Встреча с Джа-Джа

Убежав от армии Торговой Федерации на планете Набу, вы сталкиваетесь с местным аборигеном Джа-Джа Бинксом, который предлагает свою помощь в постижении Силы языка Java. На вопрос джедая Квай-Гона чему он может научить, Джа-Джа отвечает, что умеет заставлять компьютер говорить. «Способность говорить, еще не признак опытного программиста» - говорит Квай-Гон, но его ученик Оби-Ван Кеноби уговаривает принять помощь от местного аборигена.

Итак, первое, с чем сталкиваются во всех современных языках программирования это команда вывода текста на экран. Напишите программу, чтобы она выводила текст на экран: «Mesa called Jar Jar Binks, mesa your humble servant!».

public class Main {  
    public static void main(String[] args) {  
        //напишите тут ваш код  
    }  
}

Требования к выполнению задания

* Программа должна выводить текст.
* Выводимый текст должен соответствовать заданию.

Справочная информация

Давайте разберемся с тем, что у нас есть для дальнейшего движения по материалу.

Java - объектно-ориентированный язык, поэтому такие понятия как "класс" и "объект" являются ключевыми. Все программы в Java состоят из набора объектов. Класс является описанием объекта, а объект - экземпляром данного класса.

Например, у нас есть представитель расы гунганов, абориген Джа-Джа. В данном случае гунганы являются классом, который описывает особенности всех представителей расы:

class Gungans {  
}

Класс объявляется служебным словом class, затем записывается имя данного класса (какое мы захотим), а после, ставятся фигурные скобки - это границы класса, внутри которых будут описаны объекты.

Сам же Джа-Джа будет объектом данного класса, т.к. он является представителем данной расы, а значит обладает соответствующим набором свойств, определяющих его как гунгана(цвет кожи, длинные уши, большие глаза и т.д.):

class Gungans {  
    String name = "Jar-Jar";  
      
    void display\_info(){  
        System.out.printf("Name: %s", name);  
    }  
}

Состояние объекта содержат переменные, а его поведение - функции (еще их называют методами). Состоянием нашего объекта из примера выше является переменная строкового типа (String) name. Служебным словом String мы объявили, что наша переменная name принадлежит строковому типу данных, а с помощью символа "=" присвоили ей значение и поставили необходимый элемент ";" говорящий о конце данной строчки.  
Ниже мы написали метод класса display\_info(), внутри которого записали команду, чтобы при вызове данного метода он выводил на экран имя нашего персонажа.

Классы могут не содержать переменных и методов, но минимальное требование для работы программы - программа должна состоять минимум из одного класса, который должен содержать минимум один метод. Имя этого класса должно совпадать с именем файла-программы. В нашем случае следующий код будет минимальной программой:

public class Main {  
    public static void main(String[] args) {  
    }  
}

Файл в данном случае должен называться по имени класса - Main. При запуске на исполнения вашего файла, Java будет обращаться к классу отталкиваясь от имени этого файла, соответственно, если такого класса не будет найдено, то машине не будет понятно где ей искать инструкции.  
Важный момент - public static void main(String[] args){} - это главный и неизменный метод для исполнения программы. Таким образом, программа всегда начинает искать инструкции, которые написаны внутри класса с соответствующим именем и внутри этого базвого метода. Методы, как и классы, могут содержать переменные и подфункции, поэтому их границы тоже обозначаются фигурными скобками.

В таком виде наша программа ничего не будет делать, поэтому внутри метода main нам необходимо добавить некоторую инструкцию. Давайте это будет команда на вывод информации, о том, что программа работает:

public class Main {  
    public static void main(String[] args) {  
        System.out.print("Моя первая программа и она работает!");  
    }  
}

Команда System.out.print() выводит информацию на экран, которая расположена внутри круглых скобок. Эта информация может быть разных типов, как строковые, так и, например, целочисленные (int) или дробные (double). Если мы попробуем записать несколько таких команд подряд, то мы получим результат, который будет выведен в одну строчку (попробуйте сами вызвать несколько System.out.print() внутри базового метода и убедитесь в этом). Для этого можно использовать вариацию данной команды: System.out.println(); - данная команда ставит невидимый элемент переноса строки ("\n"), тем самым позволяет выводить информацию на отдельной строке:

public class Main {  
    public static void main(String[] args) {  
        System.out.println("Первая строка");  
        System.out.println("Вторая строка");  
    }  
}

Можно самостоятельно использовать элемент переноса строки:

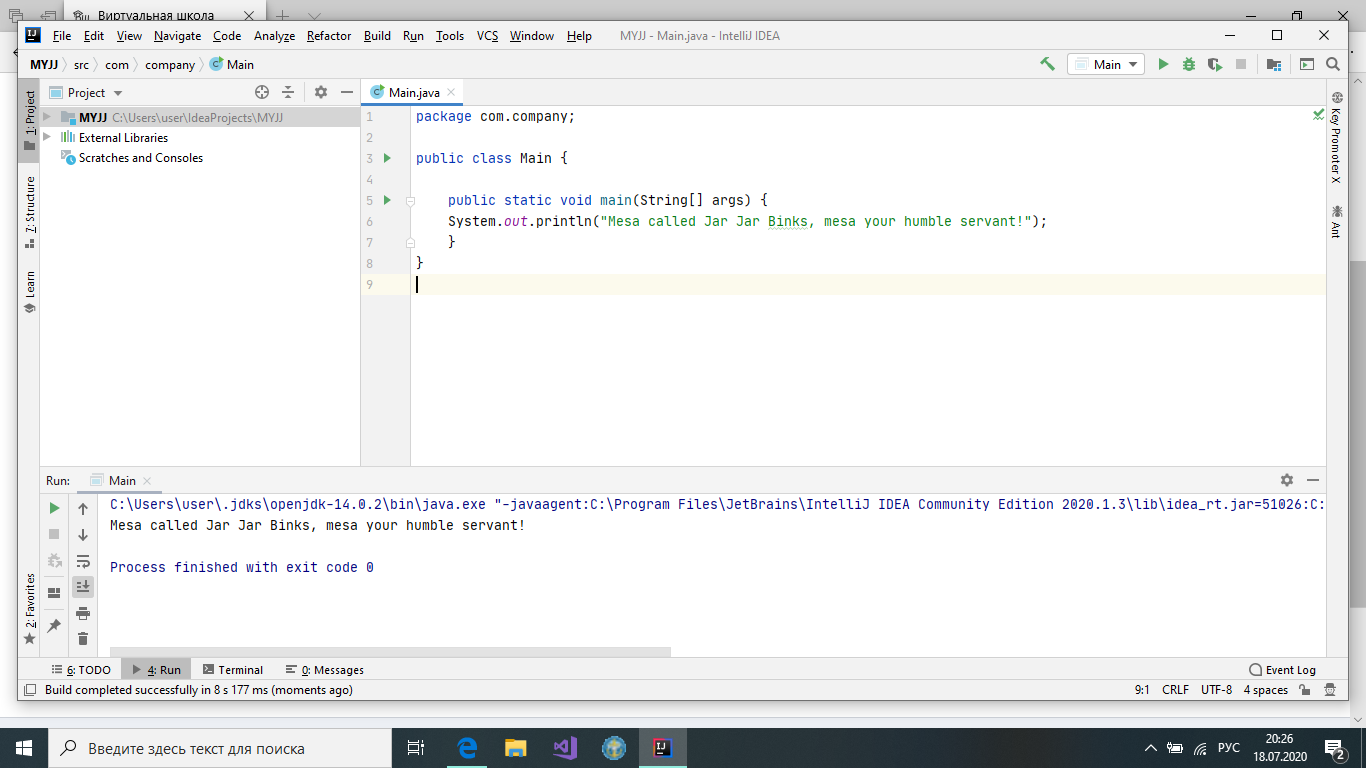
public class Main {  
    public static void main(String[] args) {  
        System.out.print("Первая строка\n" + "Вторая строка\n");  
    }  
}

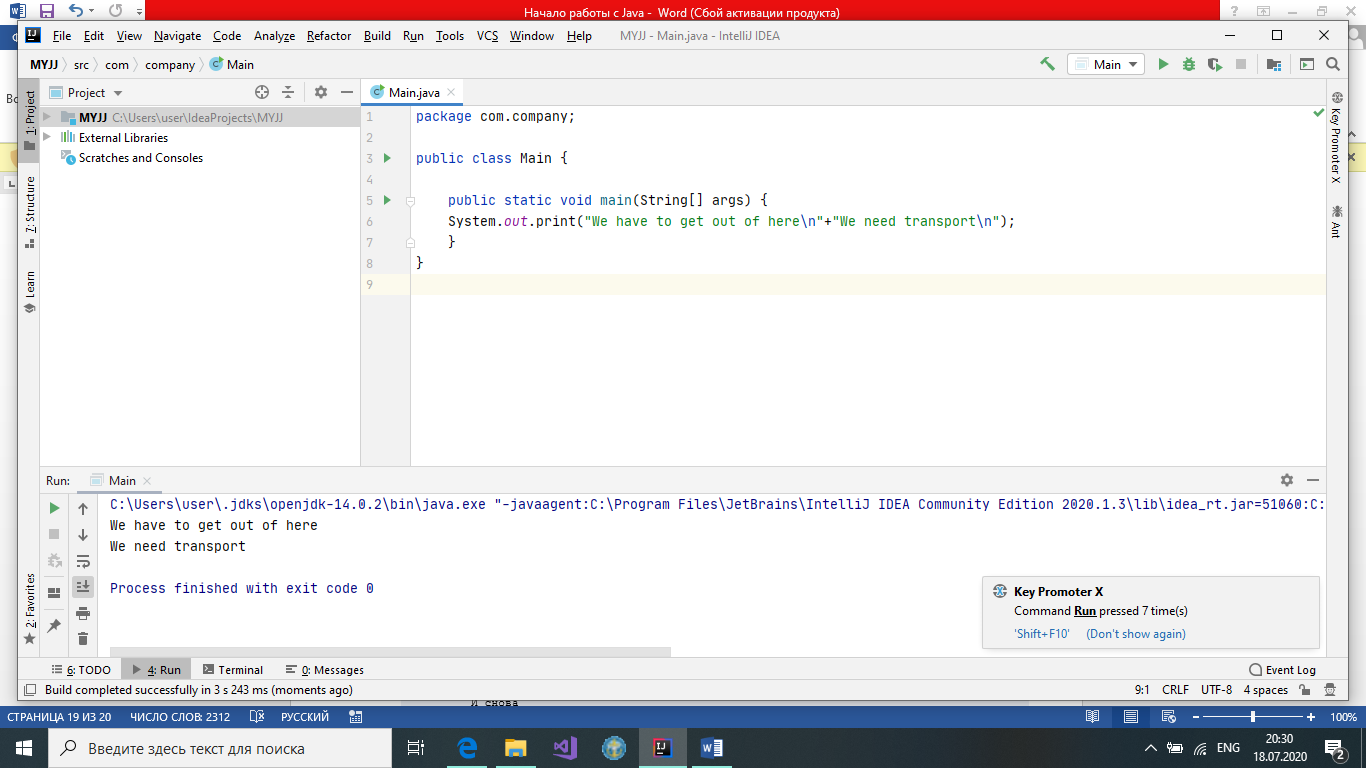
На случай, если нам необходимо закомментировать строки, или написать внутри кода какую-то служебную информацию для себя, можно использовать символы "//" или "/\*\*/". Обычные два слэша нужны для закомментирования одной строчки, слэши со свездочкой - для многострочных комментариев. Закомментированные строки не обрабатываются при исполнении программы и никакие инструкции не будут исполняться.

