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 8-ой главе из учебника Васильева А.Н.

Выполнил: студент группы: ИБ-321

Борисков Дмитрий Андреевич

Преподаватель: Окулич Виктор Иванович

Нижний Новгород

2023 г.

**Оглавление**

[**Вступление** 4](#_Toc132415782)

[**Основы наследования** 4](#_Toc132415783)

[Пример наследования классов 5](#_Toc132415784)

[Рисунок 1. Реализация программы «Наследование классов». 5](#_Toc132415785)

[Рисунок 2. Результат реализации программы «Наследование классов». 6](#_Toc132415786)

[Описание работы программы 6](#_Toc132415787)

[**Конструктор подкласса** 7](#_Toc132415788)

[Пример конструктора подкласса 8](#_Toc132415789)

[Рисунок 3. Реализация программы «конструктор класса». 8](#_Toc132415790)

[Рисунок 4. Результат реализации программы «Конструктор класса». 9](#_Toc132415791)

[Описание работы программы 9](#_Toc132415792)

[**Переопределение методов** 10](#_Toc132415793)

[Пример переопределения методов 11](#_Toc132415794)

[Рисунок 5. Реализация программы «Переопределение методов». 11](#_Toc132415795)

[Рисунок 6. Результат реализации программы «Переопределение методов». 12](#_Toc132415796)

[Описание работы программы 12](#_Toc132415797)

[**Закрытые члены класса** 13](#_Toc132415798)

[Пример закрытых членов класса 13](#_Toc132415799)

[Рисунок 7. Реализация программы «Закрытые члены класса». 13](#_Toc132415800)

[Рисунок 8. Результат реализации программы «Закрытые члены класса». 14](#_Toc132415801)

[Описание работы программы 14](#_Toc132415802)

[**Объектные переменные суперклассов** 15](#_Toc132415803)

[Пример объектных переменных суперклассов 15](#_Toc132415804)

[Рисунок 9. Реализация программы «Объектные переменные суперклассов». 15](#_Toc132415805)

[Рисунок 10. Результат реализации программы «Объектные переменные суперклассов». 16](#_Toc132415806)

[Описание работы программы 16](#_Toc132415807)

[**Абстрактные классы и интерфейсы** 17](#_Toc132415808)

[Пример абстрактных классов 17](#_Toc132415809)

[Рисунок 11. Реализация программы «Абстрактные классы». 17](#_Toc132415810)

[Рисунок 12. Результат реализации программы «Абстрактные классы». 18](#_Toc132415811)

[Описание работы программы 18](#_Toc132415812)

[Пример использования интерфейса 19](#_Toc132415813)

[Рисунок 13. Реализация программы «Использование интерфейса». 19](#_Toc132415814)

[Рисунок 14. Результат реализации программы «Использование интерфейса». 20](#_Toc132415815)

[Описание работы программы 20](#_Toc132415816)

[Пример интерфейсной переменной 21](#_Toc132415817)

[Рисунок 15. Реализация программы «Интерфейсная переменная». 21](#_Toc132415818)

[Рисунок 16. Результат реализации программы «Интерфейсная переменная». 21](#_Toc132415819)

[Описание работы программы 21](#_Toc132415820)

[**Пакеты и уровни доступа** 23](#_Toc132415821)

# **Вступление**

Наследование является одним из фундаментальных механизмов ООП. Механизм состоит в том, что на основе уже существующих классов можно создавать новые классы, которые получают, или наследуют, свойства исходных классов.

Преимущества наследования:

* обеспечивается высокая степень прямой и обратной совместимости программного кода;
* Повышается гибкость программного кода, а также снижается вероятность совершения ошибок.

# **Основы наследования**

При создании нового класса, основанного на существующем, есть два пути:

1. Внести изменения в уже существующий класс (но, старый класс мог использоваться в других проектах и это нарушит работу программ, да и увеличивается количество подобных классов, что так же отрицательно сказывается);
2. Создать класс, используя механизм наследования (позволяет выстраивать целую иерархию классов, достаточно удобную в использовании).

**Родительский класс (суперкласс)** - это класс, на основе которого создается новый класс.

**Подкласс** - это новый класс, который создается на основе суперкласса.

Для создания подкласса на основе суперкласса, в строке объявления подкласса после ключевого слова class и имени подкласса указывается ключевое слово extends и имя суперкласса.

Код подкласса описывается как в обычном классе. В подклассе автоматически известны и доступны поля и методы суперкласса (все, за исключением закрытых, с которыми мы еще не имели дела).

## Пример наследования классов

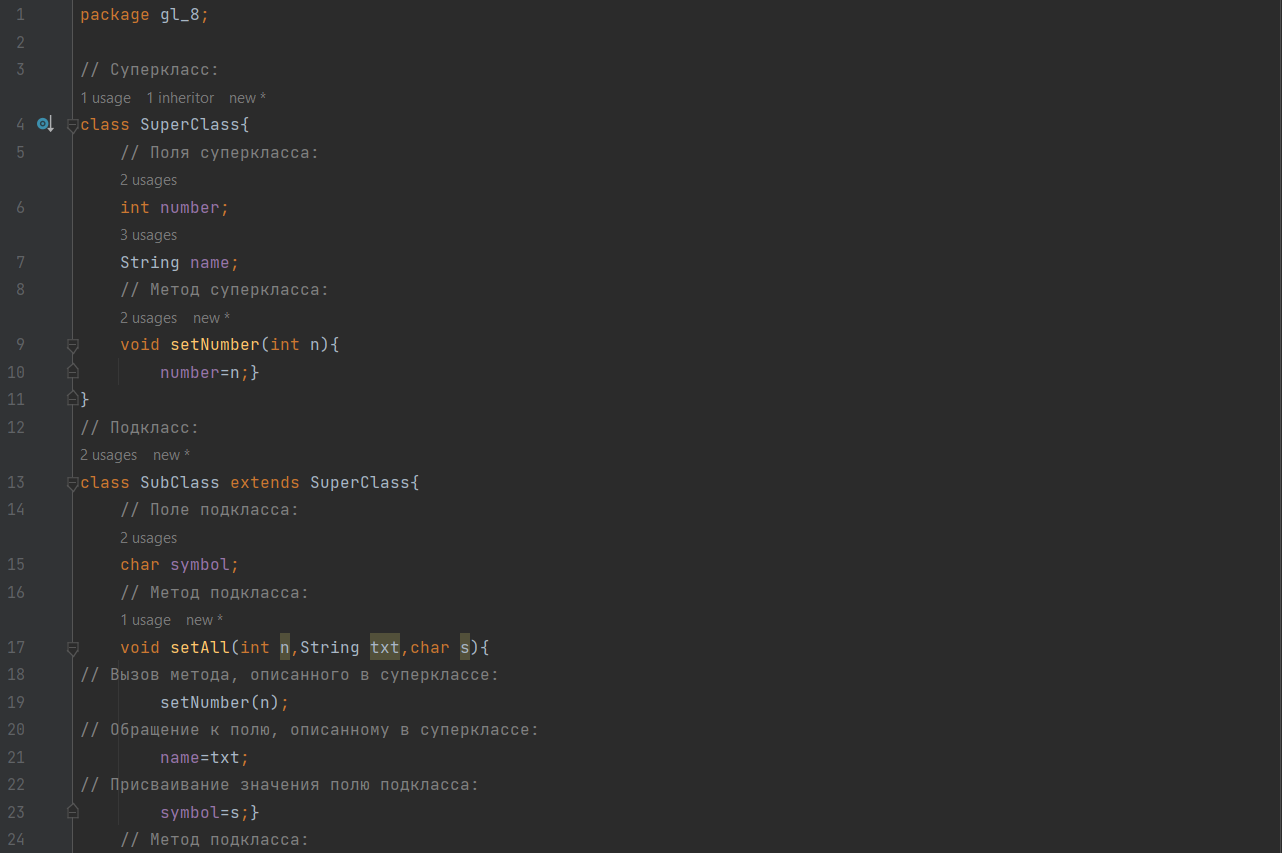
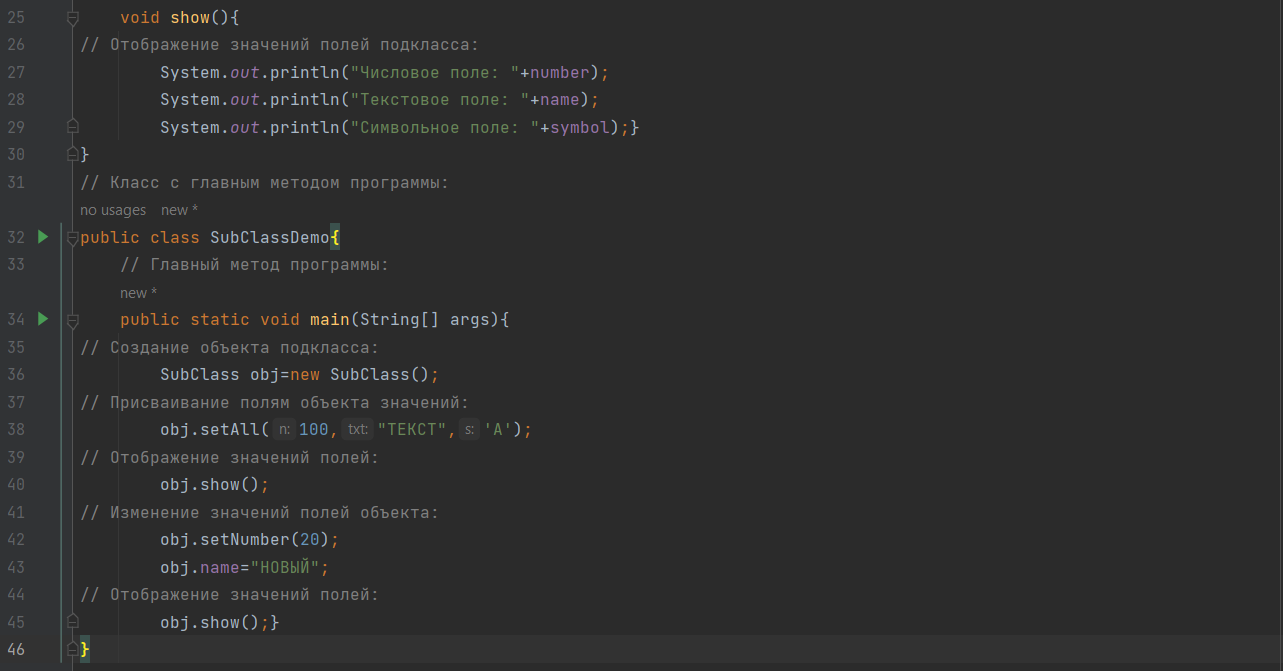
 

