызова метода Math.random() на 10. Получаем действительное число от 0 включительно и меньшее 10.

Метод getCode() класса TheSmall возвращает в качестве значения текстовую строку (тип String). Эта строка содержит вспомогательный текст и код, который представляет собой последовательность цифр, записанных в массив code. В теле метода командой String txt="Код объекта: |" объявляется текстовая строка с начальным значением, к которому последовательно дописываются в рамках оператора цикла цифры из массива code. В качестве разделителя между цифрами используется вертикальная черта. Полученное текстовое значение (переменная txt) командой return txt возвращается в качестве результата метода.

Метод show(). В методе командой String txt=getName()+getCode() создается текстовая строка, которая отображается в диалоговом окне с помощью команды JOptionPane.showMessageDialog(null,txt).