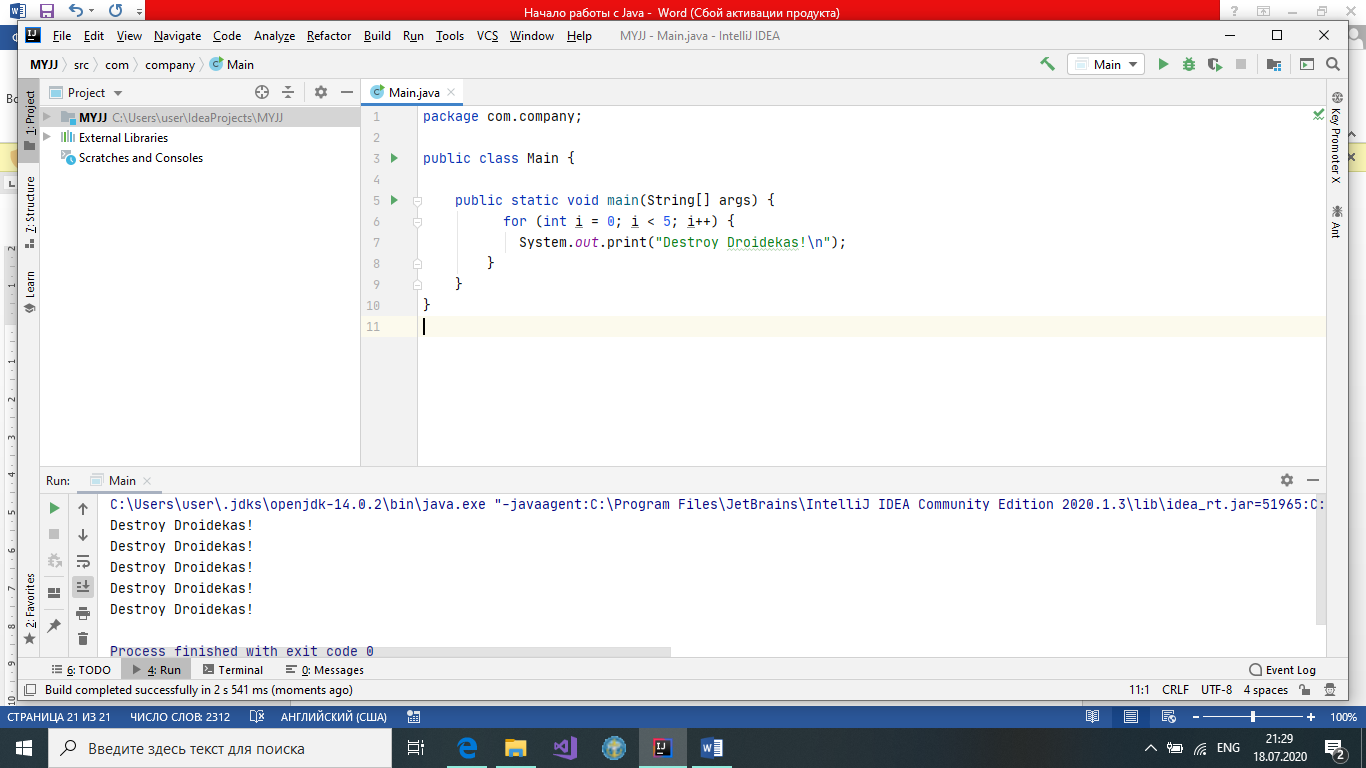
public class Main {  
    public static void main(String[] args) {  
        //System.out.println("Первая строка");  
        /\*System.out.println("Вторая строка");  
          Продолжаем комментировать строки  
          И снова  
          И снова  
          пожалуй тут закончим\*/  
    }  
}

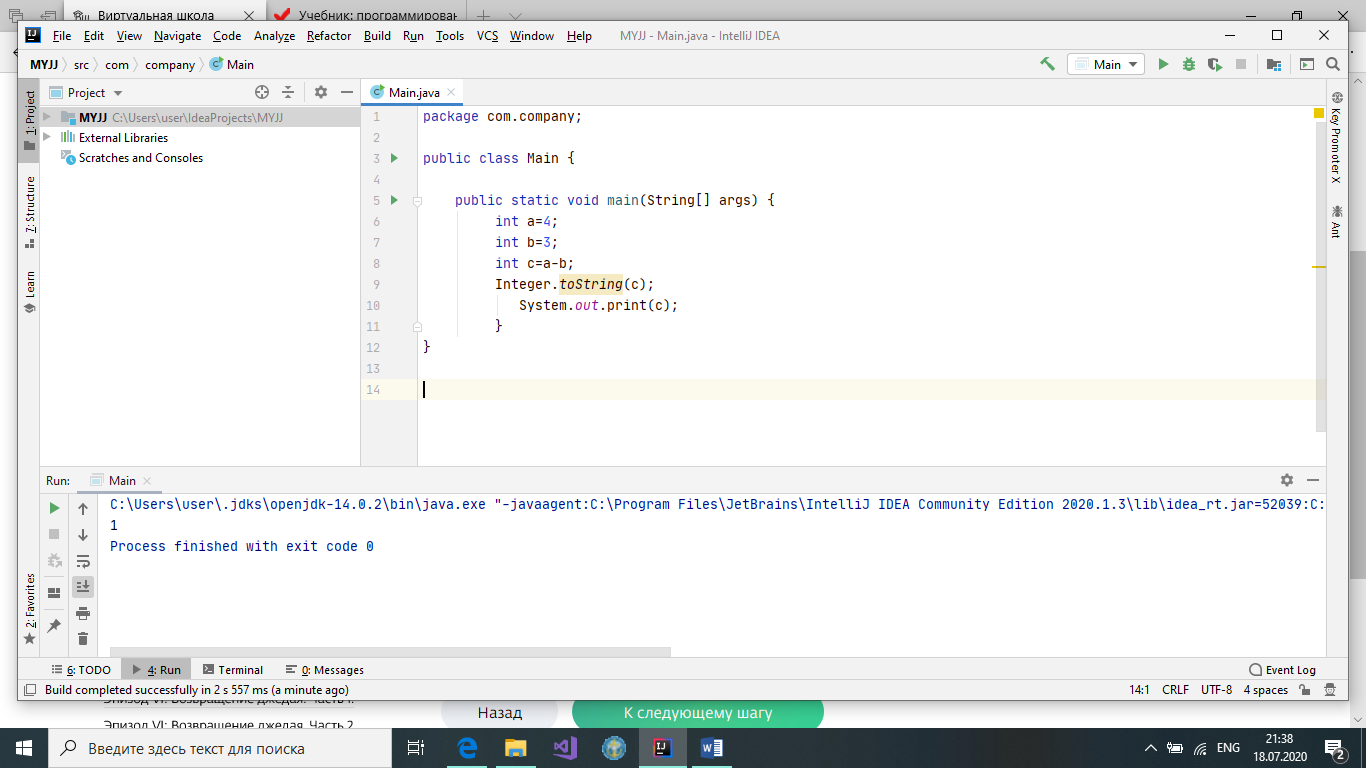
Слэши со звездочками подобно фигурным скобкам обозначают границы закомментированных строк.

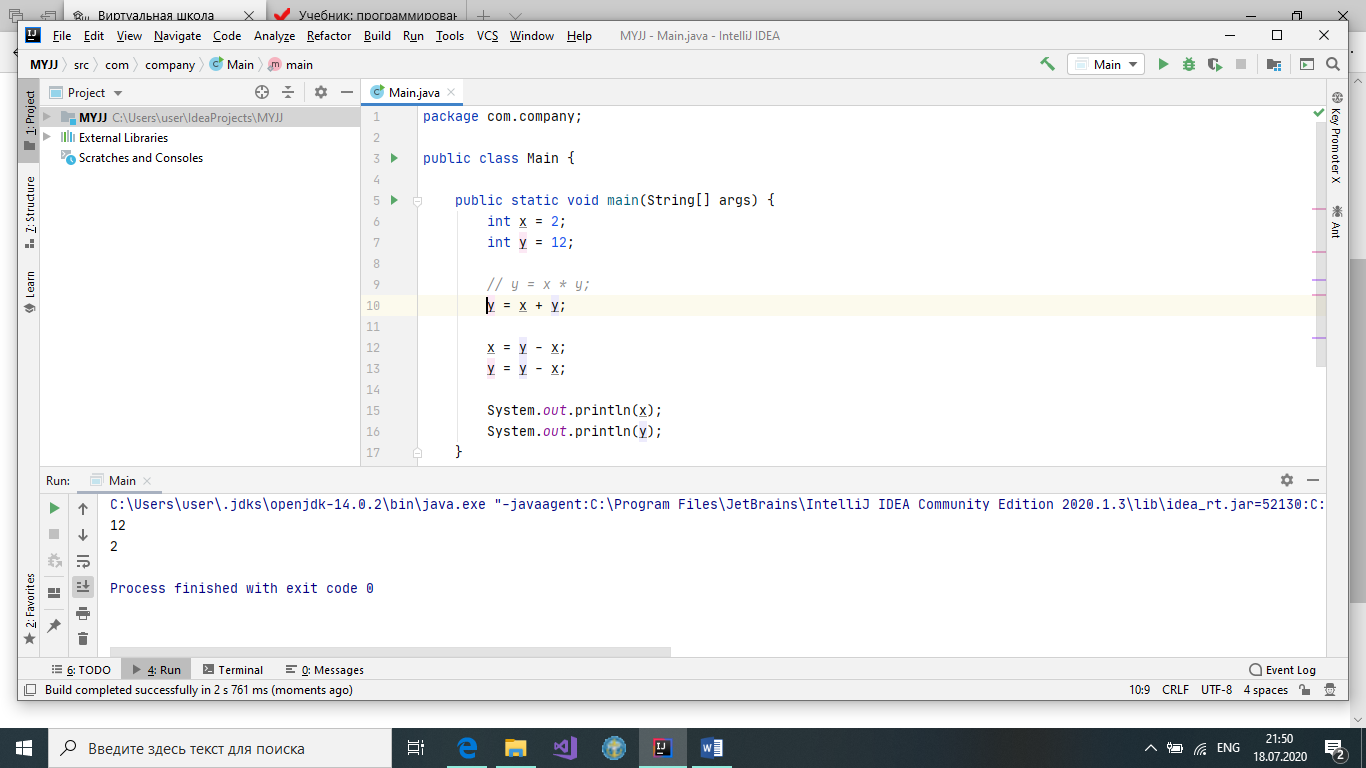
На этом мы завершим справочную информацию для данного Эпизода, некоторые задания будут содержать подсказки, указывающие на что обратить внимание при решении заданий. До встречи в следующем Эпизоде!

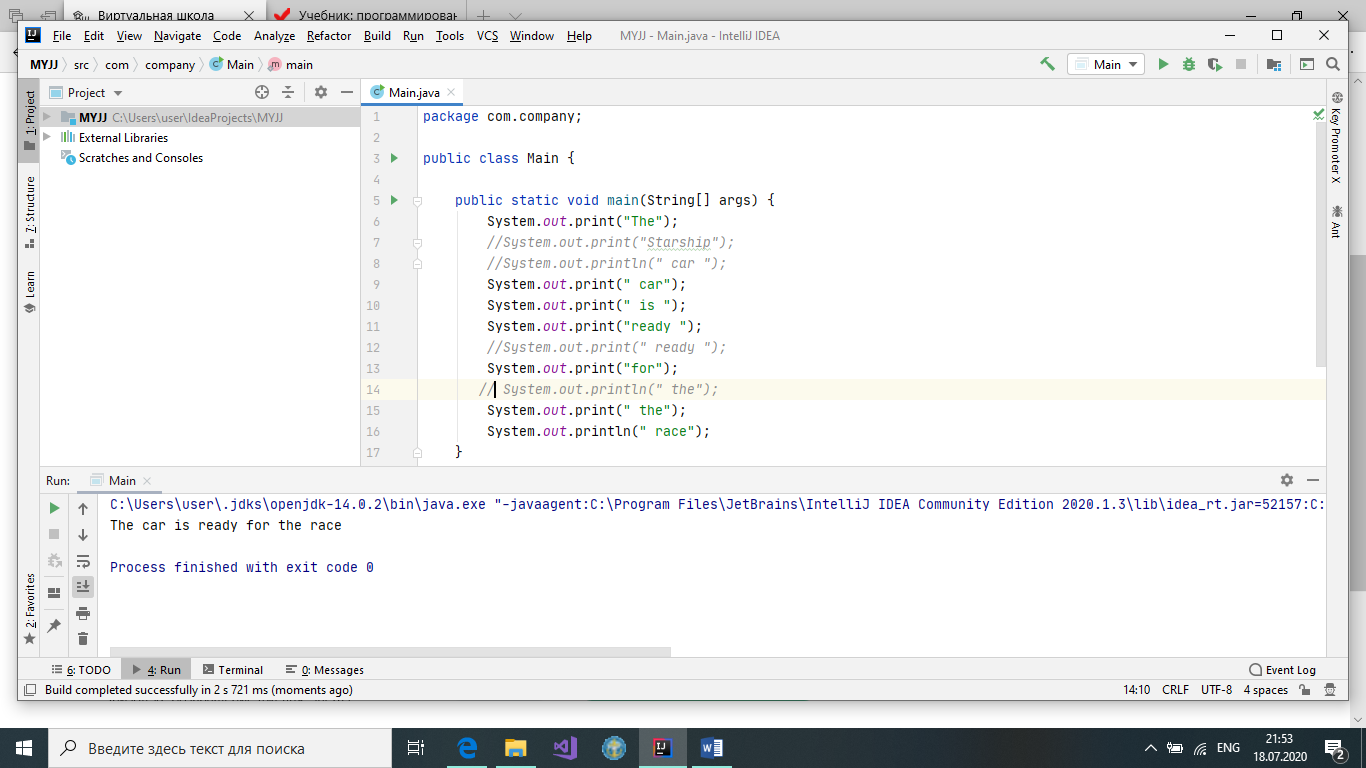












Переменные в Java - это своеобращный контейнер для хранения различных данных. Эти данные нужны нам для того, чтобы использовать на протяжении рабочего цикла программы.