Рисунок 1. Реализация программы «Наследование классов».

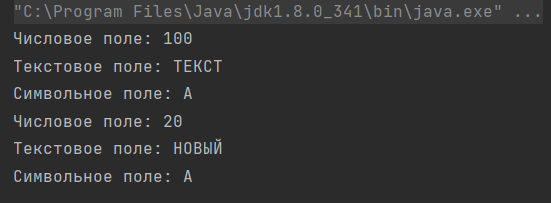


Рисунок 2. Результат реализации программы «Наследование классов».

### Описание работы программы

Сначала в программе описан класс SuperClass, на основе которого затем будет создан новый класс SubClass.

В суперклассе SuperClass описано два поля: целочисленное поле number и текстовое поле name. Также в суперклассе описан метод setNumber(), с помощью которого можно задать значение поля number.

Класс SubClass создается на основе класса SuperClass путем наследования, о чем свидетельствует инструкция extends. В теле класса SubClass описано символьное поле symbol. К унаследованным полям и методам подкласса можно обращаться как к обычным полям и методам и извне кода подкласса. Пример такого обращения можно найти в главном методе программы main(). В частности, после создания командой SubClass obj=new SubClass() объекта подкласса командой obj.setAll(100,"ТЕКСТ",'A') полям объекта присваиваются значения, а командой obj.show() выполняется вывод значений полей на консоль.

# **Конструктор подкласса**

Есть некоторая проблема, дело в том, что о при создании объекта подкласса автоматически сначала вызывается конструктор суперкласса, и уже после этого непосредственно конструктор подкласса. И в случае, если конструктору суперкласса аргументы передавать не нужно, то проблем не возникает. Но если же конструктор суперкласса должен получать аргумент или аргументы и без этого не обойтись никак, то возникает загвоздка.

Для того чтобы такой загвоздки не было, при создании конструктора подкласса следует предусмотреть способ вызова конструктора суперкласса. В код конструктора подкласса включается инструкция super(), в которых перечисляются аргументы, которые передаются конструктору суперкласса. При этом, инструкция super() должна быть первой командой в коде конструктора подкласса.

## Пример конструктора подкласса

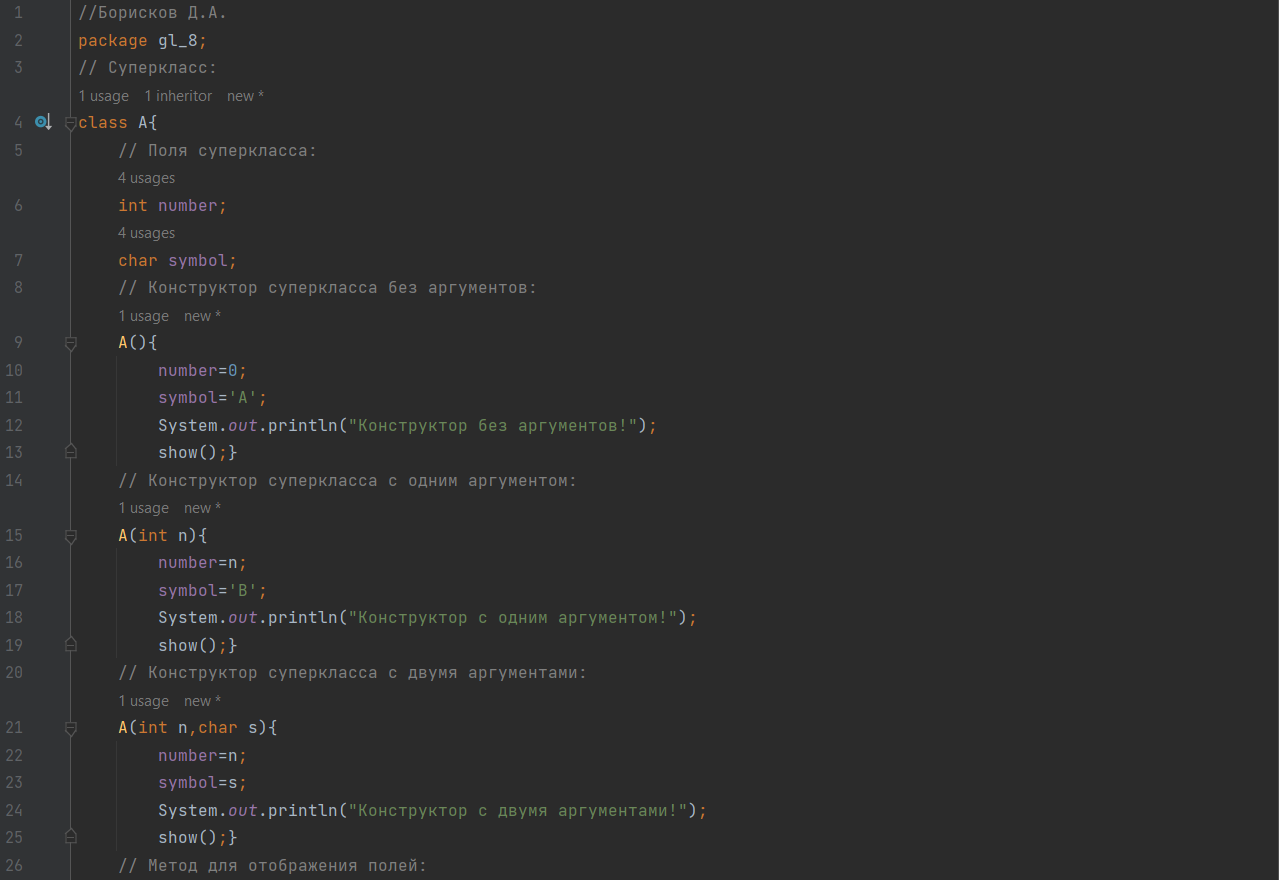
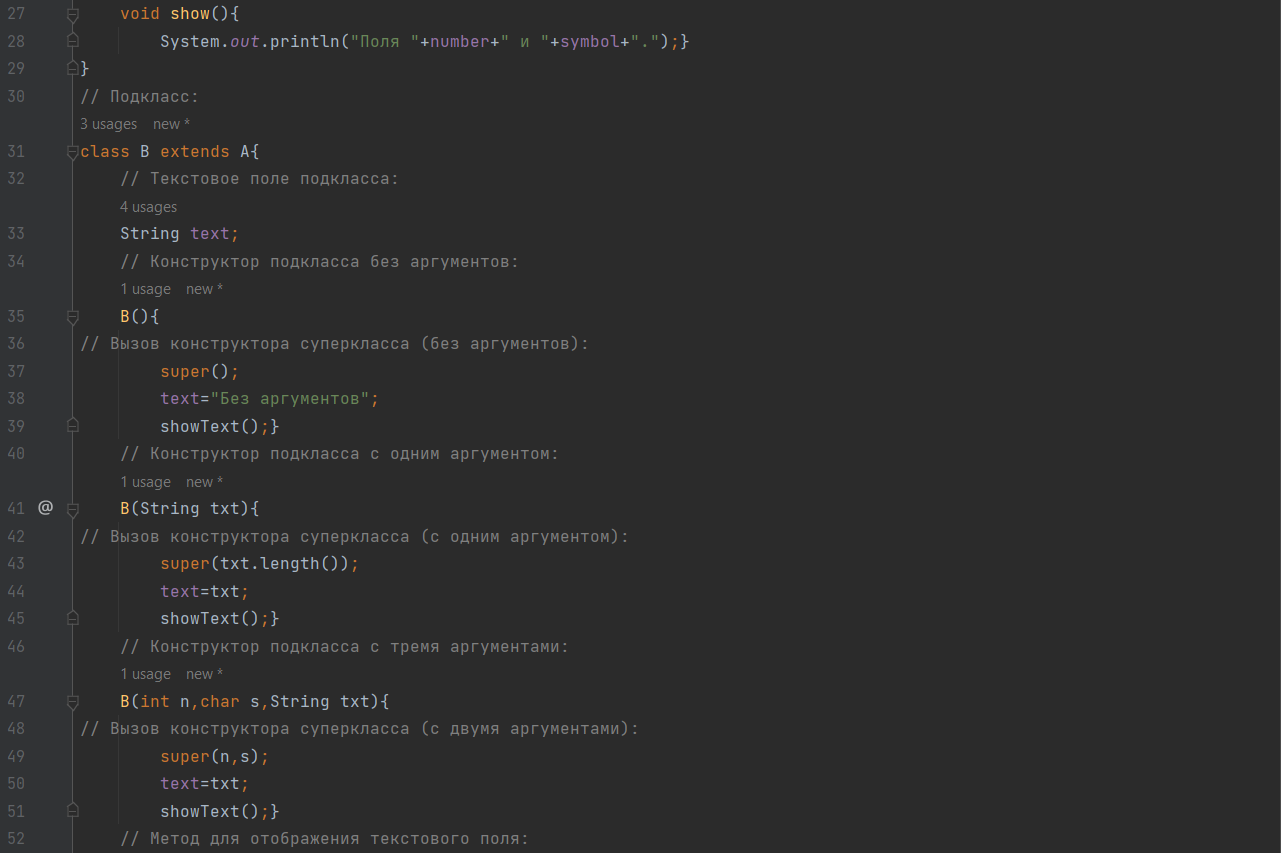
  

