**План-конспект**

**к занятию №4**

**Тема: Работа со строками**

**Цель:**

Сформировать знания о строках в Java.

**Результат:**

Самостоятельно разработанные программы “99 бутылок лимонада” и “Переверни-ка!”.

**Программа занятия:**

1. Введение в строки. Класс “String”.
2. Методы обработки строковых данных.
3. Подготовка к практической части.
4. Практическая часть.
5. Анализ результатов практической части занятия.
6. Заключение (повтор важных моментов занятия).

**Структура организованной клубной деятельности:**

1. Знакомство с материалом (теория).
2. Описание результата (описание работы программы).
3. Разработка программы.
4. Тестирование (испытание программы).
5. Подведение итогов.

**Ход знакомства с материалами:**

1. Организационный момент.
2. Материал для ознакомления.

**Введение в строки. Класс “String”**

Строка - это последовательность символов. Для работы со ***строками*** в Java определен класс String, который содержит некоторые важные методы. По сути объект класса String представляет собой ссылку на область памяти, в которой эти символы находятся.  
Создать новую строку можно несколькими способами:

1. Просто присвойте экземпляру класса String строку.

String str = "Java";

// Строка может быть пустой. Для этого ей можно присвоить пустые кавычки

1. Воспользоваться одним из конструкторов класса String (их на самом деле несколько).

String str2 = **new** String(); *// пустая строка*

String str3 = **new** String(**new char**[] {**'h'**, **'e'**, **'l'**, **'l'**, **'o'**});

String str4 = **new** String(**new char**[]{**'w'**, **'e'**, **'l'**, **'c'**, **'o'**, **'m'**, **'e'**}, 3, 4);*//3 -начальный индекс, 4 -кол-во символов*

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

System.***out***.println(str3); *// hello*

System.***out***.println(str4); *// come*

Теперь начнем рассматривать некоторые методы этого полезного класса.

**Методы обработки строковых данных**

1. Чтобы узнать длину строки, можно воспользоваться методом .length()

String str = "Dima";

System.***out***.println(str.length()); // 4

1. С помощью метода .toCharArray() можно превратить строку в символьный массив:

String str = new String (new char[] {'h', 'i'});

char[] array = str.toCharArray();

1. Можно проверить содержит ли строка какие-либо символы или является пустой. Тут можно использовать .isEmpty() метод. Он вернет значение true, если строка пуста и false в противном случае.
2. Метод .concat(String str) объединяет 2 строки в одну.

String str1 = **"Java"**;

String str2 = **"Hello"**;

str2 = str2.concat(str1); *// HelloJava*

Однако, на практике чаще используется другой способ объединения строк (при помощи оператора “+”).

String str = "Здравствуй," + " слушатель" + "!" *//Здравствуй, слушатель!*

1. Также для соединения строк можно использовать метод .join(String delimiter, String str1, String str2), который позволяет объединить строки с учетом разделителя.

String str1 = **"Java"**;

String str2 = **"Hello"**;

String str3 = String.*join*(**" "**, str2, str1); *// Hello Java*

1. Метод .charAt(int index) вернет символ, расположенный по указанному индексу строки. (Важно, что индексы строк в Java начинаются с нуля!)

String str = "Я приветствую вас!";

char c = str.charAt(2); *//п*

1. Для чтения группы символов из строки следует использовать метод .getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin)**.** Он принимает такие параметры:

* srcBegin: индекс в строке, с которого начинается подстрока;
* srcEnd: индекс в строке, до которого идет чтение подстроки;
* dst: массив символов, в который будут извлекаться символы;
* dstBegin: индекс в массиве dst, с которого надо добавлять извлеченные из строки символы;

String str1 = **"Hello world"**;

**int** start = 6;

**int** end = 10;

**char**[] dst=**new char**[end - start];

str1.getChars(start, end, dst, 0);

System.***out***.println(dst); *// world*

1. Существуют два метода сравнения строк : .equals(String str) (с учетом регистра) и .equalsIgnoreCase(String str) (без учета регистра).