Каждый тип характеризуется диапазоном допустимых значений и объемом занимаемой памяти. Мы выделим 4 группы простых типов данных, которыми мы можем манипулировать в программе:

1. Целочисленные:
   * byte [Диапазон значений: от -128 до 127; Объем занимаемой памяти: 1 байт]
   * short [Диапазон значений: от -32768 до 32767; Объем занимаемой памяти: 2 байта]
   * int [Диапазон значений: от -2147483648 до 2147483647; Объем занимаемой памяти: 4 байта]
   * long [Диапазон значений: от -9223372036854775808 до 9223372036854775807; Объем занимаемой памяти: 8 байт]

Как видно из информации выше, если ваша программа не использует целочисленные данные, которые превышали бы диапазон [-128:127], то вам нет необходимости использовать тип данных long, поскольку в противном случае вы будете расходовать болшьий ресурс памяти, чем требуется. Это некорректно с точки зрения ресурсопотребления программой.

1. Числа с плавающей точкой:
   * float [Диапазон значений: от -3.4E+38 до 3.4E+38; Объем занимаемой памяти: 4 байта]
   * double [Диапазон значений: от -1.7Е+308 до 1.7Е+308; Объем занимаемой памяти: 8 байт]

Запись типа E+38 - упрощенная запись выражения \*10^38. Это означает, что точка сместится на 38 знаков вправо от числа, записанного перед E. Также, чтобы объявить переменную типа float, нам необходимо добавить символ "f" в самом конце: -3.4E+38f.

1. Символьные:
   * char [Диапазон значений: от 0 до 65536; Объем занимаемой памяти: 2 байта]

char - символьный тип данных, предназначенный для одиночных символов, записываемых в одинарных кавычках. Например char symb = 'a';  
Диапазон значений - это цифровая запись символов по стандарту Unicode(UTF-16). Например char symb = 97 (это символ 'a' по стандарту UTF-16).

1. Логические:
   * boolean [Диапазон значений: true или false; Объем занимаемой памяти: достаточно 1 бита, но в реальности зависит от виртуальной машины, т.к. память не выделяется в таких малых объемах]

Для объявления переменной необходимо выбрать тип и присвоить ей имя:

byte n;

В таком виде переменная n не содержит никакого значения (null). С помощью символа "=" мы можем присвоить значение данной переменной:

byte n;  
n = 4;

Объявление переменной и присвоение ей значения можно проивзести и в одну строчку:

byte n = 4;

Над переменными можно производить различные операции, например увеличим переменную n на 6:

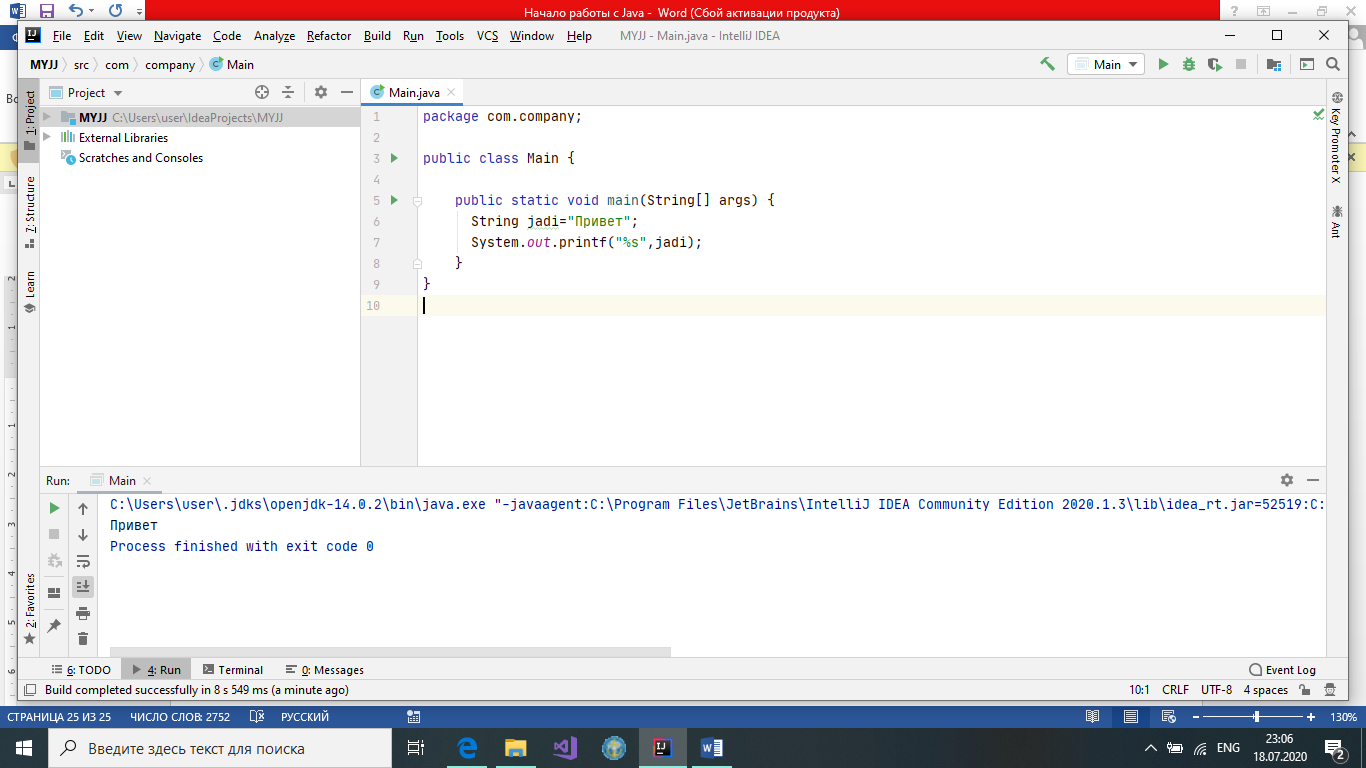
n = n + 6;  
System.out.println(n);

В результате переменная n будет равна 10, при условии, что изначально она была равна 4.

На этом мы завершим справочную информацию для данного Эпизода. До встречи в следующем Эпизоде!

Переменные можно объявлять в программах, присваивать им значения и выводить их значения на экран.  В этой задаче необходимо создать переменную типа String, присвоить ей значение и вывести на экран.

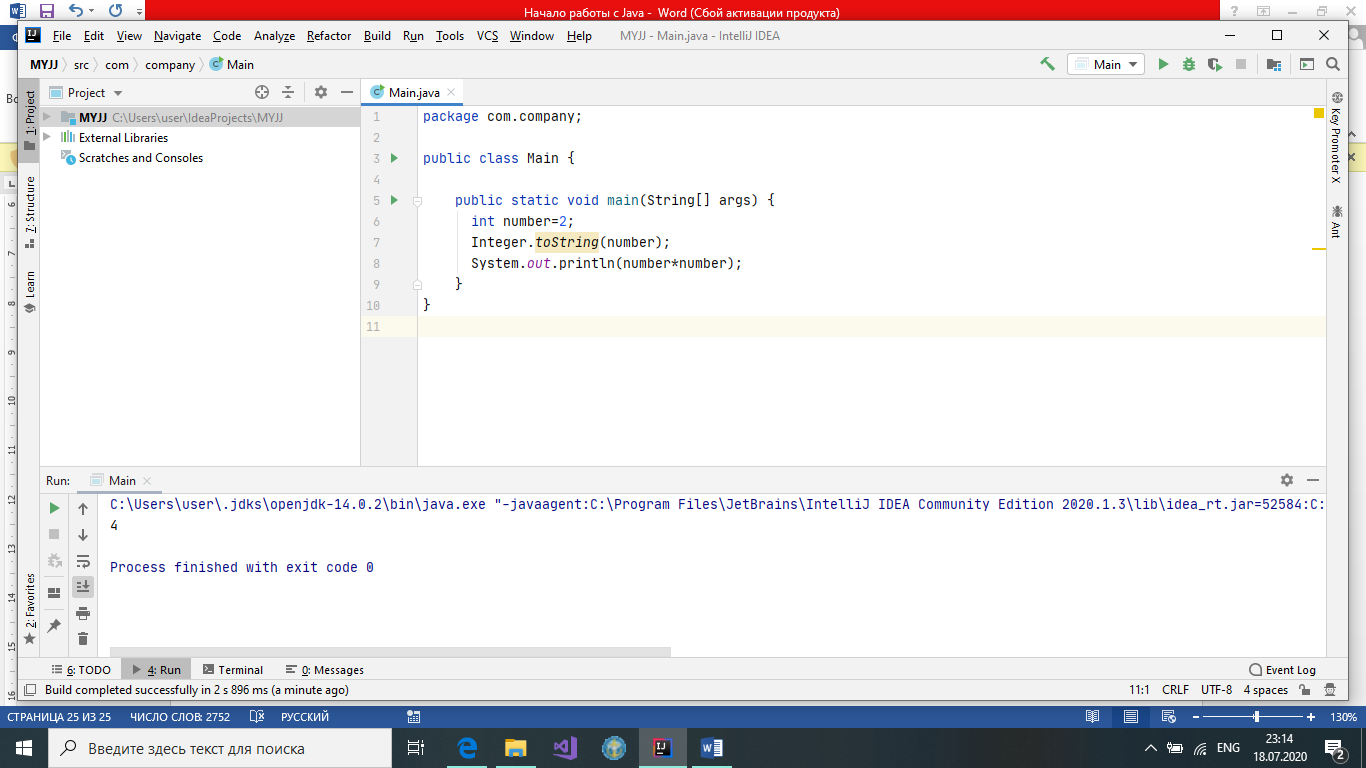
1. Объявите переменную jedi типа String.
2. Сразу же в строке объявления присвойте ей какое-нибудь значение.
3. Выведите на экран переменную jedi.