Рисунок 3. Реализация программы «конструктор класса».

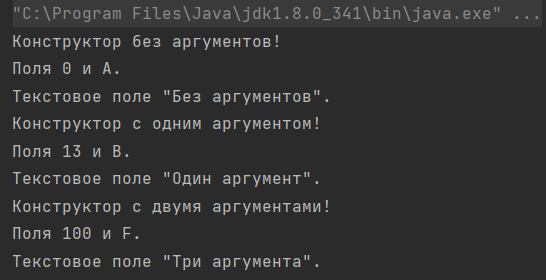


Рисунок 4. Результат реализации программы «Конструктор класса».

### Описание работы программы

В программе выполнено наследование суперкласса А подклассом В, соответственно подкласс В наследует конструктор суперкласса А.

Метод super() реализует наследование конструктора базового класса, в нём передаётся аргументы конструктора базового класса.

В классе SubConstrDemo создаются объекты дочернего класса и к ним применяются конструкторы дочернего и родительского класса. Как раз в нём наследуется конструкторы класса.

# **Переопределение методов**

Суть переопределения методов связана с тем, что программный код унаследованного в подклассе метода может быть переопределен. В результате подкласс имеет такой же метод (с тем же названием), что и суперкласс, но выполняются они по-другому.

Для переопределения метода в подклассе необходимо заново описать унаследованный метод в подклассе.

## Пример переопределения методов

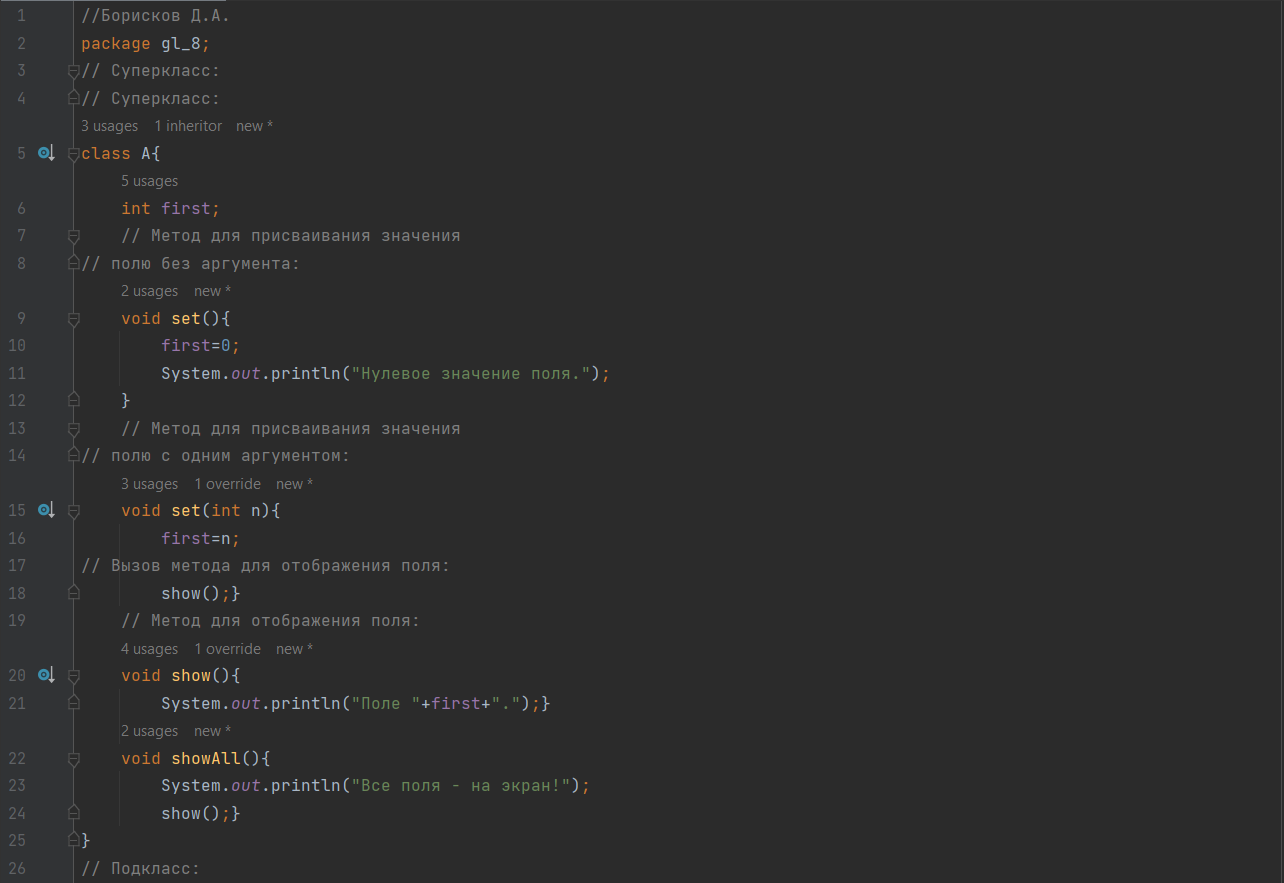
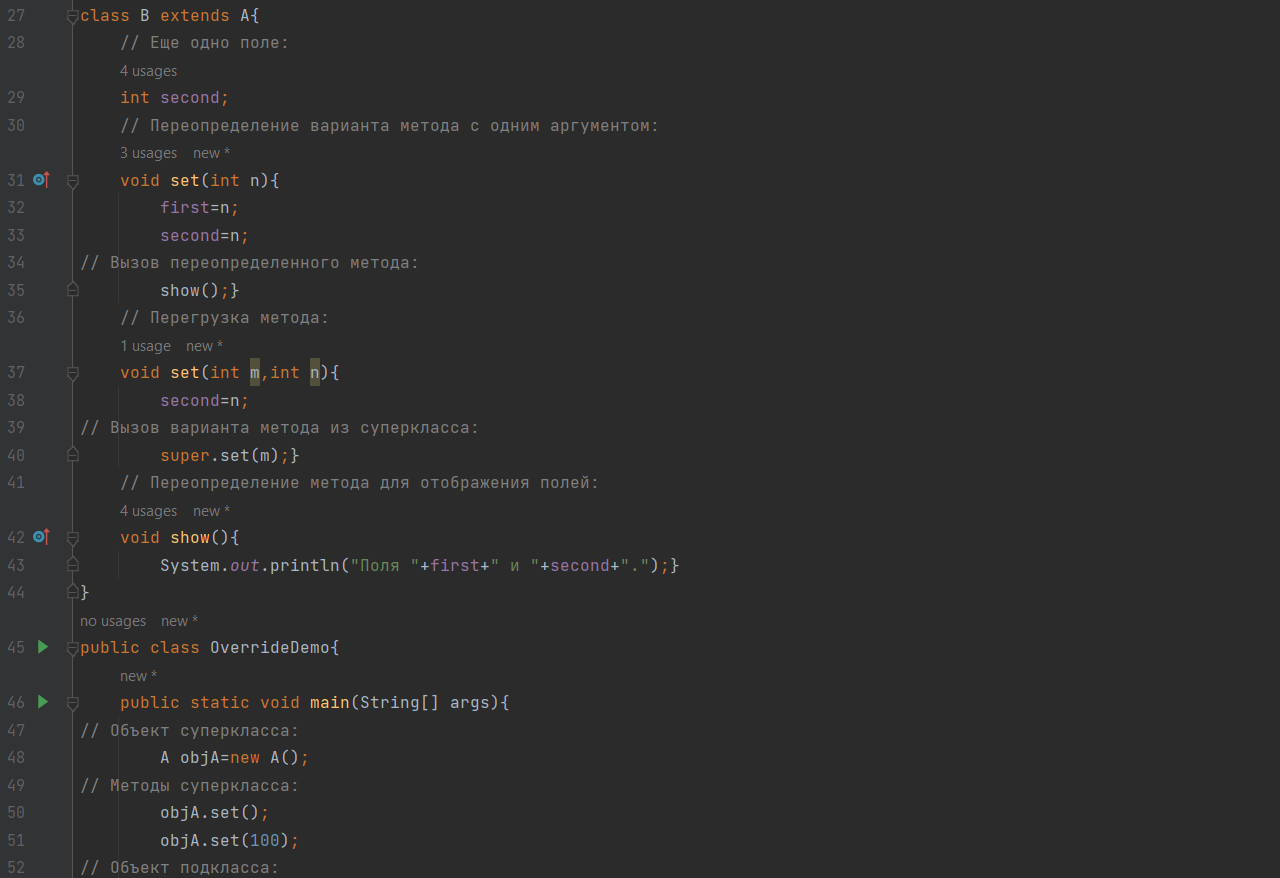
  

