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**

Выполнила: студент группы:

ИБ-321

Кодоева Нели Тамазиевна

Нижний Новгород

2023 г.

Оглавление

[**Основы наследования** 3](#_Toc132483583)

[**Программа «Наследование классов»** 4](#_Toc132483584)

[**Реализация программы** 4](#_Toc132483585)

[**Объяснение работы программы «Наследование классов»** 5](#_Toc132483586)

[**Конструктор подкласса** 6](#_Toc132483587)

[**Программа «Конструктор подкласса»** 6](#_Toc132483588)

[**Реализация программы** 7](#_Toc132483589)

[**Объяснение работы программы «Конструктор подкласса»** 8](#_Toc132483590)

[**Переопределение методов** 9](#_Toc132483591)

[**Программа «Переопределение метода»** 9](#_Toc132483592)

[**Реализация программы** 10](#_Toc132483593)

[**Объяснение работы программы «Переопределение метода»** 11](#_Toc132483594)

[**Закрытые члены класса** 12](#_Toc132483595)

[**Программа «Закрытые члены класса»** 12](#_Toc132483596)

[**Реализация программы** 13](#_Toc132483597)

[**Объяснение работы программы «Закрытые члены класса»** 13](#_Toc132483598)

[**Объектные переменные суперклассов** 14](#_Toc132483599)

[**Программа «Объектные переменные суперклассов»** 14](#_Toc132483600)

[**Реализация программы** 15](#_Toc132483601)

[**Объяснение работы программы «Объектные переменные суперклассов»** 15](#_Toc132483602)

[**Абстрактные классы и интерфейсы** 16](#_Toc132483603)

[**Программа «Абстрактный класс»** 16](#_Toc132483604)

[**Реализация программы** 17](#_Toc132483605)

[**Объяснение работы программы «Абстрактный класс»** 17](#_Toc132483606)

[**Программа «Использование интерфейса»** 19](#_Toc132483607)

[**Реализация программы** 20](#_Toc132483608)

[**Объяснение работы программы «Использование интерфейса»** 20](#_Toc132483609)

[**Программа «Интерфейсная переменная»** 21](#_Toc132483610)

[**Реализация программы** 22](#_Toc132483611)

[**Объяснение работы программы «Интерфейсная переменная»** 22](#_Toc132483612)

[**Пакеты и уровни доступа** 23](#_Toc132483613)

# **Основы наследования**

Наследование – это один 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.

Данный подход имеет ряд преимуществ. В первую очередь, это то, что создание новых классов возможно без внесения изменений в уже существующие. А второе, это то, что я повышается гибкость кода и снижается вероятность совершения ошибок.

Для создания нового класса, на основе уже существующего стоит использовать механизм наследования. При создании нового класса с помощью специальных синтаксических конструкций делается указание на автоматическое включение содержимого исходного класса в новом классе. Таким образом получается выстраивать удобную в использовании иерархию классов.

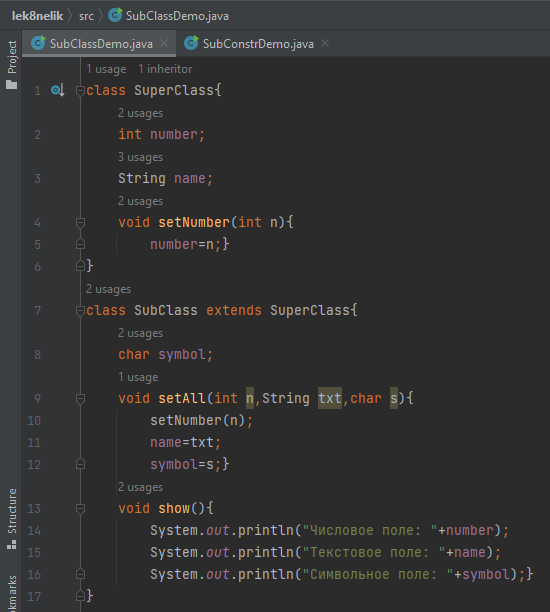
Суперкласс - это класс, на основе которого создается новый класс.