String str1 = **"Hello"**;

String str2 = **"hello"**;

System.***out***.println(str1.equals(str2)); *// false*

System.***out***.println(str1.equalsIgnoreCase(str2));

*// true*

1. Метод .indexOf(String str) возвращает индекс первого совпадения подстроки со строкой, а метод .lastIndexOf(String str) вернет индекс последнего совпадения. Если методы ничего не нашли, то оба возвращают -1:

String str = **"Hello Boris"**;

**int** index1 = str.indexOf(**'l'**); *// 2*

**int** index2 = str.indexOf(**"Bo"**); *//6*

**int** index3 = str.lastIndexOf(**'l'**); *//3*

1. Так же существуют методы .startWith(String str) и .endsWith(String str)

Оба этих метода возвращают boolean и позволяют узнать, начинается или заканчивается проверяемая строка с подстроки str.

String str = **"Hello"**;

**boolean** start = str.startsWith(**"He"**); *//true*

**boolean** end = str.endsWith(**"lo"**); *//true*

1. Метод .replace(String target, String replacement) позволяет заменить в строке нужную подстроку на другую:

String str = **"Hello world"**;

String replStr1 = str.replace(**'l'**, **'d'**); *// Heddo wordd*

String replStr2 = str.replace(**"Hello"**, **"Bye"**); *// Bye world*

1. Метод .trim() позволяет удалить пробелы в начале и конце строки:

String str = **" hello world"**;

str = str.trim(); *// hello world*

1. Метод .substring() возвращает подстроку, начиная с определенного индекса до конца или до определенного индекса:

String str = **"Hello world"**;

String substr1 = str.substring(6); *// world*

String substr2 = str.substring(3,5); *//lo*

1. Метод .toLowerCase() переводит строку в нижний регистр, а .toUpperCase() - верхний.

String str = **"Hello World"**;

System.***out***.println(str.toLowerCase()); *// hello world*

System.***out***.println(str.toUpperCase()); *// HELLO WORLD*

1. Метод .split(String regex) позволяет разделить строку на подстроки и возвращает массив объектов типа String:

String text = **"Thiskiskaknormalkline."**;

String[] words = text.split(**"k"**);

**for**(String word : words){

System.***out***.println(word); *//This*

*//is*

*//a*

*//normal*

*//line.*

1. Метод .valueOf() преобразует значение типа параметра в тип результата:

**double** d = 102939939.939;

**boolean** b = **true**;

**long** l = 1232874;

**char**[] arr = {**'a'**, **'b'**, **'c'**, **'d'**, **'e'**, **'f'**,**'g'** };

System.***out***.println(**"Return Value : "** + String.*valueOf*(d) );

System.***out***.println(**"Return Value : "** + String.*valueOf*(b) );

System.***out***.println(**"Return Value : "** + String.*valueOf*(l) );

System.***out***.println(**"Return Value : "** + String.*valueOf*(arr) );

**Вывод терминала:**

Return Value : 1.02939939939E8

Return Value : true

Return Value : 1232874

Return Value : abcdefg

Process finished with exit code 0

**Подготовка к практической части занятия**

Практическая часть состоит из 2 задач,которые предстоит решить детям.

1. Задача “Переверни-ка”, суть которой состоит в написании статического метода, который будет принимать на вход строку и возвращать ее перевернутый вид (Например, если на вход подана строка “hello”, то возвращена будет строка со значением “olleh” ).
2. Знаменитая задача “99 бутылок лимонада”. Она представляет собой традиционную песенку в Канаде.

Формат куплета является шаблонным и может быть представлен следующим образом:

**<количество> бутылок лимонада на стене**

**<количество> бутылок лимонада!**

**Возьми одну, пусти по кругу**

**<количество минус 1> бутылок лимонада на стене!**

Песня должна оканчиваться, когда количество бутылок будет 0 со словами “Нет бутылок на стене”. И следите за окончаниями : 19 бутылок, но 3 бутылки!