Рисунок 5. Реализация программы «Переопределение методов».

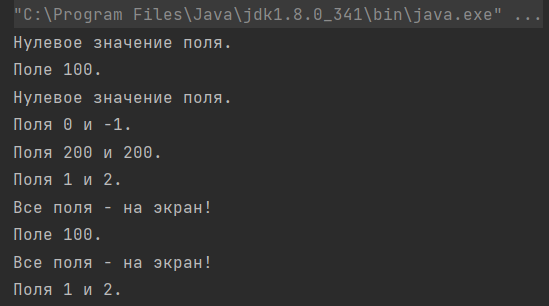


Рисунок 6. Результат реализации программы «Переопределение методов».

### Описание работы программы

В программе представлены суперкласс A и подкласс B.

У суперкласса А есть одно целочисленное поле(first), перегруженный метод set() для присваивания значения полю, метод show() для отображения значения поля и еще один метод showAll(). При помощи метода showAll() в консоль отображается текстовое сообщение "Все поля – на экран!", потом вызывается метод show().

В подклассе B добавляется еще одно числовое поле, которое называется second. В классе автоматически наследуется поле first и вариант метода set() без аргументов.

В классе OverrideDemo сначала создается объект objA класса A, после чего проверяется работа метода set() с разными аргументами. Затем создается объект objB класса B и проверяется работа методов set() и show(), но уже для класса B. Наконец, командами objA.showAll() и objB.showAll() наследуемый в классе B метод showAll() вызывается последовательно из объекта класса A и объекта класса B.

# **Закрытые члены класса**

Не все члены суперкласса наследуются.

Чтобы член класса не наследовался, его необходимо объявить в суперклассе закрытым. Для этого достаточно в описании поля или метода указать ключевое слово private.

## Пример закрытых членов класса

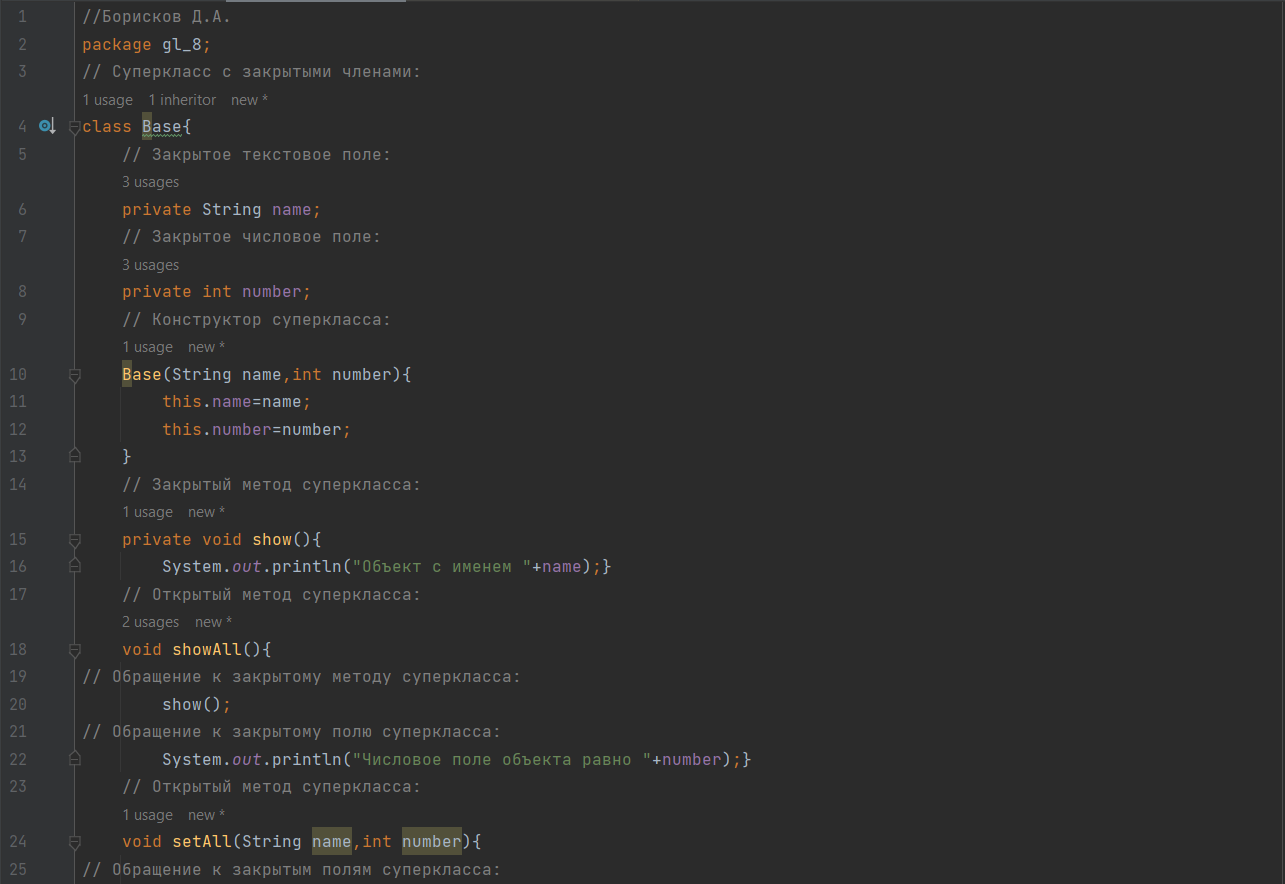
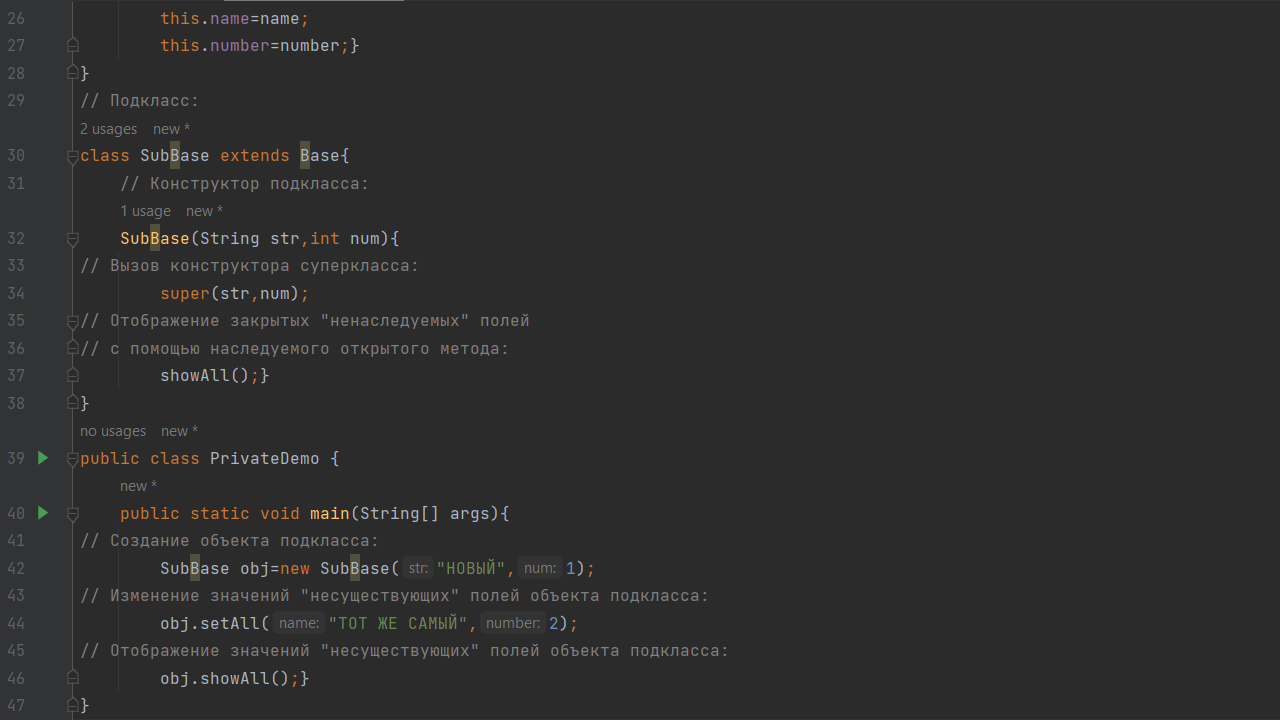
 