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

Для создания подкласса, в строке объявления подкласса, после ключевого слова class и имени подкласса, указывается ключевое слово extends и имя суперкласса основание которого берёт подкласс.

Наследуются те члены суперкласса, которые не являются закрытыми. Те члены классов, с которыми мы имели дело до этого, были по умолчанию открытыми.

## **Программа «Наследование классов»**



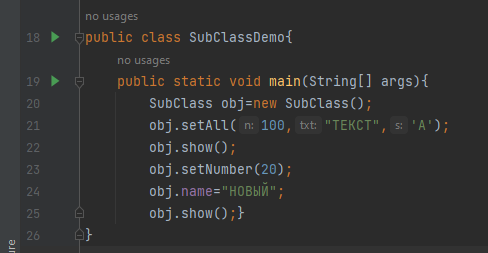


Рисунок -2 Программа «Наследование классов»

## **Реализация программы**

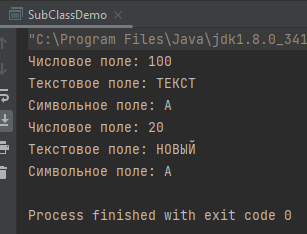


Рисунок 3 Результат программы

## **Объяснение работы программы «Наследование классов»**

В программе описан суперкласс SuperClass, на основе которого затем будет создан подкласс SubClass.

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

В теле класса SubClassописано символьное поле symbol. Так же в классе SubClass из класса SuperClass наследуются поля number, name и метод setNumber().

К унаследованным полям и методам подкласса можно обращаться как к обычным полям и методам и извне кода подкласса. 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.

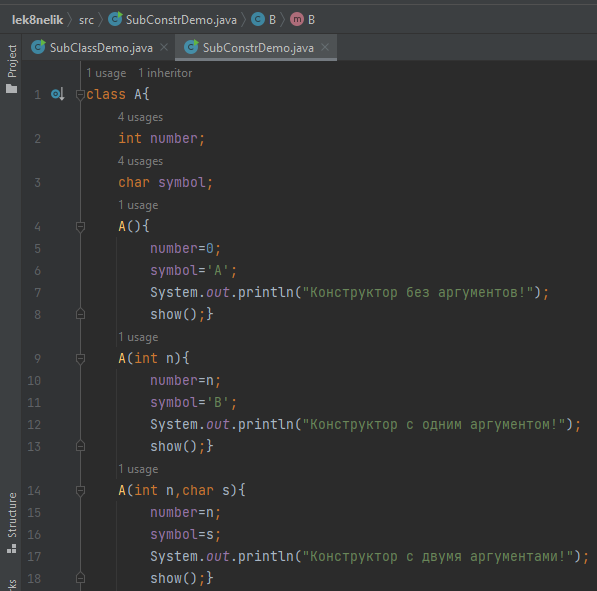
# **Конструктор подкласса**

При создании объекта подкласса автоматически сначала вызывается конструктор суперкласса, после этого непосредственно сам конструктор подкласса. И если конструктор суперкласса должен получать аргумент или аргументы и без этого не обойтись никак, то возникает проблема.

Решается она так: при создании конструктора подкласса нужно обязательно предусмотреть способ вызова конструктора суперкласса. Для этого используется инструкция super с круглыми скобками, в которых перечисляются аргументы, которые передаются конструктору суперкласса.

Инструкция super должна быть первой командой в коде конструктора подкласса.