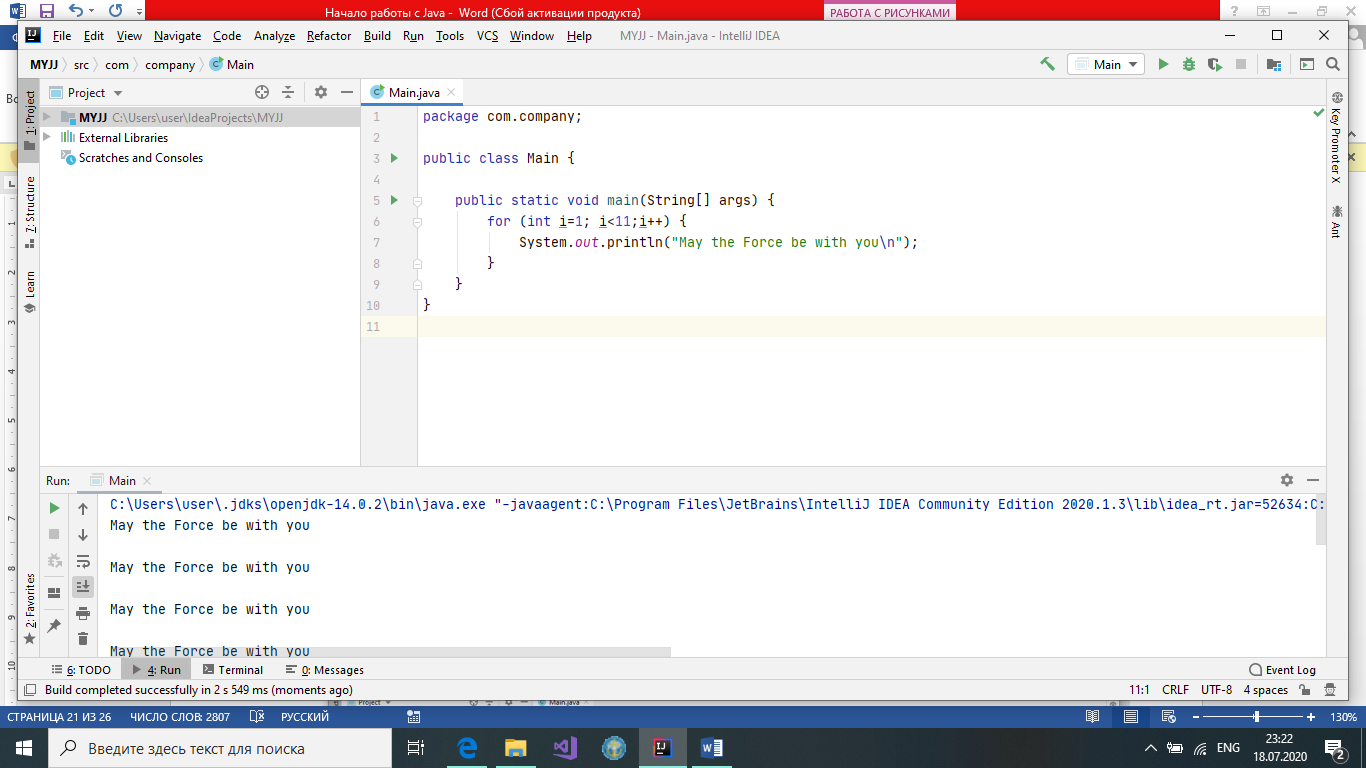
Дана переменная number.

Напишите программу, которая выводит на экран квадрат этой переменной (number \* number).

* Переменную number менять нельзя, менять можно только ее значение.
* Программа должна выводить текст.
* Выводимый текст должен быть числом.
* Выводимый текст должен быть квадратом переменной number.

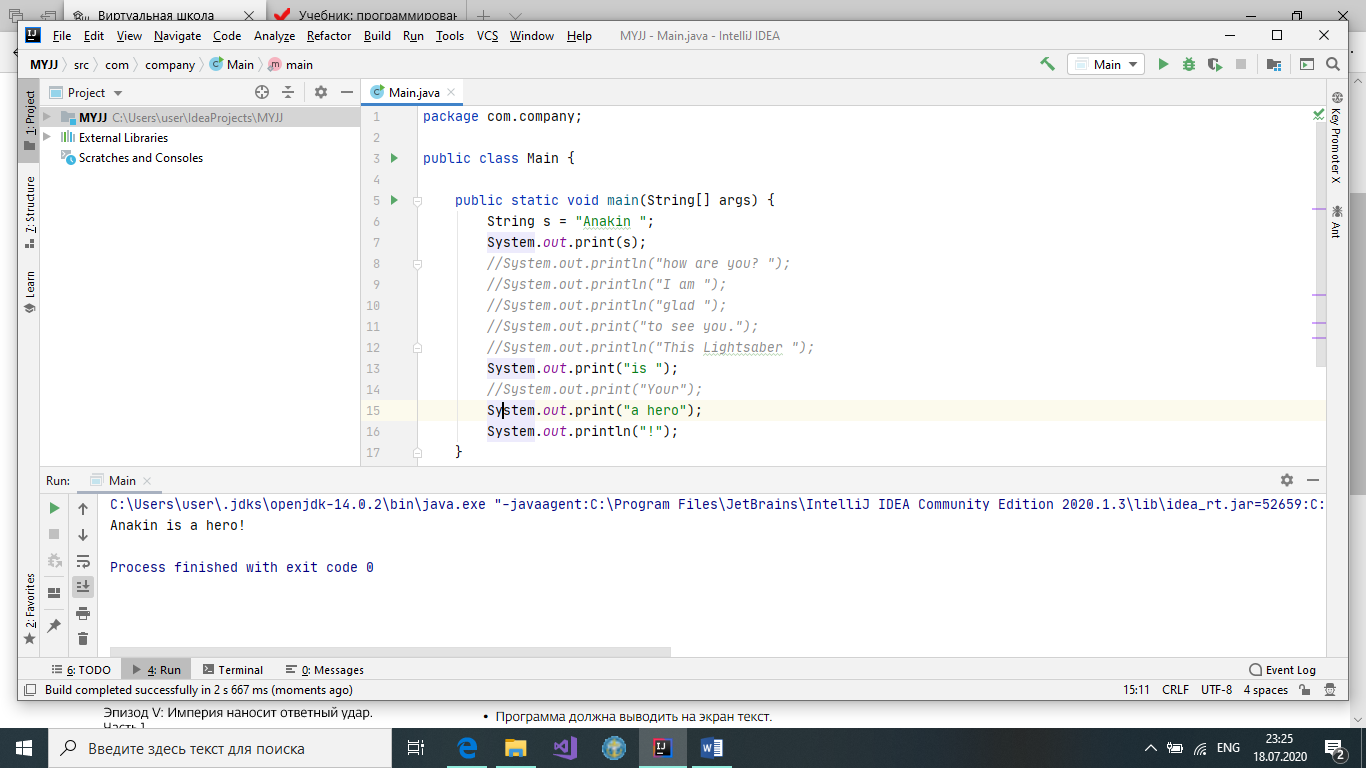


Напишите программу, которая выводит на экран надпись: "May the Force be with you." 10 раз.



Закомментируйте часть кода, чтобы на экран вывелось сообщение «Anakin is a hero!».

public class Solution {  
    public static void main(String[] args) {  
         String s = "Anakin ";  
         System.out.print(s);  
         System.out.println("how are you? ");  
         System.out.println("I am ");  
         System.out.println("glad ");  
         System.out.print("to see you.");  
         System.out.println("This Lightsaber ");  
         System.out.print("is ");  
         System.out.print("Your");  
         System.out.print("a hero");  
         System.out.println("!");  
    }  
}

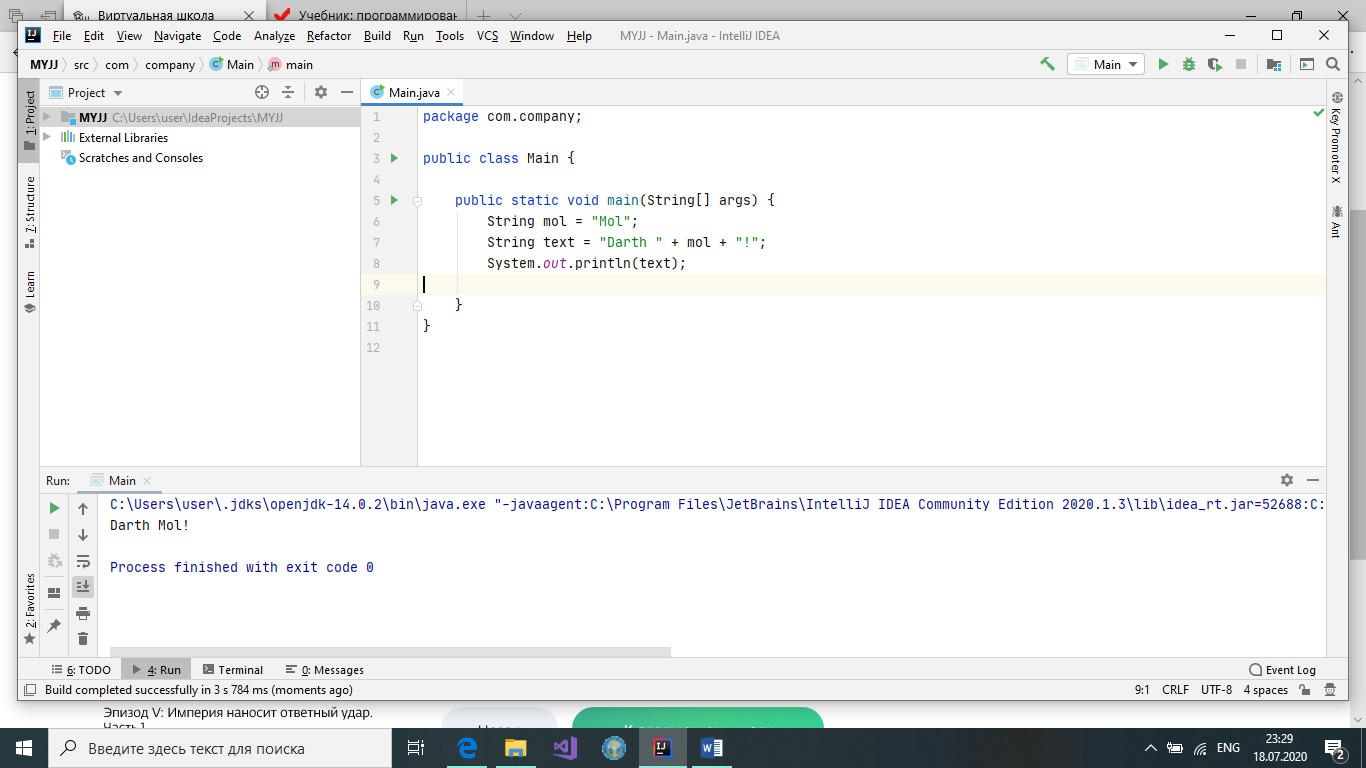


Внесите изменения в программу, чтобы переменная mol имела значение Mol.

public class Solution {  
    public static void main(String[] args) {  
        String darth = "Mol";  
        String text = "Darth " + mol + "!";  
        System.out.println(text);  
    }  
}

Требования:

* Переменной mol должно присваиваться значение сразу после создания.
* Должно быть объявлено 2 переменных.
* Значение переменной text изменять нельзя.
* Команду вывода текста на экран изменять нельзя.
* Программа должна выводить текст "Darth Mol!".



