Частное учреждение образования
«Колледж бизнеса и права»

УТВЕРЖДАЮ
Ведущий
методист колледжа
_____Е.В. Паскал
«___»_____2021

Специальность: 2-40 01 01 «Программное обеспечение информационных технологий»	Учебная дисциплина: «Основы кроссплатформенного программирования»

Лабораторная работа № 6
Инструкционно-технологическая карта

Тема: «Работа со строками в языке Java»

Цель: Научиться создавать строки, искать символы в строке, заменять символы в строках, менять регистр, определять вхождение символа.

Время выполнения: 4 часа

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Изучить теоретические сведения;
2. Ответить на контрольные вопросы;
3. Откомпилировать примеры программ из раздела «Теоретические сведения»;
4. Выполнить ИДЗ.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ ДЛЯ ВЫПОЛНЕНИЯ РАБОТЫ

Строка — это упорядоченная последовательность символов. В Java строка является основным носителем текстовой информации. Для работы со строками здесь используются следующие классы: String, StringBuilder, StringBuffer.

Создание строк

Строка в Java является объектом, поэтому ее можно создать, как и любой другой объект, при помощи оператора new.

```
String str1 = new String ("Строка созданная при помощи оператора new");
```

Также строку можно создать при помощи литерала (фразы заключенной в кавычки) следующим образом.

```
String str2 = "Эта строка создана при помощи литерала.";
```

Обе строки, независимо от способа создания являются объектами — экземплярами класса String.

Важный момент: создание объектов при помощи литерала возможно только в классе String. Объекты любого другого класса при помощи литерала создать нельзя.

Можно также создать массив строк. Например, так:

String[] auto = {"Маша", "Петя", "Вася", "Коля"};

Конкатенация или слияние строк в Java

Для того, чтобы объединить несколько разных строк в одну, в Java можно использовать перегруженные (специально для объектов String) операторы «+» и «=+».

Еще один важный момент: операторы «+» и «=+», перегруженные для String, являются единственными перегруженными операторами в Java. Программист здесь не имеет возможности самостоятельно перегружать какие-либо операторы (как, например, в C++ и некоторых других языках).

Пример 1:

```
String str1 = "Мама ";
String str2 = "мыла ";
String str3 = "раму";

String result = str1 + str2 + str3;
System.out.print(result);
```

На консоль будет выведено «Мама мыла раму»

Пример 2:

```
String[] animals = {"Хаски","Морж"};// массив строк 1
String[] food = {"колбаски","корж"}; // массив строк 2

//составляем строки из элементов массивов и связующего слова
String result1 = animals[0] + " ест " + food[0];
String result2 = animals[1] + " ест " + food[1];

//выводим на консоль
System.out.println(result1);
System.out.println(result2);
```

Пример 3:

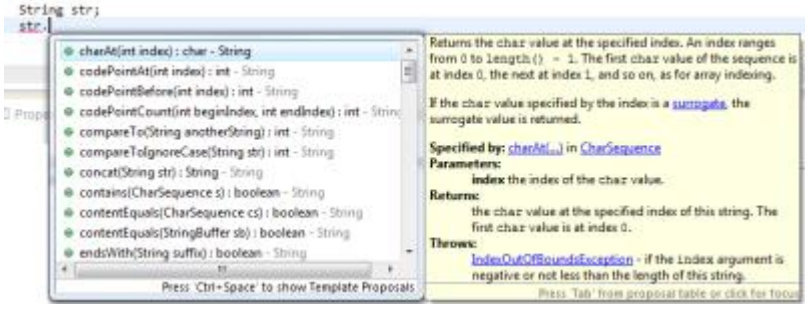
```
String[] auto = {"Волга", "Чайка", "Жигули"}; //задан массив строк
String result = "В гараже стоят машины: "; //задана строка

//прибавляем к строке элементы массива
for(int i = 0; i < auto.length; i++){
//если элемент не последний, разделяем запятой
if(i != auto.length-1 )
result += auto[i] + ", ";
//если последний, ставим после него точку
else
result += auto[i] + ".";
}
```

```
//выводим результат
System.out.print(result);
```

Наиболее употребительные методы класса String

При использовании IDE можно легко увидеть, какие методы есть у класса и получить подсказку по их использованию. На примере IDE Eclipse: для того, чтобы открыть список методов и быстро выбрать подходящий, нужно после имени переменной поставить точку и нажать комбинацию клавиш CTRL + Space (пробел). После этого появится окно, как на рисунке, где будут перечислены все доступные методы класса.