## **Программа «Конструктор подкласса»**





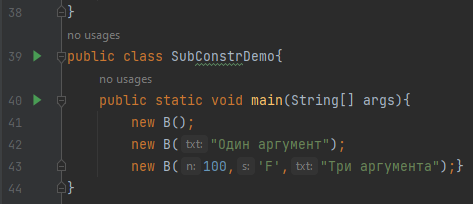


Рисунок 4-6 Программа "Конструктор класса"

## **Реализация программы**

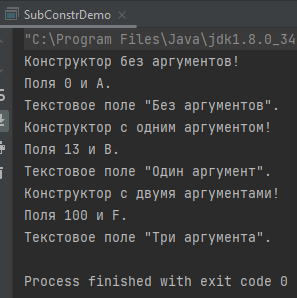


Рисунок 7 Результат программы

## **Объяснение работы программы «Конструктор подкласса»**

Суперкласс A имеет два поля: целочисленное поле number и символьное поле symbol. При вызове конструктора суперкласса, помимо присваивания значения полям, выводится консольное сообщение соответствующего содержания. Для отображения значений полей используем метод show().

В подклассе В описывается текстовое поле text, а также метод showText(), который выводит значение на консоль. В конструкторе подкласса В командой super() вызывается конструктор суперкласса. Для вычисления количества символов в текстовой переменной вызывается встроенный метод length().

В конструкторе подкласса с тремя аргументами первой командой super(n,s) вызывается конструктор суперкласса с двумя аргументами.

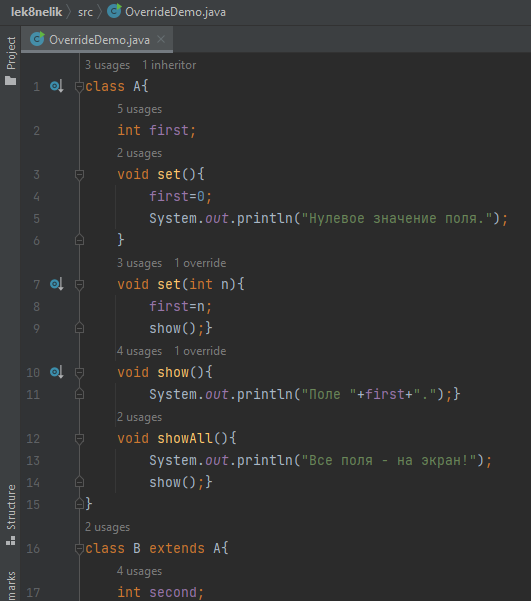
# **Переопределение методов**

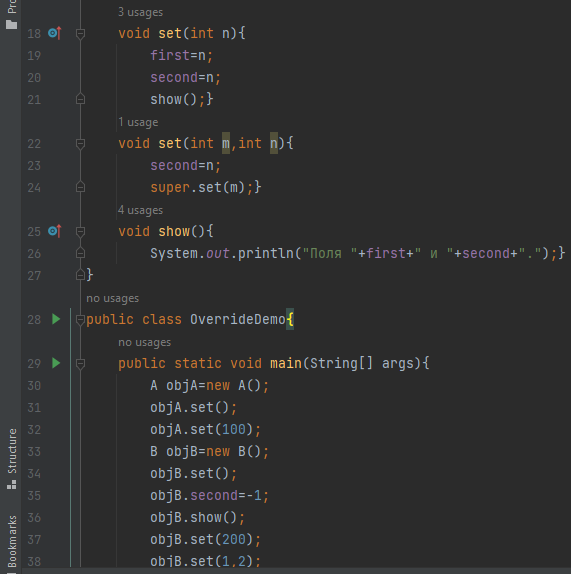
Суть переопределения методов состоит в том, что программный код унаследованного в подклассе метода может быть переопределен. В результате подкласс имеет такой же метод, что и суперкласс, но выполняются они по-разному.

При переопределении нужно, чтобы был суперкласс и подкласс. Переопределенный в подклассе имеет такую же сигнатуру, что и метод в суперклассе.

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

## **Программа «Переопределение метода»**





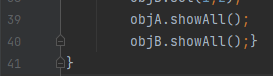


Рисунок 8-10 Программа «Переопределение метода»

## **Реализация программы**

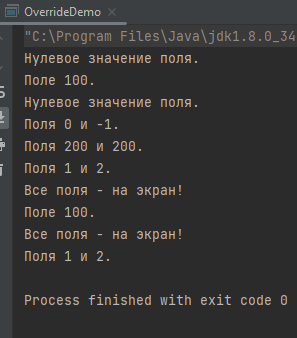


Рисунок 11 Результат программы

## **Объяснение работы программы «Переопределение метода»**

Суперкласс A имеет одно целочисленное поле, которое называется first, перегруженный метод set() для присваивания значения полю, метод show() для отображения значения поля и еще один метод showAll().