Рисунок 7. Реализация программы «Закрытые члены класса».

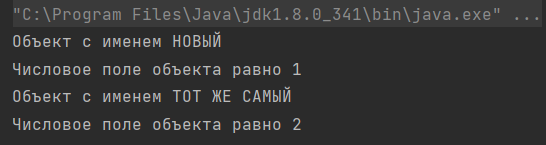


Рисунок 8. Результат реализации программы «Закрытые члены класса».

### Описание работы программы

В программе объявляется суперкласс с именем Base и затем на его основе создается подкласс с именем SubBase. В суперклассе Base объявлены два закрытых поля: текстовое name и целочисленное number. В классе Base есть три метода:

1. Метод setAll(), принимающий два аргумента, которые присваиваются в качестве значений закрытым полям объекта;
2. Методом show() на консольное окно выводится сообщение о значении текстового поля name;
3. Метод showAll() в консольное окно выводит два сообщения со значениями закрытых полей name и number.

На основе класса Base путем наследования создается его подкласс SubBase.

У конструктора подкласса SubBase есть два аргумента. Эти аргументы передаются конструктору суперкласса, который, напомним, вызывается с помощью инструкции super().

# **Объектные переменные суперклассов**

Объектные переменные суперклассов могут ссылаться на объекты подклассов. Доступ через объектную переменную суперкласса можно получить только к тем полям и методам подкласса, которые описаны в суперклассе.

## Пример объектных переменных суперклассов

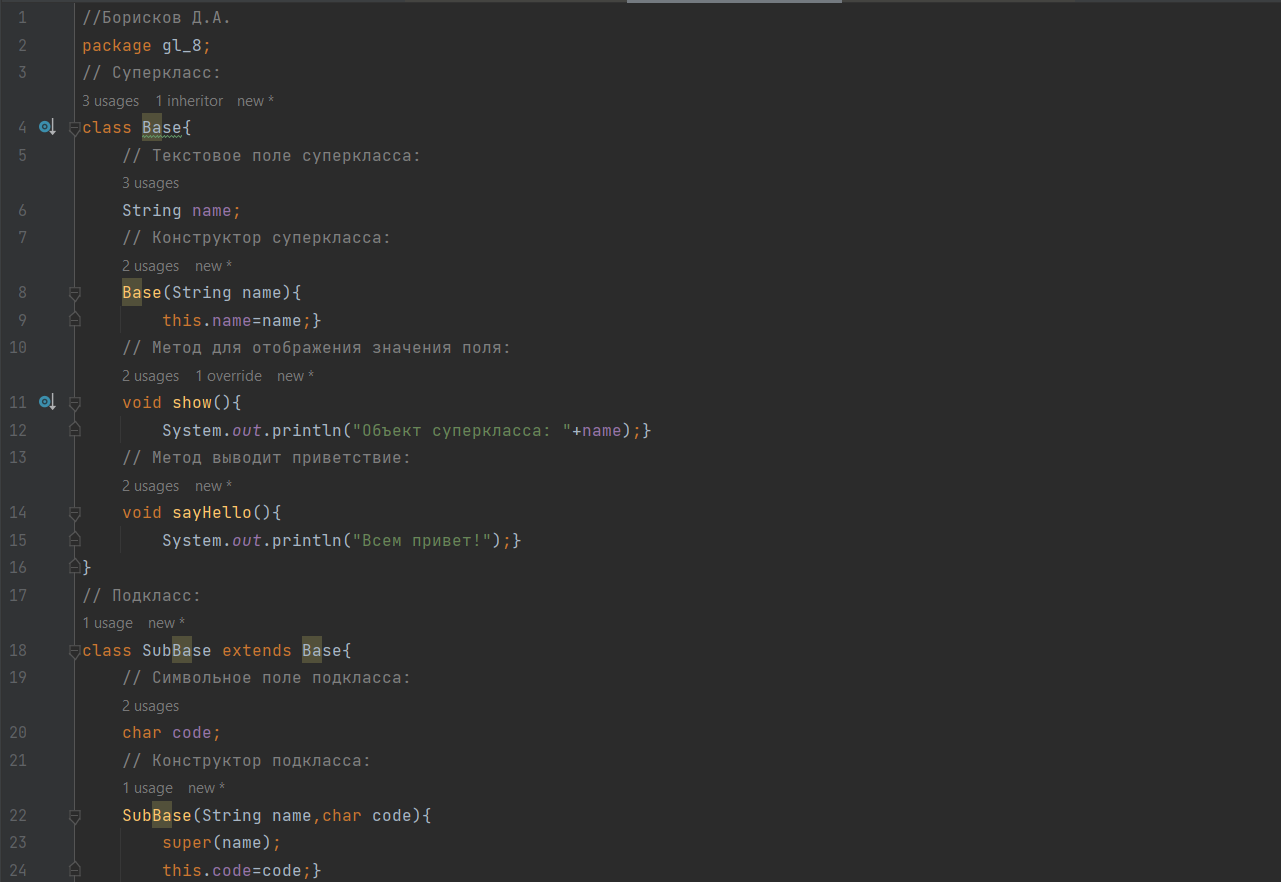
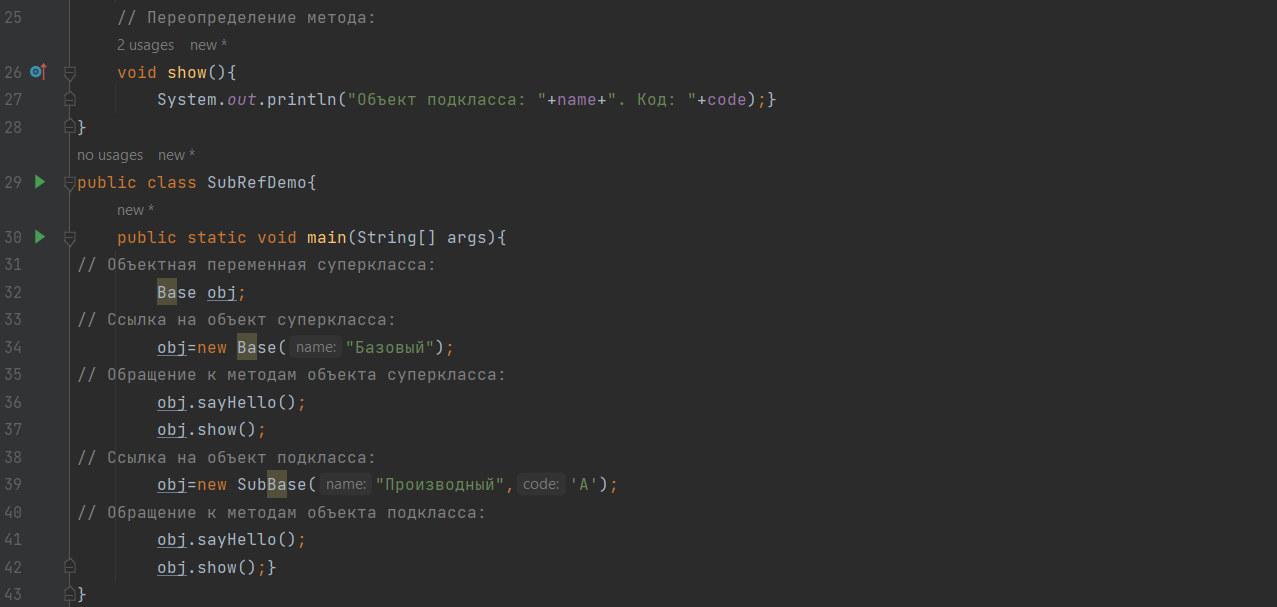
 

Рисунок 9. Реализация программы «Объектные переменные суперклассов».