При выборе метода из этого списка, справа (или слева) появится желтое окно с подсказкой по его использованию. При помощи нажатия Enter или двойного клика мыши метод можно вставить в ваш код, не прибегая к ручному набору.

Также после имени переменной и точки можно начать набирать вручную имя метода и после введения нескольких первых букв нажать CTRL + Space (пробел). При этом, если метод, начинающийся на эти буквы один, то он автоматически подставится в код, если методов несколько, то откроется окно, как на рисунке, где будут перечислены только те методы, которые начинаются с этих введенных вами букв.

Далее рассмотрим методы, которые чаще всего используются при работе со строками. Некоторые задачи можно решить и без применения этих методов, но их знание значительно облегчает процесс программирования. В дальнейшем описании, первое слово, которое стоит перед названием метода — тип значения, которое возникнет в результате работы метода (значение, которое метод возвращает).

Конкатенация

String concat(String str) — производит ту же конкатенацию, что была описана выше, но использование этого метода из класса String положительно влияет на производительность и скорость программы. На небольших примерах это незаметно и не существенно, но в более серьезных приложениях стоит использовать этот метод. Результатом работы метода будет строка. Параметр, который нужно передавать в метод для конкатенации — тоже строка, о чем нам говорит значение в скобках (String str).

Перепишем пример 2, при помощи concat():

```
String[] animals = {"Хаски","Морж"};// массив строк 1
String[] food = {"колбаски","корж"}; // массив строк 2

//составляем строки из элементов массивов и связующего слова
String result1 = animals[0].concat(" ест ").concat(food[0]);
String result2 = animals[1].concat(" ест ").concat(food[1]);

//выводим на консоль
System.out.println(result1);
System.out.println(result2);
```

Определение количества символов в строке

Для того чтобы определить количество символов в строке, используется метод length.

int length() — возвращает длину строки. Длина равна количеству символов Unicode в строке.

```
String str = "Строка из букв, цифр 492 и специальных символов %*;№?";
int length = str.length();
System.out.println("Длина строки = " + length);
```

Извлечение символов из строки

Если нам требуется узнать, какой символ находиться в строке на конкретной позиции, можем использовать метод charAt.

char charAt(int index) — возвращает символ, находящийся по указанному индексу в строке. Результатом работы метода будет символ типа char. Параметр, который передается в метод — целое число. Первый символ в строке, подобно массивам, имеет индекс 0.

Пример 5: определить последний символ в строке.

```
String str = "Последний символ в этой строке - о";
int last = str.length()-1;//длина строки - 1, так как отсчет начинается с 0
char ch = str.charAt(last);
System.out.println(ch);
```

Если мы хотим работать со строкой, как с массивом символов, можем конвертировать строку в массив при помощи метода toCharArray.

char[] toCharArray() — преобразует строку в новый массив символов.

Пример 6: поменять в строке символы пробела на точки при помощи преобразования в массив символов (для этой задачи есть более простое решение, нежели преобразование в массив, но об этом чуть позже).

Примечание: в данном случае мы не сможем использовать метод charAt. При помощи этого метода мы бы смогли только найти пробелы в строке, но не поменять их.

```
String str = "1 000 000 000";
//преобразовываем строку в массив
char[] chArray = str.toCharArray();
//перебираем все элементы массива
for(int i = 0; i<chArray.length; i++){
//находим пробел
if(chArray[i] == ' '){
```

```
//заменяем на точку
chArray[i] = '.';
} }
//выводим результат
System.out.println(chArray);
```

Извлечение подстроки из строки

String substring(int beginIndex, int endIndex) или substring(int beginIndex) — возвращает новую строку, которая является подстрокой используемой строки. В параметрах метода нужно указать индекс строки, с которого начинается подстрока и индекс, которым заканчивается. Также возможно указывать только начальный индекс. В этом случае будет возвращена подстрока от начального индекса и до конца строки.

```
String s = "www.mysite.com";
String name = s.substring(4, s.length()-4);
System.out.println(name); // на консоль выведет "mysite"

String domain = s.substring(4);
System.out.println(domain); // на консоль выведет "mysite.com"
```