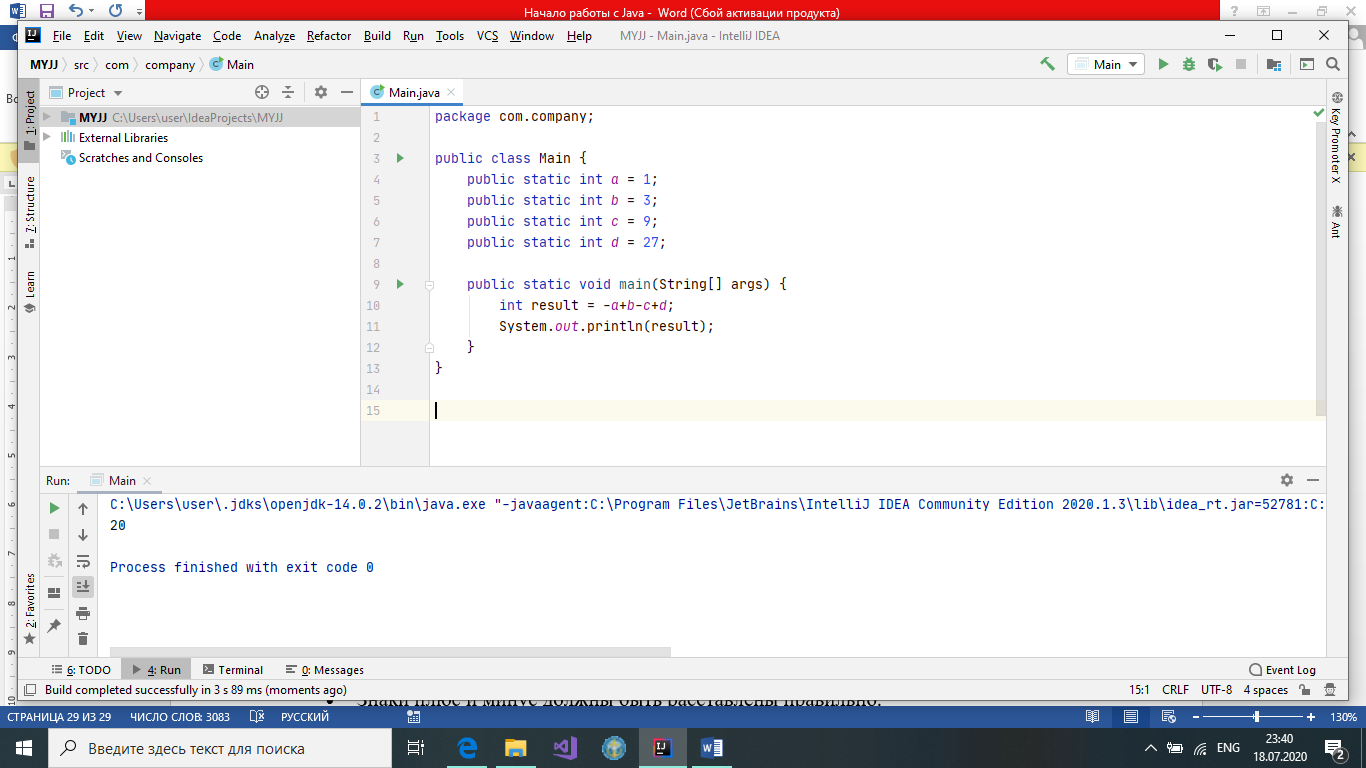
В данной задаче вам надо расставить знаки плюс и минус так, чтобы значение переменной result равнялось 20.

1. В методе main расставьте правильно знаки плюс и минус, чтобы значение переменной result получилось равным 20.
2. Знаки нужно расставить только в строчке, в которой объявляется переменная result.
3. Порядок следования переменных в этой строке изменять нельзя.
4. Перед каждой переменной должен стоять знак либо плюс, либо минус.

public class Solution {  
    public static int a = 1;  
    public static int b = 3;  
    public static int c = 9;  
    public static int d = 27;

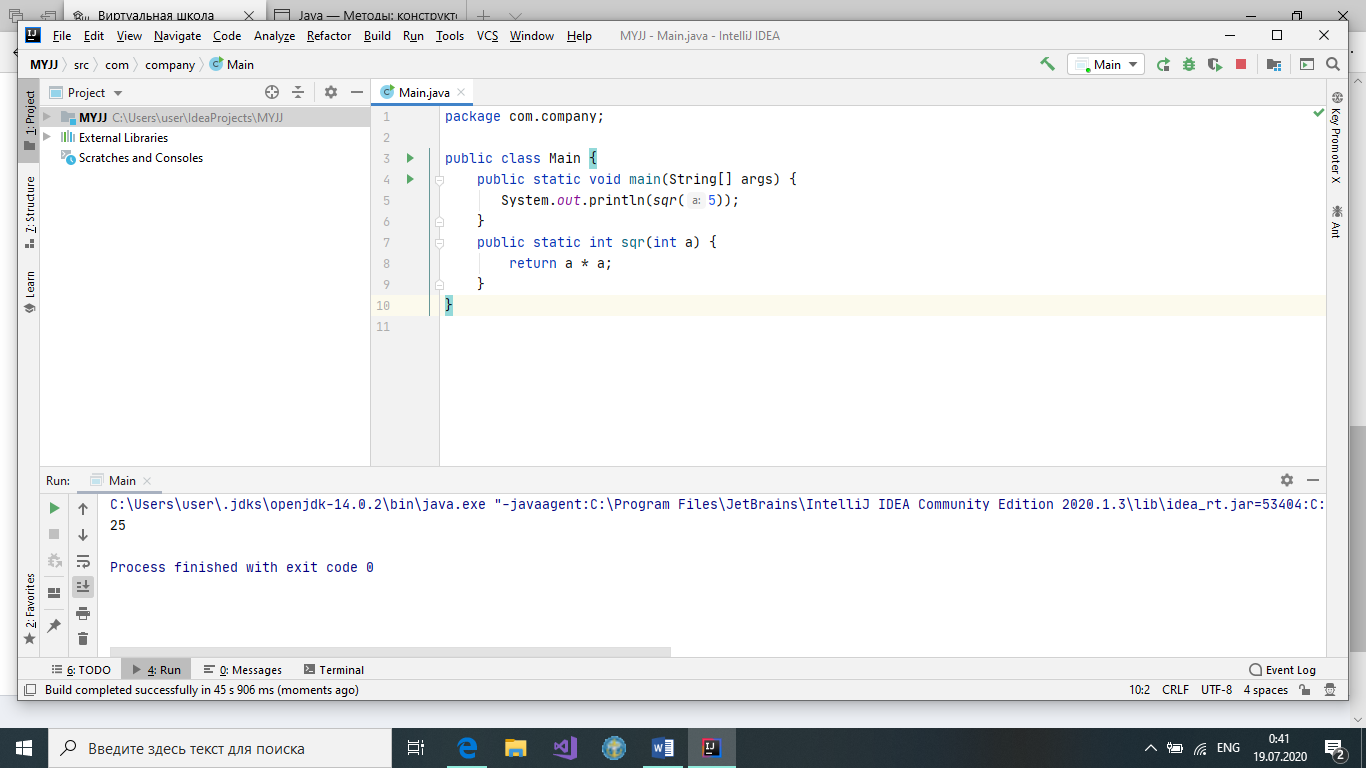
    public static void main(String[] args) {  
        int result = a b c d;  
        System.out.println(result);  
    }  
}

* Значения переменных: a, b, c, d изменять нельзя.
* Перед каждой из переменных: a, b, c, d в строке с объявлением переменной result должен стоять один знак плюс либо минус.
* В результате работы программы на экран нужно вывести число 20.
* Знаки плюс и минус должны быть расставлены правильно.



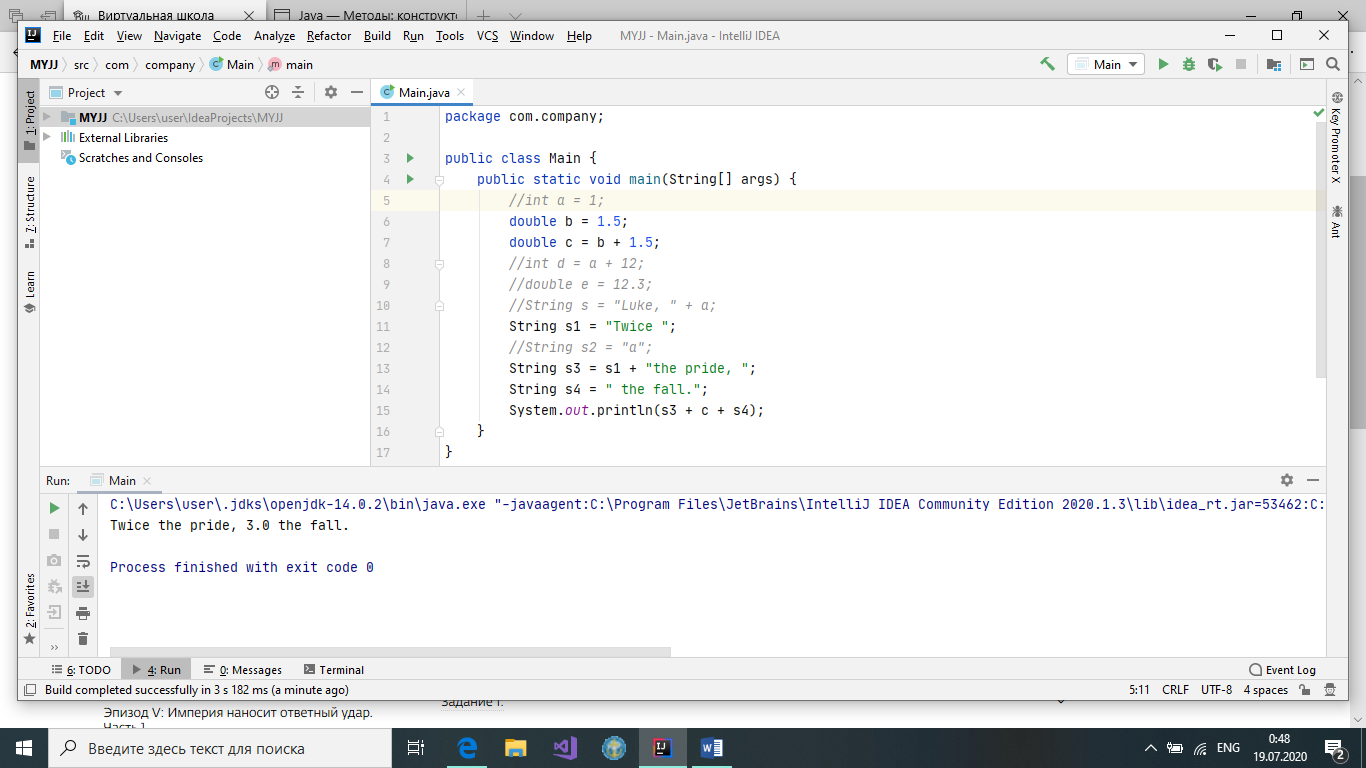
Напишите программу, которая выводит на экран квадрат числа 5.

* Программа должна выводить текст.
* Метод main должен вызывать метод sqr с параметром 5.
* Метод main должен выводить результат метода sqr.
* Метод sqr изменять нельзя.
* Выводимый текст должен соответствовать заданию.



закомментируйте переменные, которые нигде не используются. Программа должна компилироваться.

public class Solution {  
    public static void main(String[] args) {  
        int a = 1;  
        double b = 1.5;  
        double c = b + 1.5;  
        int d = a + 12;  
        double e = 12.3;  
        String s = "Luke, " + a;  
        String s1 = "Twice ";  
        String s2 = "a";  
        String s3 = s1 + "the pride, ";  
        String s4 = " the fall.";  
        System.out.println(s3 + c + s4);  
    }  
}



10 лет спустя

Вам предстоит написать метод print, который будет выводить на экран строку 4 раза. Строка — аргумент метода, то есть подаётся на входе.

/\*   
Реализуем метод print  
\*/  
public class Solution {  
    public static void main(String[] args) {  
        print("The power is easy to use!");  
        print("The power opens many opportunities!");  
    }  
    public static void print(String s) {  
        //напишите тут ваш код  
    }  
}

Требования:

* Программа должна выводить текст на экран.
* Метод main не должен вызывать System.out.println или System.out.print.
* Метод main должен вызвать метод print класса Solution ровно два раза.
* Метод print должен выводить текст на экран. Метод print должен выводить на экран строку 4 раза.

Справочная информация

В программировании стараются уходить от повторений одной операции несколько раз. Представьте, что вы нашли ошибку, которая повторяется в 10 местах и вам надо ее исправить - не очень удобно, правда? Другое дело, когда та же ошибка встречается в одном месте, но это место вызывается в 10 местах программы. В этом случае достаточно внести всего 1 исправление. Избежать таких проблем можно разными способами. Например можно создать отдельный метод, который будет отвечать за определенную операцию и потом вызывать его каждый раз, когда требуется:

public class Solution {  
    public static void main(String[] args) {  
        count(5);  
    }  
    public static void count(int n) {  
        System.out.println("Первая степень числа "+n+" = "+n);  
 System.out.println("Квадрат числа "+n+" = "+n\*n);  
 System.out.println("Куб числа "+n+" = "+n\*n\*n);  
 System.out.println("Четвертая степень числа "+n+" = "+n\*n\*n\*n);  
    }  
}

Представьте, если на примере выше вместо переменной n мы бы писали конкретное число, а потом нам надо было изменить его. Для удобства мы создали метод count() с целочисленным аргументом int n, который используется в различных командах внутри метода. Для того, чтобы метод выполнил действия, его надо вызвать. Вызов метода мы осуществляем внутри главного метода main, подставляя значение аргумента в виде числа - count(5).