Метод showAll() отображает текстовое сообщение "Все поля – на экран!", после чего вызывается метод show().

В подклассе B добавляется числовое поле second. В классе автоматически наследуется поле first и вариант метода set() без аргументов. В классе B перегружается метод set() для случая, когда ему передаются два аргумента – значения полей first и second. Сначала присваивается значение полю second, а затем вызывается метод set() с одним аргументом.

Чтобы показать, что нам нужен старый метод set(), описанный в суперклассе, перед именем метода указываем инструкцию super.

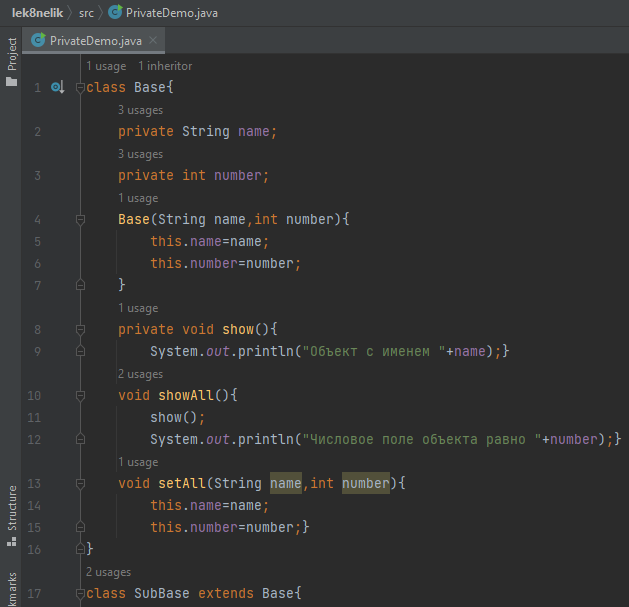
В главном методе программы в классе OverrideDemo сначала создается объект objA класса A, после чего проверяется работа метода set() с разными аргументами. командами objA.showAll() и objB.showAll() наследуемый в классе B метод showAll() вызывается последовательно из объекта класса A и объекта класса B.

# **Закрытые члены класса**

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

Члены, объявленные в суперклассе с ключевым словом private, в подклассе не наследуются.

## **Программа «Закрытые члены класса»**



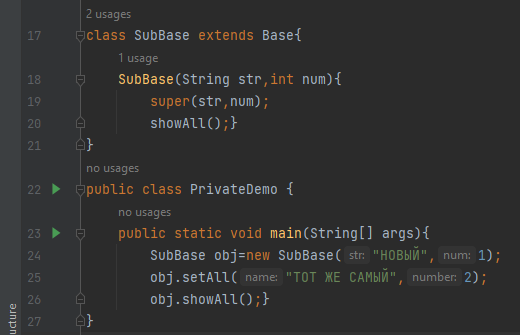


Рисунок 12-13 Программа «Закрытые члены класса»

## **Реализация программы**

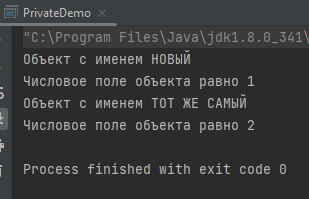


Рисунок 14 Результат программы

## **Объяснение работы программы «Закрытые члены класса»**

Суперкласс Base имеет два закрытых поля: текстовое name и целочисленное number, а также три метода (один закрытый и два открытых), которые не возвращают результат: метод setAll(), принимающий два аргумента, закрытый методод show() выводящий сообщение о значении текстового поля name и открытый метод showAll(). Этим методом в консольное окно выводятся значения закрытых полей name и number.

SubBase имеет конструктор и унаследованные из класса Base открытые методы setAll() и showAll().

