1.

Измените программу таким образом, чтобы она выводила на печать числа 11, 12, 13. Используйте инкремент.

Sample Input:

Sample Output:

11

12

13

class MyNumber {

public static void main(String[] args) {

int x = 10;

// put your code here

}

}

==================Вопрос================

Я написал так:

class MyNumber {

public static void main(String[] args) {

int x = 10;

int x1 = ++x;

int x2 = ++x1;

int x3 = ++x2;

System.out.println(x1);

System.out.println(x2);

System.out.println(x3);

}

}

Но у меня выводится 12, 13, 13 и я не понимаю почему так.

int x = 10;

int x1 = ++x; //Здесь переменная х1 должна принимать значение 11, тк начальный Х=10 и мы к нему прибавляем единицу.

//Здесь ты инкремент поставил в суффиксе, а значит ты сначало увеличил значение х на 1 и только потом присвоил это значение переменной х1

int x2 = ++x1; //Здесь переменная х2 должна принимать значение 12, так как х1 из формулы выше должна быть 11 и к нему прибавляем единицу

//Опять инкремент в начале, значит ты сначала меняешь значение переменной х1 с 11 на 12 и присваиваешь это значение х2

int x3 = ++x2; //Здесь переменная х3 должна принимать значение 13, так как х2 из формулы выше должна быть 12 и к нему прибавляем единицу

//Тоже самое

Где я не так понимаю?

==================Вопрос================

2. ***МОЖЕШЬ ПРОЧИТАТЬ СРАЗУ ВОПРОС В КОНЦЕ!!!!!!!!***

**Методы строк**

Класс String имеет множество встроенных методов для обработки строк. Разберём некоторые из них.

Общий синтаксис для всех методов выглядит так:

stringName.nameOfMethod();

Здесь метод nameOfMethod применяется к строке stringName и вызывается с помощью круглых скобок. Внутри круглых скобок в некоторых методах должны быть обязательные аргументы, а в некоторых других - нет.

1. Длина строки (количество символов).

str.length();      // возвращает длину строки str (количество символов, включая пробелы)

String word = "Java is strong";

int x = word.length();

System.out.println(x); // 14

 2. Сравнение строк.

В Java строки нельзя сравнивать с помощью операторов сравнения, таких как == и !=. Вернее, сравнивать-то можно, но результат Вас удивит ;) Для корректного сравнения строк необходимо использовать специальный метод:

str1.equals(str2); // Сравнивает строки str1 и str2

Этот метод является булевым, то есть возвращает true, если строки равны, и false, если нет.

String word1 = "Java";

String word2 = "Python";

System.out.println(word1.equals(word2)); // false

String word3 = "Ja";

String word4 = "va";

boolean result = word1.equals(word3 + word4);

System.out.println(result); // true

 3. Получение индекса элемента в строке.

Метод **indexOf()** ищет в строке заданный символ (или строку), и возвращает  индекс его первого вхождения. Если элемент не найден, метод возвращает -1.

String word = "abracadabra";

int x = word.indexOf('b');

System.out.println(x); // 1

int y = word.indexOf('Z');

System.out.println(y); // -1

 4. Получение элемента строки по его индексу.

Для этого используется метод **charAt()**:

String word = "abracadabra";

char letter\_0 = word.charAt(0);

System.out.println(letter\_0); // a

char letter\_4 = word.charAt(4);

System.out.println(letter\_4); // c

Обратите внимание - метод возвращает значение типа char, а не String. Индексация, как обычно, начинается с нуля.

 5. Проверка строки на пустоту.

Метод **isEmpty()** является весьма полезным инструментом. Он возвращает false, если строка содержит какие-либо элементы (пробел - тоже элемент), и true - если строка пустая, т.е. не содержит ни одного элемента.

String str1 = "Hubba Bubba";

String str2 = " ";

String str3 = "";

boolean value1 = str1.isEmpty(); // false

boolean value2 = str2.isEmpty(); // false

boolean value3 = str3.isEmpty(); // true

 6. Одна строка внутри другой

Чрезвычайно полезный метод **contains()** проверяет, содержится ли одна строка внутри другой, и возвращает соответствующее логическое значение - true или false.