Методы могут не иметь аргументов, тогда их вызов будет осуществляться с пустыми внутри круглыми скобками - count(). Необходимость аргументов всегда зависит от задачи.

Давайте напишем программу, которая выводит сообщение о том, что есть некий Майк и у него есть черная кошка:

public class Solution {  
    public static void main(String[] args) {  
        System.out.println("Hello, my name is Mike and my cat is black");  
    }  
}

А что, если нам необходимо менять цвет животного и имя владельца ? Попробуем использовать переменные :

public class Solution {  
    public static void main(String[] args) {  
        String name = "Mike";  
        String cat = "black";  
        System.out.println("Hello, my name is "+name+" and my cat is "+cat);  
    }  
}

Теперь мы можем изменять значения переменных, но что, если нам понадобится использовать вывод фразы несколько раз? Попробуем реализовать в виде метода, как разбиралось в самом начале данной справки:

public class Solution {  
    public static void main(String[] args) {  
        say("Mike", "black");  
    }  
    static void say(String name, String color){  
        System.out.println("Hello, my name is "+name+" and my cat is "+color);  
    }  
}

А почему мы вообще рассматриваем только кота, а что, если у Майка вообще другое животное, или другого цвета, или владельца животного вообще зовут не Майк,или их вообще несколько. А может нам необходимо использовать данную информацию еще где-то? Для реализации сложных программ нам помогут классы и объекты, которых мы касались в первом Эпизоде. Давайте обобщим наши данные. Во-первых у нас есть владелец животного (Owner) и само животное (Pet). В случае обобщения каждый Owner имеет имя, а каждое животное обладает видом (кошка, собака) и цветом. Давайте попробуем реализовать данную схему с помощью классов. Owner и Pet в нашей модели будут классами со своими свойствами:

public class Solution {  
    public static void main(String[] args) {  
    }  
    static class Owner{  
        String name;  
  
        void say(){  
            System.out.println("Hello, my name is " + name + " and my " + kind + " is " + color);  
        }  
    }  
    static class Pet{  
         String kind;  
         String color;  
    }  
}

Как и говорили ранее, у Owner-a есть имя, у животного вид и цвет, помимо этого, владелец обладает методом - говорит фразу из предыдущих примеров. Сейчас созданные классы играют роль шаблонов и в таком виде работать программа не будет. Нам необходимы объекты - конкретные представители каждого класса, добавим одного владельца и одну черную кошку:

public class Solution {  
    public static void main(String[] args) {  
        Owner owner\_1 = new Owner();  
        Pet pet\_1 = new Pet();  
    }  
    static class Owner{  
        String name;  
  
  
        void say(){  
            System.out.println("Hello, my name is " + name + " and my " + kind + " is " + color);  
        }  
    }  
    static class Pet{  
         String kind;  
         String color;  
    }  
}

Owner owner\_1 - данной командой мы объявили переменную owner\_1 класса Owner. Без правой части переменная owner\_1 имеет значение null. С помощью конструкции "= new Owner()" мы создаем объект класса Owner и связываем его с переменной owner\_1. Подобно операции String m = "12345", мы должны объявить переменную, создать объект одного класса и связать их. Аналогичное происходит с объектом pet\_1. Теперь наши объекты обладают свойствами, прописанными в классах, однако они пока не имеют значений. Давайте добавим их:

public class Solution {  
    public static void main(String[] args) {  
        Owner owner\_1 = new Owner();  
        Pet pet\_1 = new Pet();  
        owner\_1.name = "Mike";  
        pet\_1.kind = "cat";  
        pet\_1.color = "black";  
    }  
    static class Owner{  
        String name;  
  
        void say(){  
            System.out.println("Hello, my name is " + name + " and my " + kind + " is " + color);  
        }  
    }  
    static class Pet{  
         String kind;  
         String color;  
    }  
}

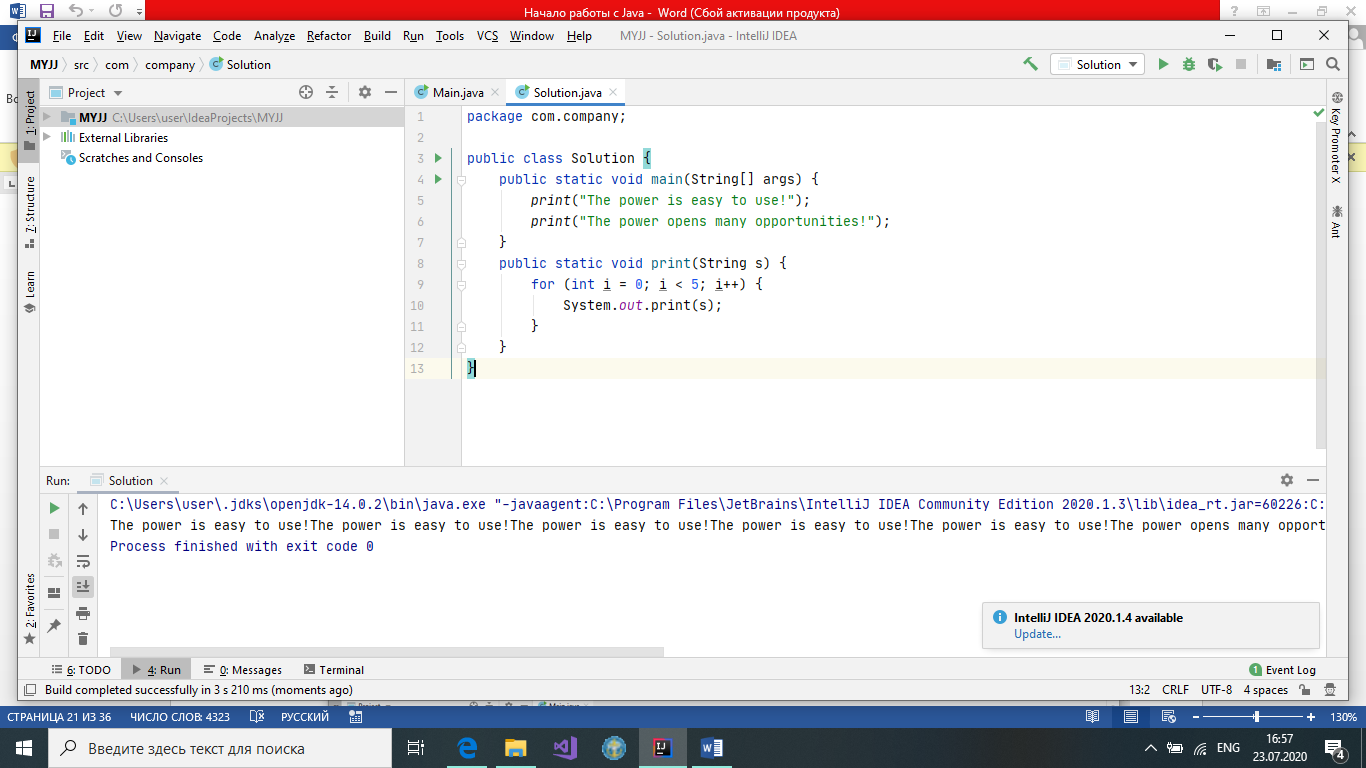
Мы связали свойство name для owner\_1 со значением "Mike". Обратите внимание, что типы свойства и значения совпадают. Обратиться к свойству объекта можно записав их через точку - owner\_1.name . Теперь, когда наши объекты обладают определенными свойствами, нам надо связать их между собой. Связь необходима, чтобы мы могли передать свойства класса Pet в класс Owner для вывода сообщения.

public class Solution {  
    public static void main(String[] args) {  
        Owner owner\_1 = new Owner();  
        Pet pet\_1 = new Pet();  
        owner\_1.name = "Mike";  
        pet\_1.kind = "cat";  
        pet\_1.color = "black";  
        owner\_1.pet = pet\_1;  
        owner\_1.say();  
    }  
    static class Owner{  
        String name;  
        Pet pet;  
        void say(){  
            System.out.println("Hello, my name is " + name + " and my " + pet.kind + " is " + pet.color);  
        }  
    }  
    static class Pet{  
         String kind;  
         String color;  
    }  
}

В классе Owner мы объявляем переменную pet класса Pet (Pet pet) и связываем ее с объектом pet\_1 (owner\_1.pet = pet\_1). Таким образом, у переменной pet есть ссылки на свойства kind и color из класса Pet. Обратившись к ним (pet.kind и pet.color) мы можем наконец вызвать метод (owner\_1.say()) и получить вывод сообщения на экран.

Такое разбиение по классам позволит нам создать новые объеты с другими именами и видами животных и связать их между собой. Каждый новый объект будет создаваться со стандартным набором свойств, что очень удобно. Также это позволит нам хранить данные связи на протяжении времени работы программы. Подобные приемы сильно помогают упросить код при разработке больших и солжных программ.

На этом мы завершим справочную информацию для данного Эпизода. До встречи в следующем Эпизоде!

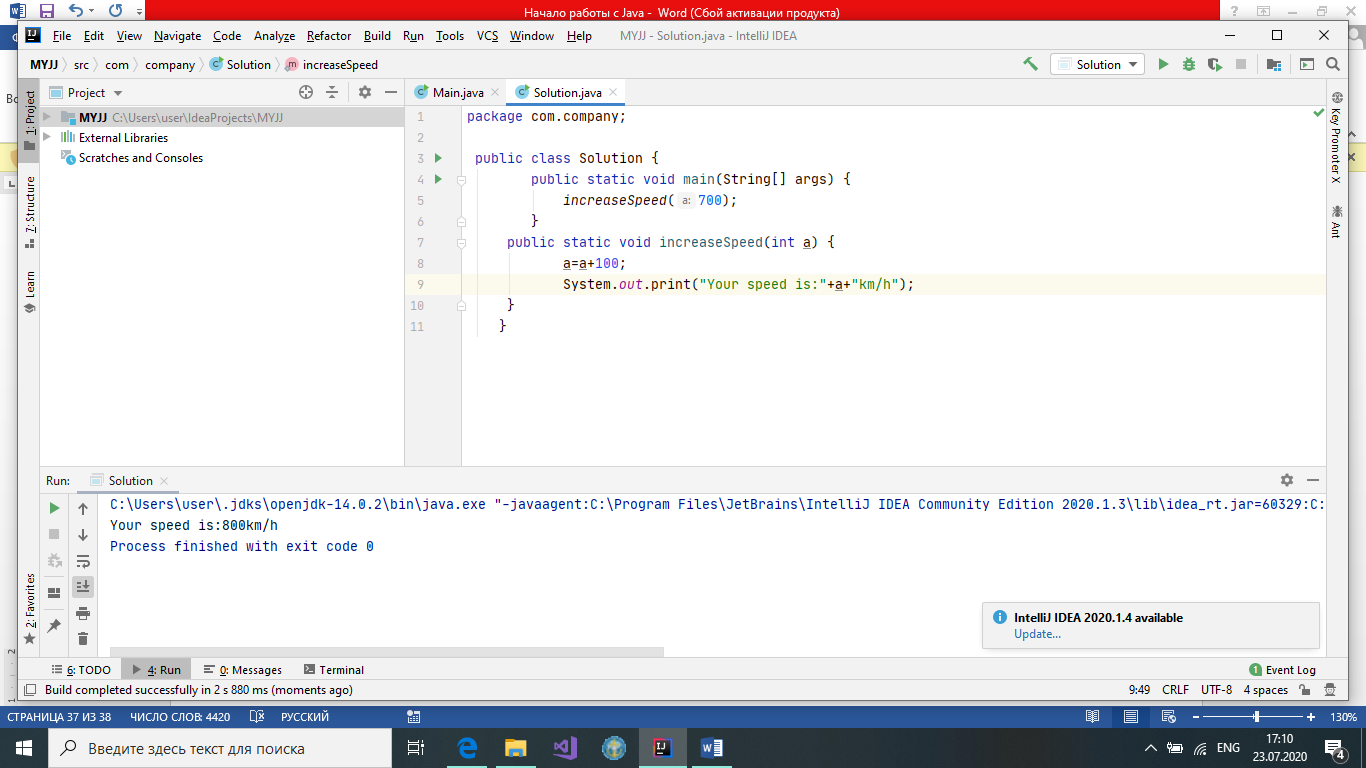


Напишите метод public static void increaseSpeed(int a), который будет принимать значение скорости n,  увеличивать ее  на 100 и выведите на экран надпись:

"Your speed is: <n+100> km/h."