В главном методе программы командой SubBase obj=new SubBase ("НОВЫЙ",1) создается объект подкласса. Аргументы, переданные конструктору подкласса, предназначены для "ненаследуемых" из суперкласса полей. obj.setAll выполняет изменение значений "несуществующих" полей.

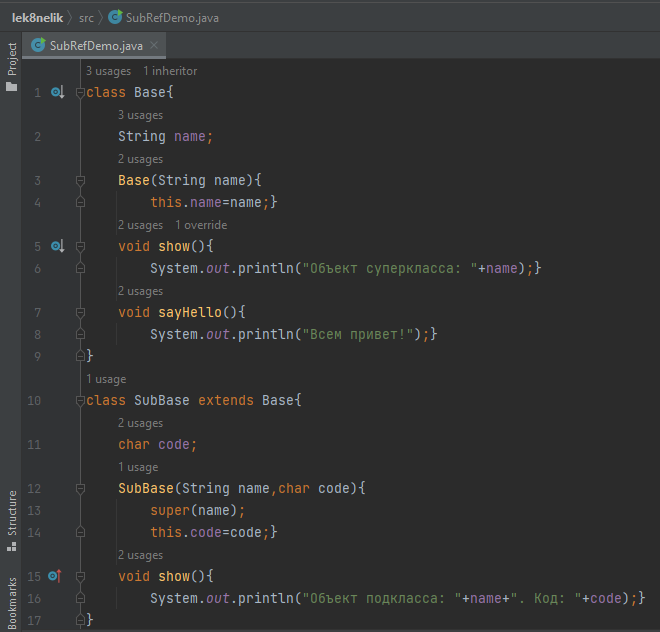
Отображение значений "несуществующих" полей выполняется командой obj.

# **Объектные переменные суперклассов**

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.

## **Программа «Объектные переменные суперклассов»**



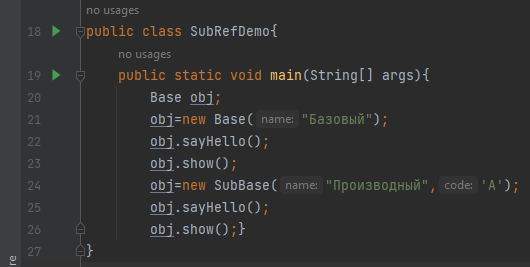


Рисунок 15-16 Программа «Объектные переменные суперклассов»

## **Реализация программы**

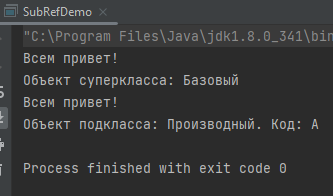


Рисунок 17 Результат программы

## **Объяснение работы программы «Объектные переменные суперклассов»**

У суперкласса Base есть текстовое поле name и два метода:

1. Методом show() в консоли отображается текстовое сообщение со значением текстового поля;
2. Методом sayHello() отображается простое текстовое сообщение.

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

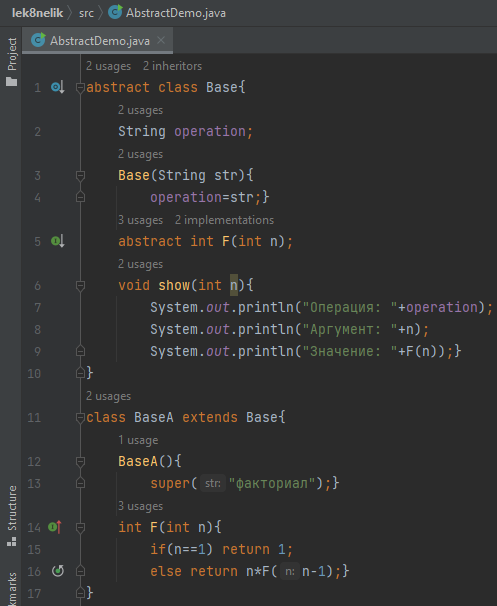
# **Абстрактные классы и интерфейсы**

Абстрактный метод имеет сигнатуру, но не содержит блока с программным кодом и объявляется с ключевым словом abstract.