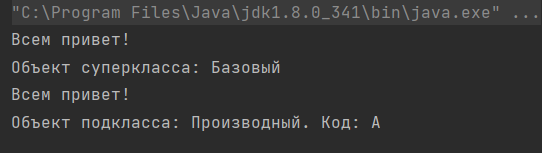


Рисунок 10. Результат реализации программы «Объектные переменные суперклассов».

### Описание работы программы

У суперкласса Base есть текстовое поле name и два метода. Методом show() в консоли отображается текстовое сообщение со значением текстового поля.

На основе суперкласса Base создается подкласс SubBase. Подкласс SubBase наследует текстовое поле name суперкласса, а также его методы sayHello() и show(), но последний в подклассе переопределяется.

В главном методе программы командой Base obj объявляется объектная переменная obj суперкласса Base. Следующей командой obj=new Base("Базовый") в качестве значения этой переменной присваивается ссылка на вновь созданный объект суперкласса со значением "Базовый" текстового поля name объекта.

Командами obj.sayHello()и obj.show() вызываются методы объекта суперкласса. В результате сначала в консольном окне появляется сообщение "Всем привет!", а затем сообщение "Объект суперкласса: Базовый".

После этого выполняется команда obj=new SubBase("Производный",'A'). Этой командой создается объект подкласса, а ссылка на него записывается в объектную переменную суперкласса. Затем мы снова выполняем уже знакомые команды obj.sayHello() и obj.show().

# **Абстрактные классы и интерфейсы**

Абстрактный метод, в отличие от обычного, имеет сигнатуру, но не содержит блока с программным кодом. Абстрактный метод объявляется с ключевым словом abstract.

Необходимость в существовании абстрактных методов отсутствует, это далеко не так. Абстрактные классы создают для того, чтобы на их основе, путем наследования, создавать другие классы.

## Пример абстрактных классов

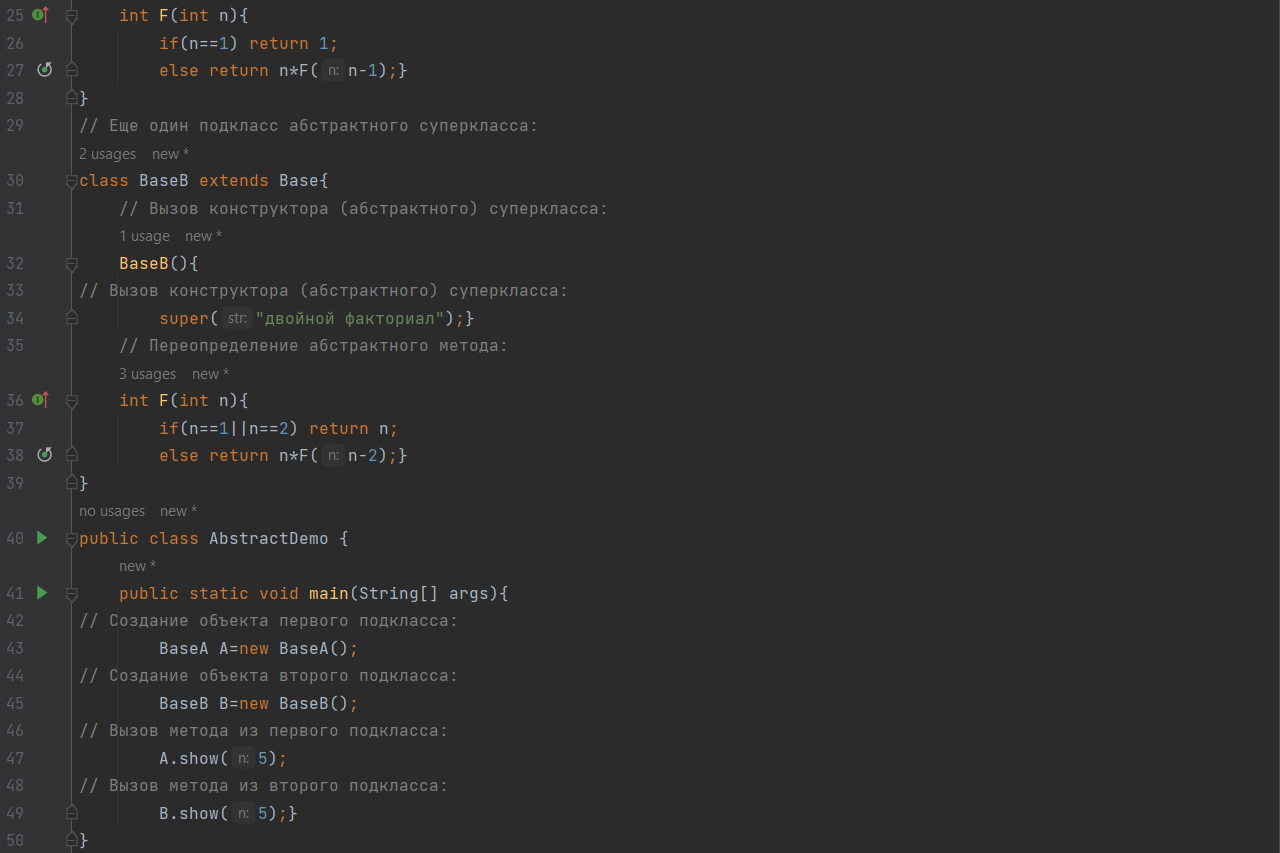
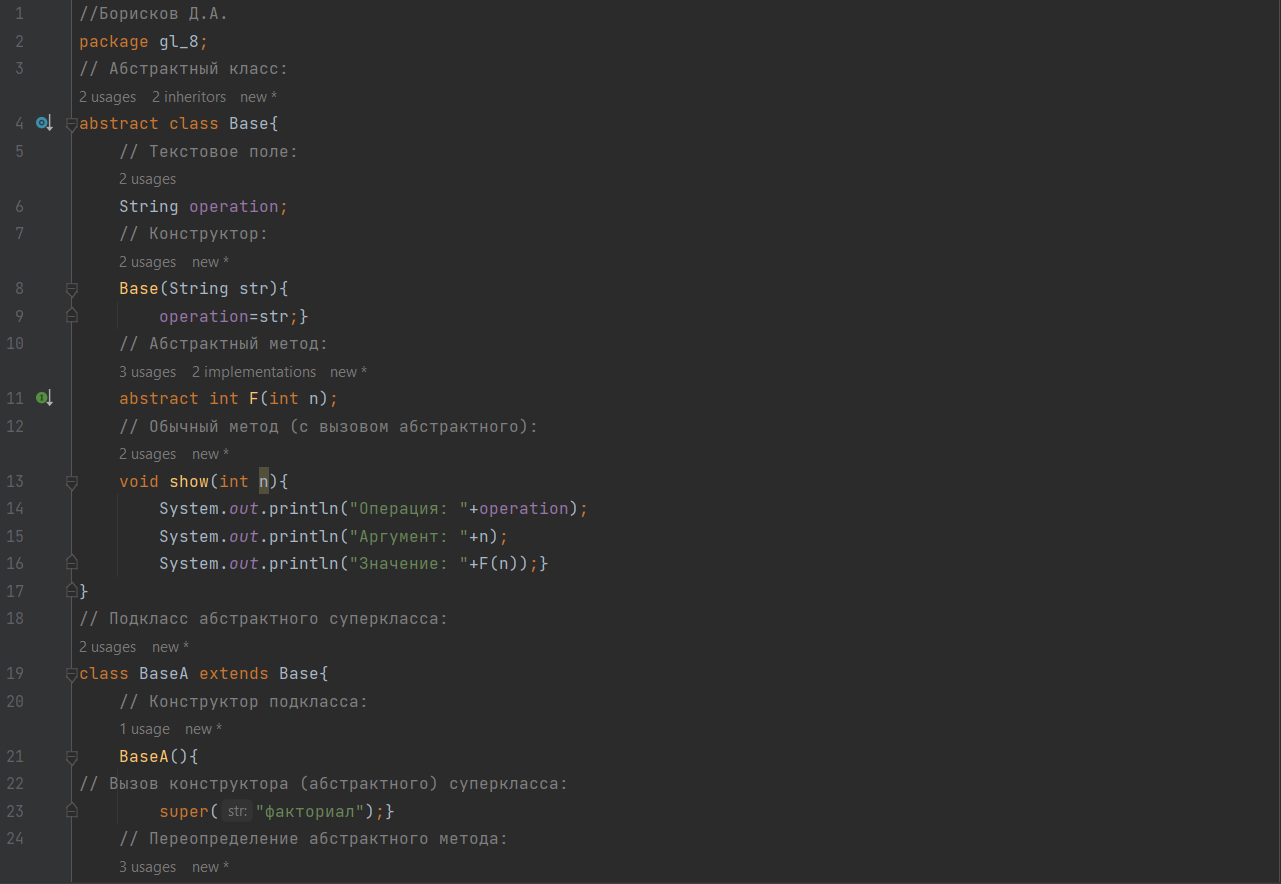


Рисунок 11. Реализация программы «Абстрактные классы».