Где <n+100> - это скорость увеличенная на 100.

* public class Solution {  
      public static void main(String[] args) {  
          increaseSpeed(700);  
      }  
      //напишите тут ваш код  
  }
* Программа должна выводить текст на экран.
* Метод main не должен вызывать System.out.println или System.out.print напрямую.
* Метод increaseSpeed не должен возвращать результат.
* Метод main должен вызвать метод increaseSpeed только один раз.
* Метод increaseSpeed должен увеличить переданное число на 100 и вывести на экран надпись согласно шаблону.



public class Solution {  
 public static void main(String[] args) {  
 increaseSpeed(700);  
 }  
  
 public static void hackSalary(int a) {  
 System.out.println("Your speed is: " + (a + 100) + " km/h.");  
 }  
}

В методе main создайте объект Zam, сохраните ссылку на него в переменную zam.

Создайте также объект Dron и сохраните ссылку на него в переменную dron.

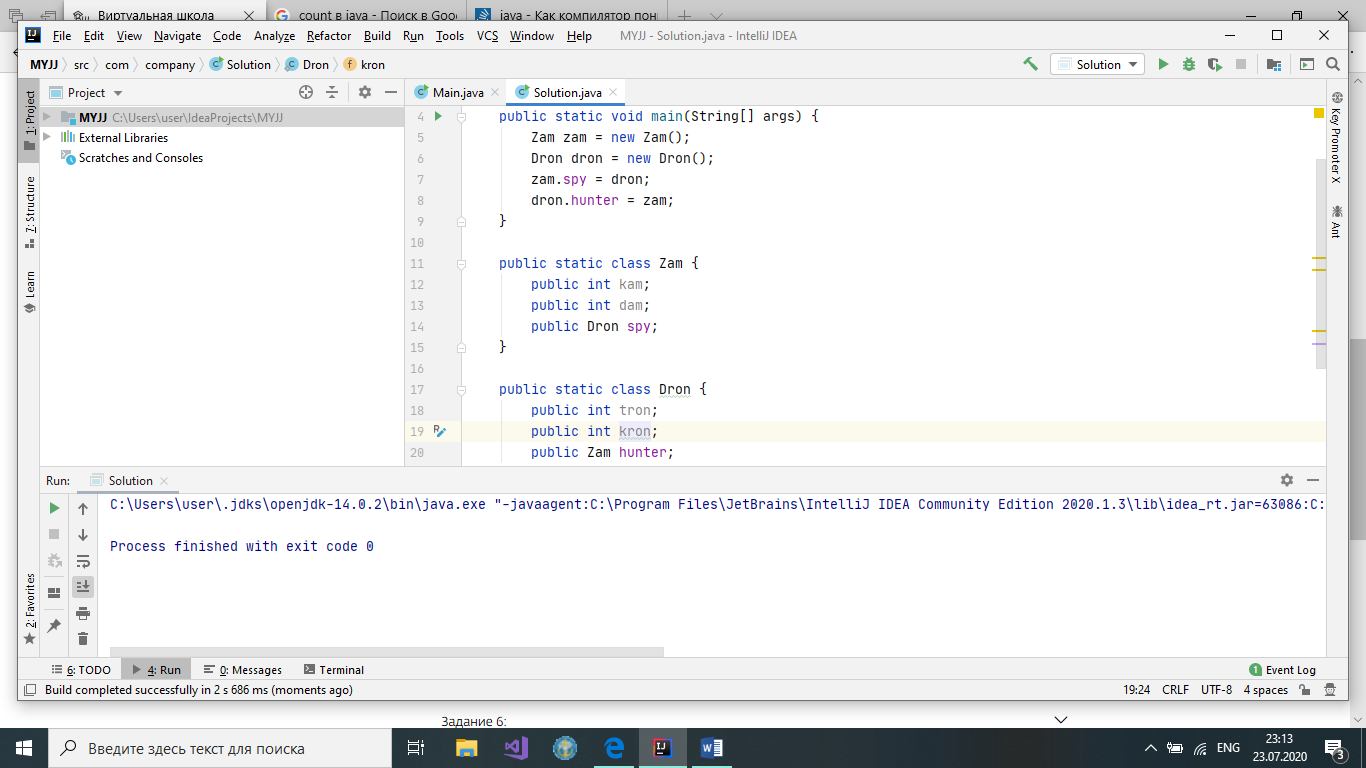
public class Solution {  
    public static void main(String[] args) {  
        //напишите тут ваш код  
    }  
    public static class Zam {  
        //напишите тут ваш код  
    }  
    public static class Dron {  
        //напишите тут ваш код  
    }  
}

Требования:

* В методе main создайте объект Zam и сразу сохраните ссылку на него в переменную zam.
* В методе main создайте объект Dron и сразу сохраните ссылку на него в переменную dron.
* В методе main сохраните ссылку на ранее созданный объект Dron в zam.spy.
* В методе main сохраните ссылку на ранее созданный объект Zam в dron.hunter.
* Класс Zam должен содержать 3 переменные.
* Класс Dron должен содержать 3 переменные

Подсказка:

В данной задаче программа не должна ничего выводить, достаточно того, что она компилируется. Вам необходимо прописать связи в классах Dron и Zam, чтобы можно было сохранить ссылку на объект (zam.spy = dron). 2 из 3 переменных для класса Zam и Dron могут быть любыми.



Создать 3 объекта типа Jedi.

Сохрани каждый экземпляр в отдельную переменную.

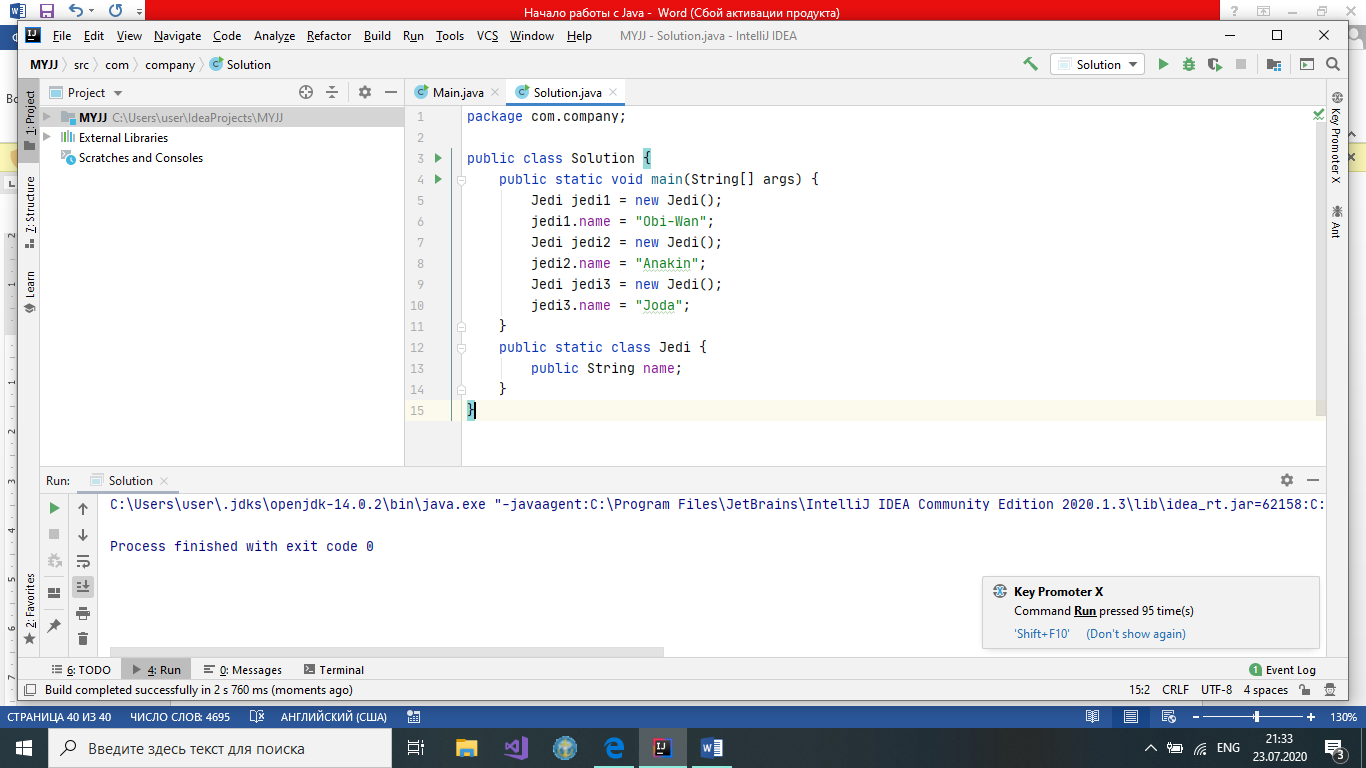
Присвоить им имена "Obi-Wan", "Anakin", "Joda".

public class Solution {  
    public static void main(String[] args) {  
        Jedi jedi1 = new Jedi();  
        jedi1.name = "Obi-Wan";  
        //напишите тут ваш код  
    }  
    public static class Jedi {  
        public String name;  
    }  
}

Требования:

бования:

* Программа не должна выводить текст на экран.
* В методе main должно быть только три переменные типа Jedi.
* Переменным сразу должны быть присвоены значения.
* Каждому объекту типа Jedi должно быть присвоено имя. В классе Jedi должна быть одна переменная name.
* В классе Jedi не должно быть методов.

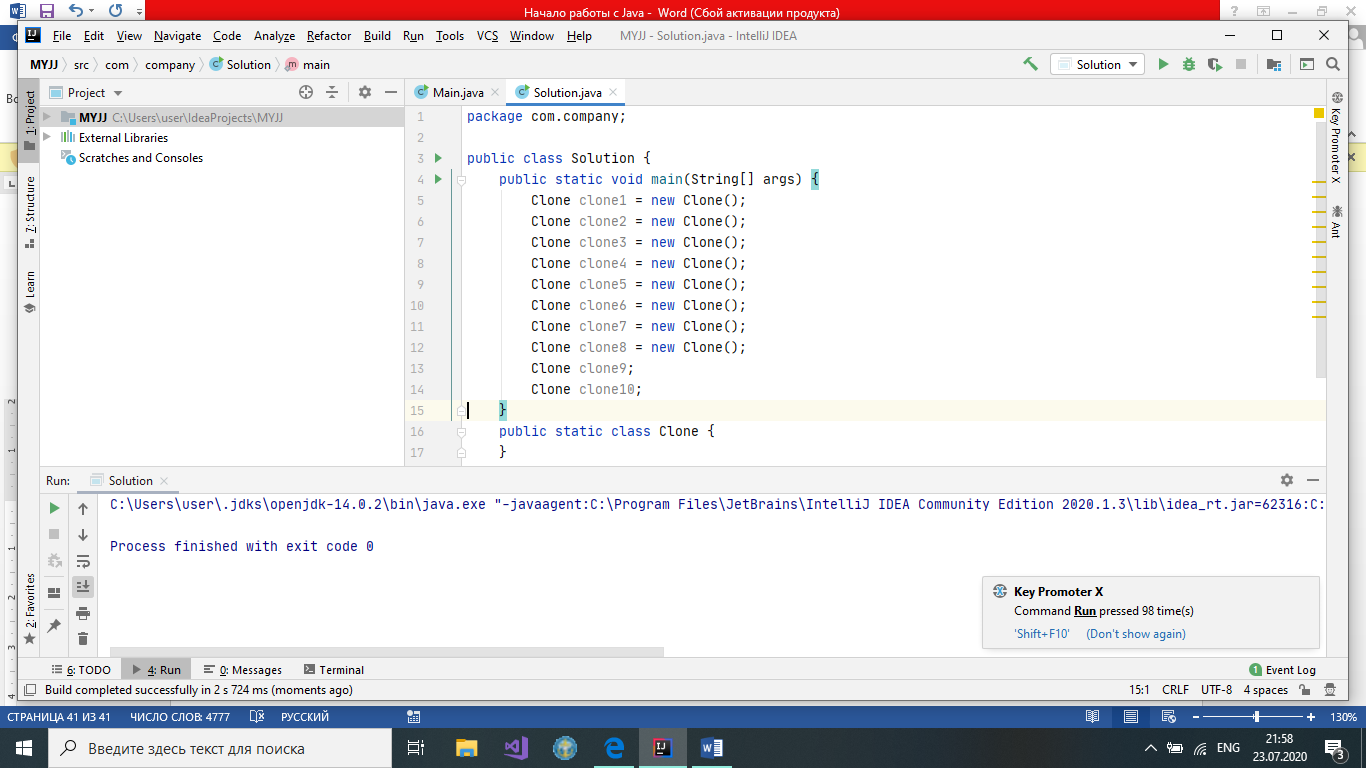


Создай 10 переменных типа Clone и 8 объектов типа Clone.

public class Solution {  
    public static void main(String[] args) {  
        //напишите тут ваш код  
        Clone clone8 = new Clone();  
        Clone clone9;  
        //напишите тут ваш код  
    }  
    public static class Clone {  
    }  
}

Требования:

* Программа не должна выводить текст на экран.
* В методе main должно быть 10 переменных типа Clone.
* В методе main 8 переменных должны быть проинициализированы.
* В классе Clone не должно быть переменных.
* В классе Clone не должно быть методов.