Для абстрактного класса не может быть создан экземпляр класса, то есть объект.

Абстрактные классы создают для того, чтобы на их основе, путем наследования, создавать другие классы.

## **Программа «Абстрактный класс»**



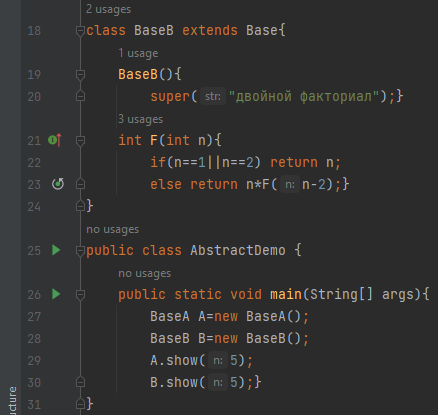


Рисунок 18-19 Программа «Абстрактный класс»

## **Реализация программы**

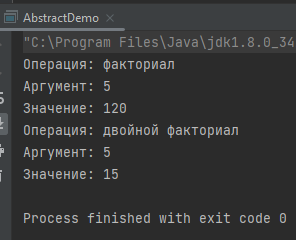


Рисунок 0 Результат программы

## **Объяснение работы программы «Абстрактный класс»**

Объявляется абстрактный класс Base, в нём есть текстовое поле operation и конструктор с одним текстовым аргументом. Команда abstract int F(int n) объявляет абстрактный метод. Метод F(), возвращает значение типа int, и у него один аргумент также типа int. Метод show() не возвращает результат, и у него один целочисленный аргумент. Методом в консольное окно выводится три сообщения.

На основе класса Base через наследование создается два разных подкласса BaseA и BaseB.

В классе BaseA инструкцией super("факториал") вызывается конструктор суперкласса с текстовым аргументом "факториал".

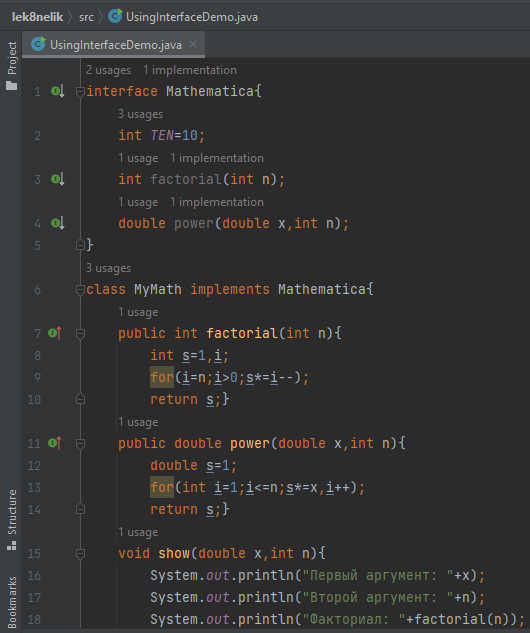
В подклассе BaseB описан конструктор без аргумента, в котором инструкцией super("двойной факториал") вызывается конструктор суперкласса. Также в классе переопределяется метод F().

В методе main() в классе AbstractDemo командами BaseA A=new BaseA() и BaseB B=new BaseB() создаются объекты подклассов BaseAи BaseB соответственно.

Если в классе описать не все методы интерфейса, то этот класс будет абстрактным и его следует объявлять с ключевым словом abstract. Если хотя бы один из этих абстрактных методов в классе не описан, класс будет содержать абстрактный метод.

Для реализации интерфейса в классе при описании класса в заголовке, после имени класса необходимо указать ключевое слово implements и имя интерфейса.