String str1 = "One Two Three";

String str2 = "One";

String str3 = "Four";

boolean value1 = str1.contains(str2); // true

boolean value1 = str1.contains(str3); // false

 7. Преобразование регистров.

Методы  **toUpperCase()** / **toLowerCase()**приводят всю строку в верхний и нижний регистр соответственно.

String s = "I'll be back";

System.out.println(s.toLowerCase()); // i'll be back

System.out.println(s.toUpperCase()); // I'LL BE BACK

 8. Представление числа в строковом формате.

Иногда полезно работать не с числом, а с его представлением в виде строки. Для этого Java предоставляет метод **toString()**. Чтобы использовать этот метод, нужно воспользоваться классом - обёрткой **Integer**. Сделать это можно разными путями.

int n = 12345; // Это число типа int

System.out.println(n); // 12345

String str1 = Integer.toString(n); // Это строка

System.out.println(str1); // 12345

Integer num = n; // Это число-объект класса Integer

System.out.println(num); // 12345

String str2 = num.toString(); // Это строка

System.out.println(str2); // 12345

 На самом деле, метод toString является мощным инструментом, который широко используется, однако мы пока ограничимся небольшой частью его функционала.

 9. Преобразование строки в число.

Метод, обратный предыдущему - **valueOf()**преобразует строку в число нужного типа.

String str = "12345";

Integer num = Integer.valueOf(str); // num - объект класса Integer

System.out.println(num); // 12345

int num1 = num; // num1 - переменная типа int

System.out.println(num1); // 12345

Поскольку оба метода (toString() и valueOf()) - это методы класса Integer, то именно с объектами этого класса их и необходимо использовать. Подробнее о типах переменных и их взаимоотношениях мы коснёмся в следующих модулях.

Есть и ещё один, более простой метод преобразования строки в число - **parseInt()**, также принадлежащий классу Integer.

String str = "12345";

int num = Integer.parseInt(str); //num - переменная типа int

System.out.println(num); //12345

Если нужно преобразовать строку в число с плавающей точкой, можно использовать соответствующий метод из класса Double.

String str = "12345";

double num = Double.parseDouble(str); //num - переменная типа double

System.out.println(num); //12345.0

 10. Создание подстроки.

Метод **substring()** возвращает новую строку, которая является подстрокой данной строки. Подстрока начинается с символа, заданного индексом, и продолжается до конца данной строки или до указанного индекса.

String str = "Добро пожаловать в мир Java!";

System.out.println(str.substring(6)); //пожаловать в мир Java!

System.out.println(str.substring(6, 15)); //пожаловат

 11. Замена элементов строки.

Несмотря на то, что строки в Java являются неизменяемыми (immutable), всё-таки их можно изменять с помощью специального метода **replace()**, который может заменить один символ на другой. Этот метод не изменяет строку, а собирает новую по заданным параметрам. Метод принимает два обязательных параметра - символ, подлежащий замене, и символ, на который его нужно заменить.

String str = "Добро пожаловать в мир Java!";

System.out.println(str.replace('о', 'А')); //ДАбрА пАжалАвать в мир Java!

Обратите внимание - метод работает с заменой символов (char), о чём говорят одинарные кавычки.

 Дополнительно методы строк можно поизучать в [официальной документации](https://docs.oracle.com/en/java/javase/16/docs/api/java.base/java/lang/String.html).

==================Вопрос================

Это все надо зазубривать или можно как то гдето подглядывать?

==================Вопрос================

==================Ответ================

Не надо зазубривать, просто примерно помни что такие есть а если надо точно узнать что тебе больше подходит то открываешь документацию <https://docs.oracle.com/javase/8/docs/api/java/lang/String.html>

Там список методов и что они делают. Так со всем, не надо за учивать все методы стандартных библиотек, памяти не хватит. Надо помнить где посмотреть. То что будешь чаще использовать и так запомнятся.

==================Ответ================