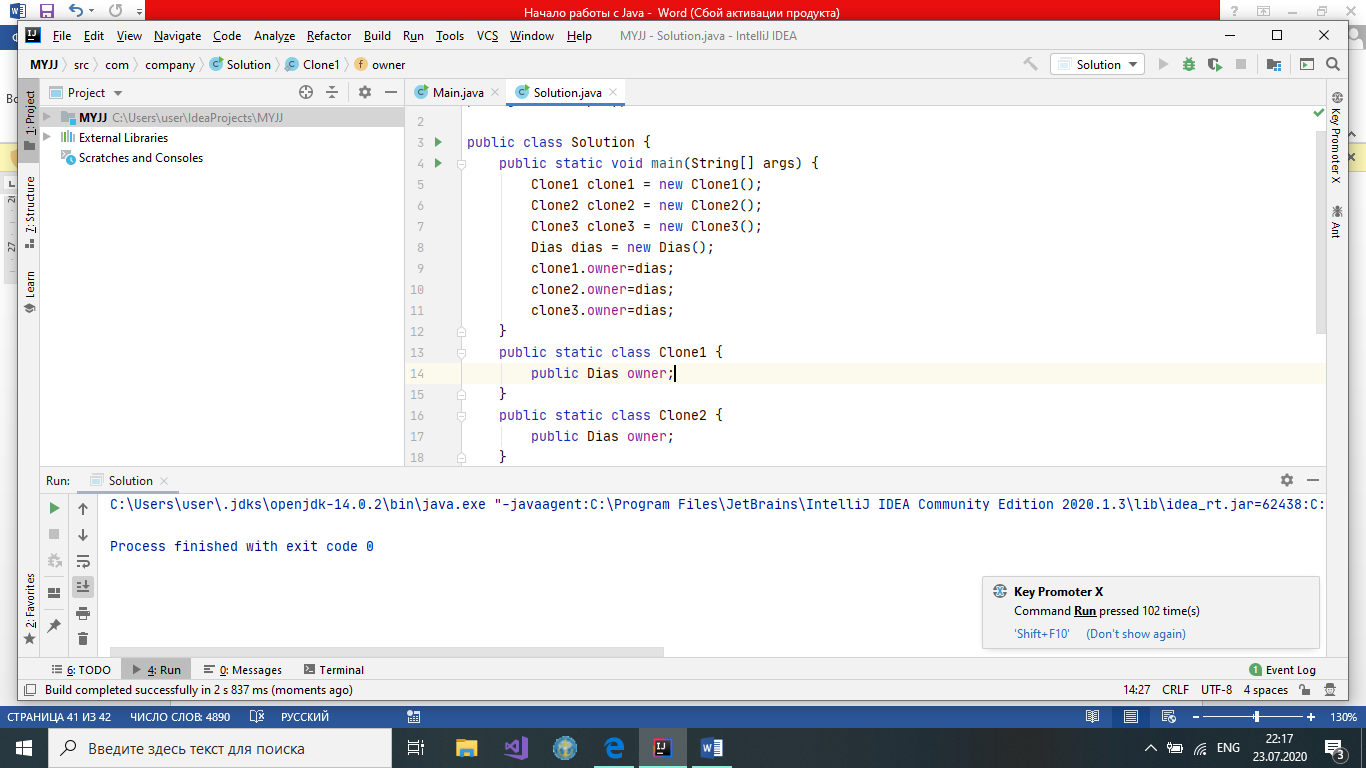
Создайте объект типа Clone1, Clone2, Clone3 и объект типа Dias.

Присвой каждому клону владельца (owner).

public class Solution {  
    public static void main(String[] args) {  
        //напишите тут ваш код  
    }  
    public static class Clone1 {  
        public Dias owner;  
    }  
    public static class Clone2 {  
        public Dias owner;  
    }  
    public static class Clone3 {  
        public Dias owner;  
    }  
    public static class Dias {  
    }  
}

Требования:

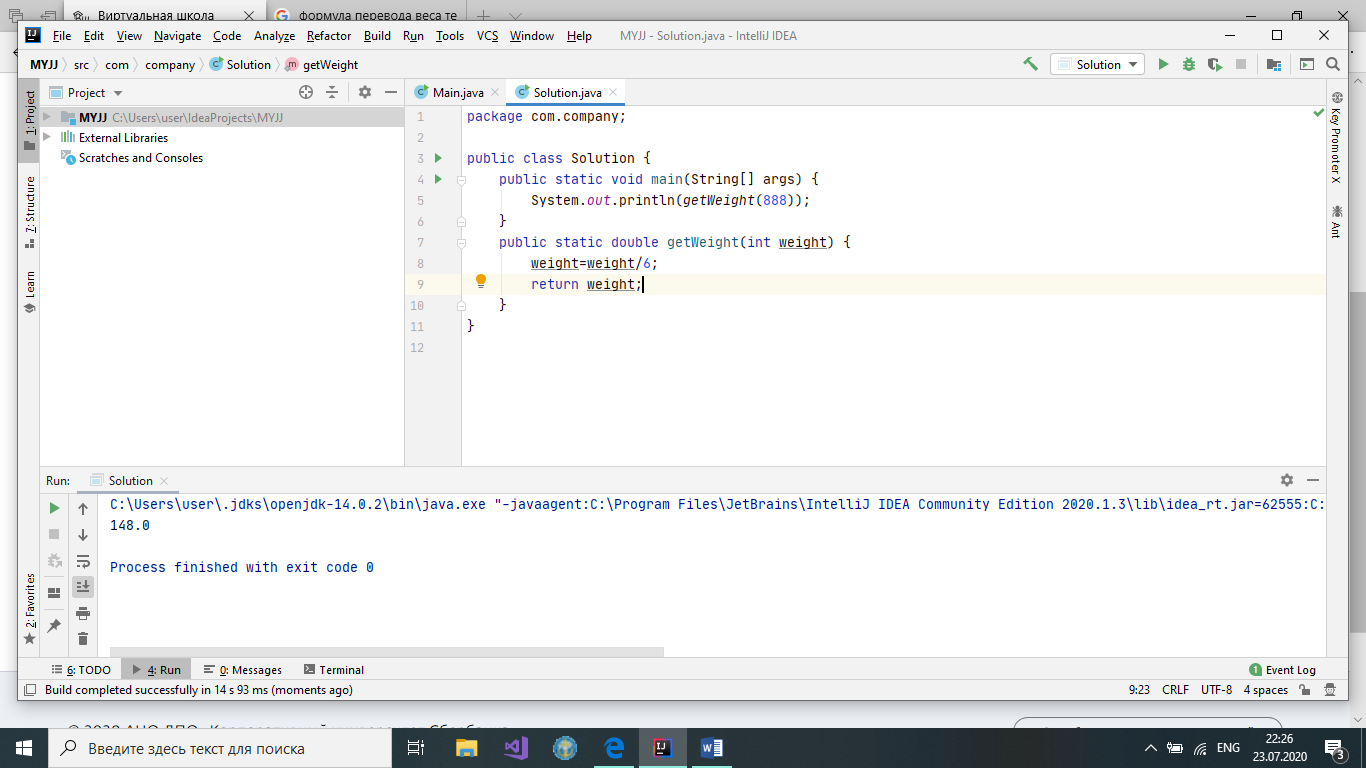
* Программа не должна выводить текст на экран.
* В методе main создайте объекты типа Clone1, Clone2, Clone3, Dias занесите их ссылки в переменные.
* В методе main присвойте каждому клону владельца Dias.
* Каждый класс Clone1, Clone2, Clone3 должен содержать только одну переменную Dias owner.
* Класс Dias не должен содержать переменных.



Зная, что на планете Камино гравитация, как на Луне, реализуйте метод, который переводит земной вес в лунный. Реализуй метод getWeight(int), который принимает вес тела (в Ньютонах) на Земле, и возвращает, сколько это тело будет весить на Луне (в Ньютонах).  Тип возвращаемого значения - double.

public class Solution {  
    public static void main(String[] args) {  
        System.out.println(getWeight(888));  
    }  
    public static double getWeight(int weight) {  
        //напишите тут ваш код  
    }  
}

* Метод getWeight(int) должен быть публичным и статическим.
* Метод getWeight должен возвращать значение типа double.
* Метод getWeight не должен ничего выводить на экран.
* Метод getWeight должен правильно переводить вес тела в Ньютонах на Земле в вес этого же тела на Луне, и возвращать это значение.



Давайте перешлем сообщение астродроиду в понятном ему формате. Для этого реализуем метод print3. Реализуйте метод print3. Метод должен вывести переданную строку (слово) на экран три раза через пробел.

public class Solution {  
    public static void print3(String s) {  
        //напишите тут ваш код  
    }  
    public static void main(String[] args) {  
        print3("dump");  
        print3("cargo");  
    }  
}

* Программа должна выводить текст на экран.
* Метод main не должен вызывать функцию System.out.println или System.out.print.
* Метод print3 должен выводить текст на экран.
* Метод main должен вызвать метод print3 ровно два раза.
* Метод print3 должен выводить переданную строку (слово) на экран три раза, но в одной строке.
* public class Solution {  
   public static void print3(String s) {  
   System.out.print(s + " " + s + " " + s + " ");  
   }  
    
   public static void main(String[] args) {  
   print3("dump");  
   print3("cargo");  
   }  
  }

Преследование продолжается

Увидев, что ракета попала по цели, Фетт улетает на планету Дженозис, вы же, успешно спрятавшись за астероидом затаились, чтобы выждать, когда наемник отлетит на минимальное безопасное расстояние, чтобы проследовать за ним.

Напишите функцию, которая возвращает минимум из двух чисел.

public class Solution {  
    public static int min(int a, int b) {  
        //напишите тут ваш код  
    }  
    public static void main(String[] args) {  
        System.out.println(min(12, 33));  
        System.out.println(min(-20, 0));  
        System.out.println(min(-10, -20));  
    }  
}

Требования:

* Программа должна выводить текст на экран.
* Метод min не должен выводить текст на экран.
* Метод main должен вызвать метод min три раза.
* Метод main должен выводить на экран результат работы метода min. Каждый раз с новой строки.
* Метод min должен возвращать минимальное значение из чисел a и b.

Ранее не требовался возврат значений, в данной задаче необходимо, чтобы метод возвращал значение. Помимо этого здесь требуется применить условную конструкцию if-else.