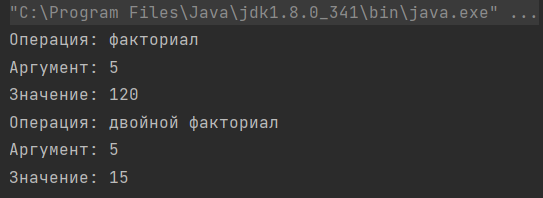


Рисунок 12. Результат реализации программы «Абстрактные классы».

### Описание работы программы

В данной программе, создаётся два подкласса BaseA и BaseB, на основе суперкласса Base.

В подклассе BaseA вызывается конструктор суперкласса с текстовым аргументом "факториал", при помощи инструкции super("факториал").

В подклассе BaseB вызывается конструктор суперкласса, с помощью инструкции super("двойной факториал").

Далее методом main() в классе AbstractDemo командами BaseA A=new BaseA() и BaseB B=new BaseB() создаются объекты подклассов BaseA и BaseB.

В завершении вызывается метод show(), такими командами, как A.show(5) и B.show(5) и данные выводятся в строку.

Продолжением идеи абстрактных классов является концепция интерфейсов. в интерфейсе методы только описываются, но не определяются (при этом в описании методов ключевое слово abstract указывать не нужно). Поля также могут присутствовать в интерфейсе, но им сразу нужно присваивать значения, и такие поля являются по умолчанию статическими и неизменными. В интерфейсе нет конструкторов.

## Пример использования интерфейса

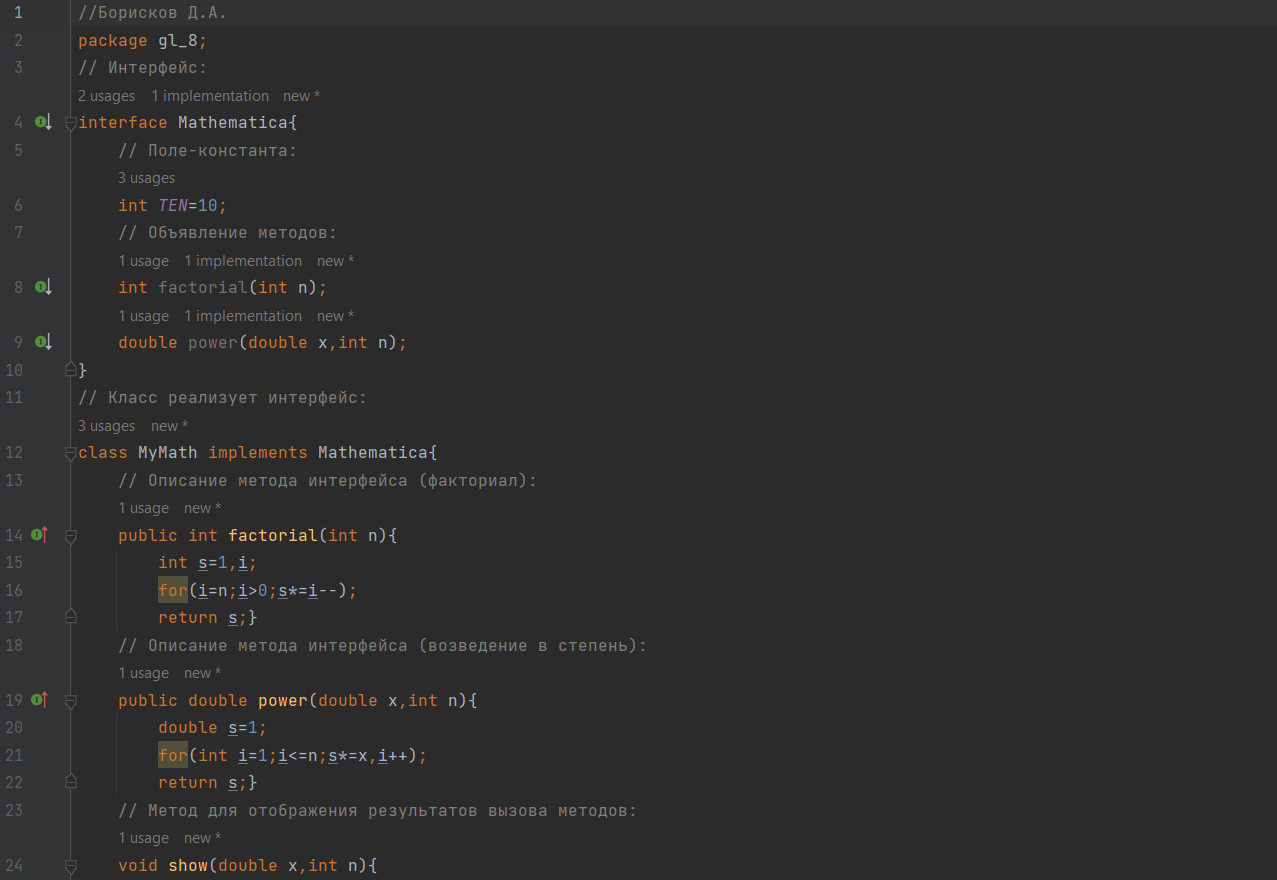
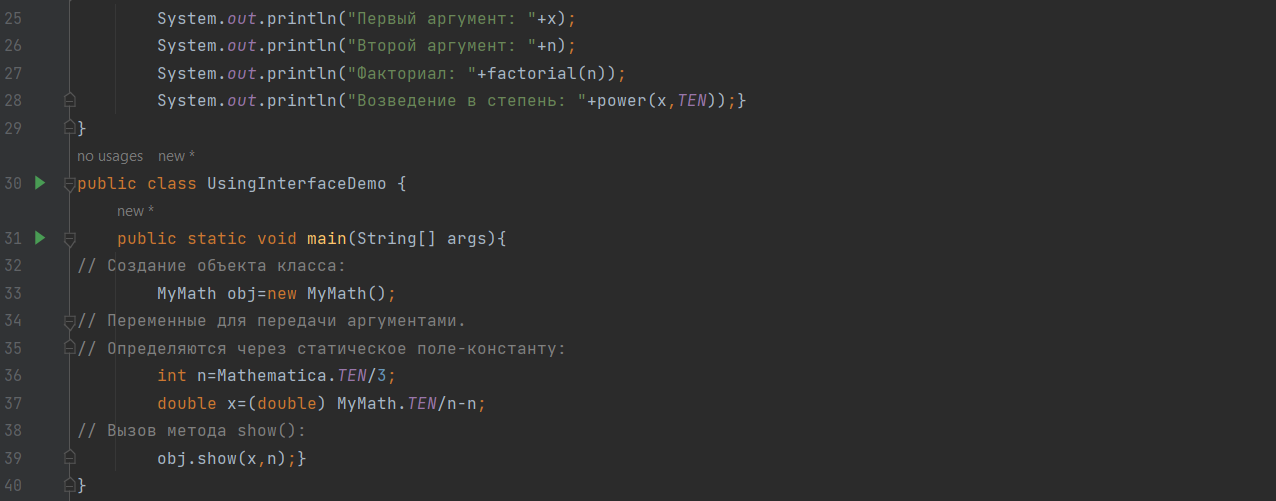
 

Рисунок 13. Реализация программы «Использование интерфейса».

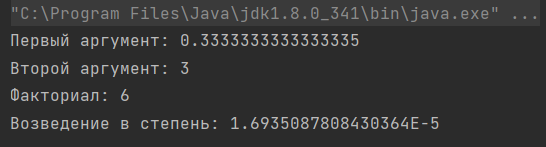


Рисунок 14. Результат реализации программы «Использование интерфейса».