## **Программа «Использование интерфейса»**



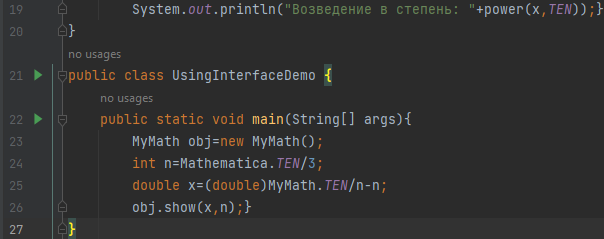


Рисунок 21-22 Программа «Использование интерфейса»

## **Реализация программы**

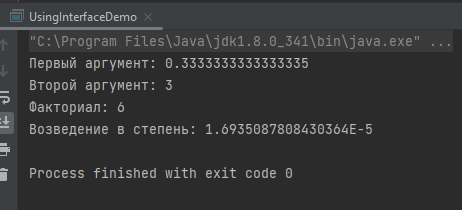


Рисунок 2 Результат программы

## **Объяснение работы программы «Использование интерфейса»**

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

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

В классе MyMath описываются методы factorial() и power() так, что методом factorial() вычисляется факториал для целого числа, переданного арументом методу, а метод power(), возводит первый аргумент в целочисленную степень, переданную через второй аргумент.

Метод show() выводит в консольное окно информацию о том, какие аргументы передаются методу и результат вызова методов factorial() и power().

В главном методе программы командами int n=Mathematica.TEN/3и double x=(double)MyMath.TEN/n-n объявляются и инициализируются на основе значения статической константы TEN, которые затем в коман де obj.show(x,n) передаются аргументами методу show().

У интерфейсов есть несколько важных особенностей. Один интерфейс может наследовать другой интерфейс (используется все то же ключевое слово extends).

Расширение интерфейса - процедура наследования одного интерфейса другим.

Интерфейс может наследовать только один интерфейс. Зато класс может реализовать сразу несколько интерфейсов.

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.

## **Программа «Интерфейсная переменная»**

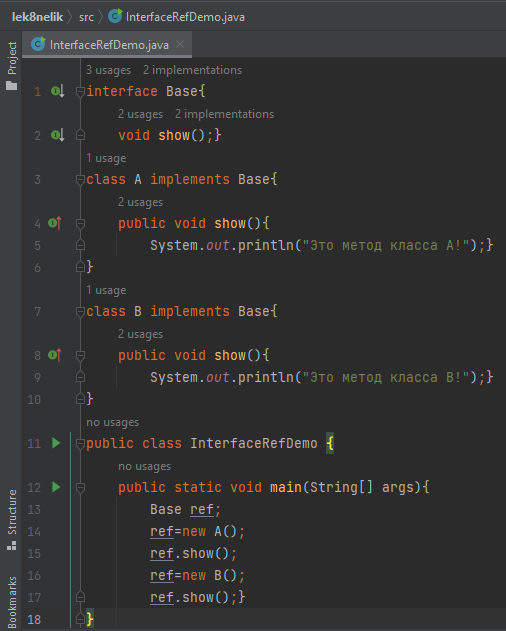


Рисунок 2 Программа «Интерфейсная переменная»

## **Реализация программы**

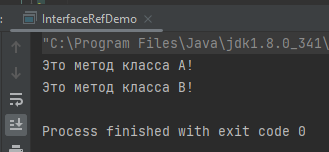


Рисунок 25 Результат программы

## **Объяснение работы программы «Интерфейсная переменная»**

Интерфейс Base содержит объявление метода show(), который не принимает аргументы и не возвращает результат.

В классе A метод show() выводит сообщение "Это метод класса A!" в консольное окно. 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. В результате появляется сообщение "Это метод класса B!".

# **Пакеты и уровни доступа**

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

В пределах пакета имена классов, интерфейсов и подпакетов уникальны.

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

Имя глубоко спрятанных внутренних подпакетов указывается в package-инструкции указывается вместе со всеми внешними пакетамиконтейнерами с использованием точки в качестве разделителя.

Для использования файлов и интерфейсов из внешнего пакета используют инструкцию import.

Инструкцию import можно и не указывать. В этом случае при использовании классов из внешних пакетов придется указывать так называемое полное имя класса.

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.