### Описание работы программы

В интерфейсе Mathematica объявлена целочисленная константа TEN со значением 10 и два метода:

1. Метод factorial() имеет целочисленный аргумент и возвращает результатом значение целого типа;
2. Метод power() принимает два аргумента - первый типа double и второй типа int. Возвращается методом значение типа double.

Интерфейс Mathematica реализуется в классе MyMath.

Методом show() в консольное окно выводится несколько сообщений, содержащих информацию о том, какие аргументы передаются методу и каков результат вызова методов factorial() и power().

В главном методе программы командой MyMath obj=new MyMath() создается объект класса MyMath. Командами int n=Mathematica.TEN/3 и double x=(double)MyMath.TEN/n-n объявляются и инициализируются на основе значения статической константы TEN.

Еще одно важное свойство, связанное с интерфейсами, состоит в том, что можно создавать объектные переменные с типом интерфейса (интерфейсные переменные).

При объявлении интерфейсной переменной в качестве типа переменной указывается имя интерфейса. Для интерфейса объект создать нельзя. В качестве значения интерфейсной переменной может присваиваться ссылка на объект класса, который реализует соответствующий интерфейс.

## Пример интерфейсной переменной

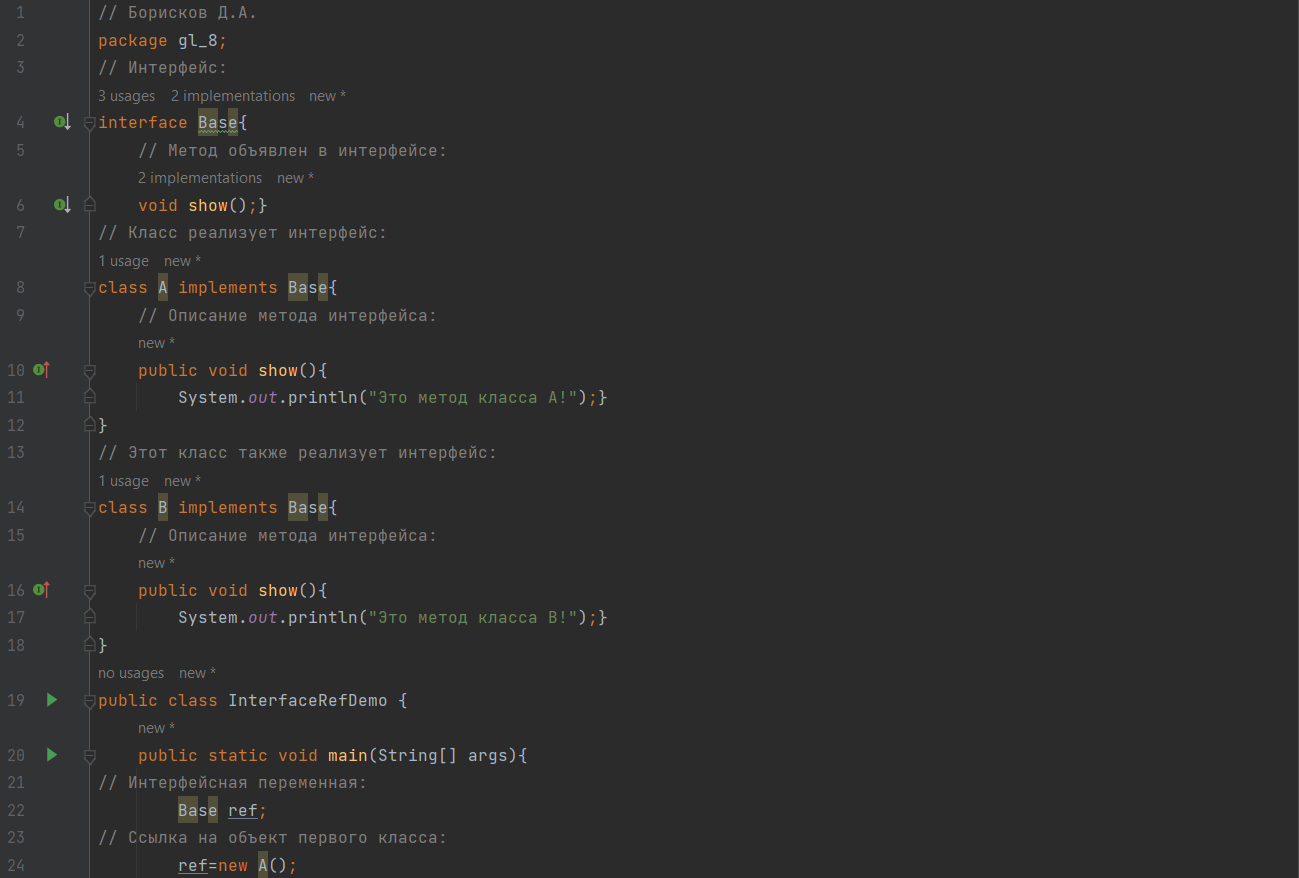
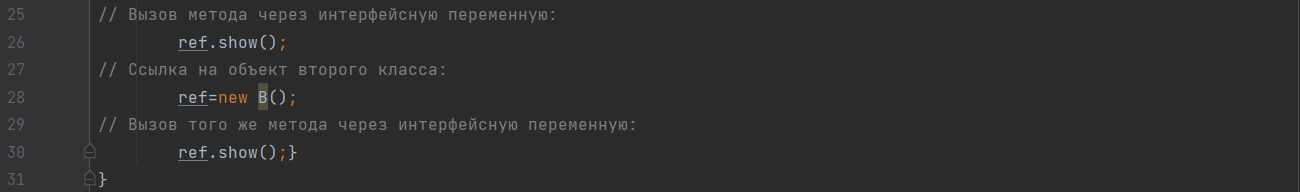
 

Рисунок 15. Реализация программы «Интерфейсная переменная».

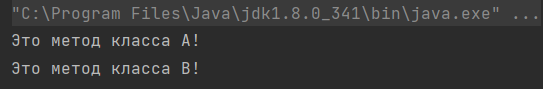


Рисунок 16. Результат реализации программы «Интерфейсная переменная».

### Описание работы программы

Интерфейс Base содержит объявление метода show(), который не принимает аргументов и не возвращает результат. В классе A метод show() описан так, что в результате вызова метода в консольное окно выводится сообщение "Это метод класса A!". В классе B метод show() определен для вывода сообщения "Это метод класса B!".

В главном методе программы командой Base ref объявляется интерфейсная переменная ref.

При вызове метода show() через интерфейсную переменную ref (команда ref.show()) вызывается версия метода, описанная в классе B.

# **Пакеты и уровни доступа**

Пакеты удобно представлять как своеобразные контейнеры, в которых хранятся классы, интерфейсы и другие пакеты (подпакеты).

В пределах пакета имена классов, интерфейсов и подпакетов уникальны. Таким образом, через пакеты формируются пространства имен.

Для того, чтобы создать пакет, необходимо в начале файла с программным кодом указать ключевое слово package и имя создаваемого пакета (в конце ставится точка с запятой). Если создается пакет подпакета, то имя подпакета указывается вместе с именем пакета (имя пакета, точка и имя подпакета).

Все классы и интерфейсы, описанные в файле с package-инструкцией, записываются в соответствующий пакет.

Файл может содержать только одну package-инструкцию, и она должна быть первой строкой в файле. Если такой инструкции нет вовсе, то описанные в классе классы и интерфейсы по умолчанию относятся к безымянному пакету.

Чтобы получить доступ к классу MyClass из подпакета subpack пакета mypack, используем такую инструкцию: import mypack.subpack.MyClass.

Чтобы получить доступ ко всем классам и интерфейсам подпакета subpack пакета mypack, используем следующую инструкцию: import mypack.subpack.\*.

Существует несколько правил, которые следует помнить и которых нужно придерживаться при работе с файлами программного кода:

1. В файле может быть только один открытый класс, то есть класс, описанный с ключевым словом public.
2. Если в файле есть открытый класс, то имя файла должно совпадать с именем этого класса.
3. Если в файле есть package-инструкция, то она указывается первой строкой кода.
4. В файле может быть несколько import-инструкций. Все они размещаются в начале файла, но после package-инструкции (если такая есть)

Чтобы запомнить, где какой член доступен, можно воспользоваться несколькими простыми правилами.

1. Члены класса, описанные с ключевым словом public, доступны везде, в том числе и во внешних классах других пакетов.
2. Члены, описанные без ключевого слова уровня доступа, доступны везде в пределах пакета, но не доступны за его пределами, вне зависимости от того, о внешнем классе или подклассе идет речь.
3. Член класса, описанный с ключевым словом private, доступен только в классе, где он описан.
4. Член класса, описанный с ключевым словом protected, доступен везде, кроме случая, когда речь идет о внешнем классе из внешнего пакета.