Разбиение строк

Для разбиения строк на части используется метод String[] split(String regex), который разбивает строку на основании заданного регулярного выражения.

```
String isbn = "978-3-16-148410-0";
String[] isbnParts = isbn.split("-");

System.out.println("префикс EAN.UCC: " + isbnParts[0]);
System.out.println("номер регистрационной группы: " + isbnParts[1]);
System.out.println("номер регистранта: " + isbnParts[2]);
System.out.println("номер издания: " + isbnParts[3]);
System.out.println("контрольная цифра: " + isbnParts[4]);
```

Поиск в строке

boolean contains(CharSequence s) — проверяет, содержит ли строка заданную последовательность символов и возвращает true или false.

```
String s = "www.mysite.com";
boolean isContain1 = s.contains("mysite");
System.out.println(isContain1);// нашел - выведет true
boolean isContain2 = s.contains("mysite.ru");
System.out.println(isContain2);// не нашел - выведет false
```

boolean endsWith(String suffix) — проверяет завершается ли строка определенными символами и возвращает true или false.

```
String s = "www.mysite.com";
//проверяем заканчивается ли строка суффиксом "com"
boolean isComEnding = s.endsWith("com");
System.out.println(isComEnding);//выведет true
//проверяем заканчивается ли строка суффиксом "ru"
boolean isRuEnding = s.contains("ru");
System.out.println(isRuEnding);//выведет false
```

boolean startsWith(String prefix) или startsWith(String prefix, int toffset) — проверяет, начинается ли строка с определенных символов. Во втором случае можно указать позицию с которой необходимо начать поиск префикса.

```
String s = "www.mysite.com";

//Проверяем, начинается ли адрес с www
boolean isWWW = s.startsWith("www");

if(isWWW){
/* Если да, проверяем начинается ли имя сайта
с "my". Поскольку адрес начинается с www
проверку начинаем с 4 символа*/
boolean isNameStarts = s.startsWith("my", 4);
}else{
/* Если нет, проверяем начинается ли имя сайта
с "my". Поскольку адрес не начинается с www
проверку производим с начала строки*/
boolean isNameStarts = s.startsWith("my");
}
```

int indexOf(int ch), indexOf(int ch, int fromIndex), indexOf(String str), indexOf(String str, int fromIndex) — метод indexOf применяется для поиска первого вхождения указанного символа в строке или первого вхождения указанной подстроки. Поиск также можно произвести с указанием позиции в строке от которой нужно начинать искать. Для поиска нужно указать соответствующие параметры. Метод возвращает число соответствующее индексу первого вхождения символа или подстроки. В случае отсутствия указанного символа или подстроки в строке, будет возвращена -1.

```
String data = "name:Igor\nsurname:Kolashnikov\nage:14\ntime:14:55";
//разбиваем строку на несколько подстрок на основании
// встречаемого символа новой строки
String[]lines=data.split("\n");

//проходим каждую подстроку
for (String line : lines){
//находим индекс первого вхождения символа ":" в подстроке
int pos = line.indexOf(":");
//вычленяем имя атрибута из подстроки
String attributeName= line.substring(0,pos);
//вычленяем значение атрибута
String value = line.substring(pos+1,line.length());
//вывод на экран вычлененных значений в нужном нам формате.
System.out.println(attributeName + " - " + value);
}
```

int lastIndexOf(int ch), lastIndexOf(int ch, int fromIndex), lastIndexOf(String str), lastIndexOf(String str, int fromIndex) — аналогично предыдущему случаю, только ищется последнее вхождение символа или подстроки в строке.

Модификация строк

Модификация строк не является модификацией как таковой. Дело в том, что объекты класса String после создания уже нельзя изменять. Но можно создать копию строки с изменениями. Именно это и делают следующие методы.

toLowerCase() — преобразовать строку в нижний регистр;
toUpperCase() — преобразовать строку в верхний регистр;
trim() — отсечь на концах строки пустые символы;

Пример 13

```
String str = " Я помню ЧУДНОЕ мгновенье ";

//убрали символы пробела в начале и конце строки
str = str.trim();

//я помню чудное мгновенье
System.out.println(str.toLowerCase());

//Я ПОМНЮ ЧУДНОЕ МГНОВЕНЬЕ
System.out.println(str.toUpperCase());
```

String replace(char oldChar, char newChar), replace(CharSequence target, CharSequence replacement) — замена в строке одного символа или подстроки на другой символ или подстроку.

Вспомним пример 6, где нужно было поменять в строке символы пробела на точки и перепишем его с использованием replace:

```
String str = "1 000 000 000";
String newStr = str.replace(" ", ".");
System.out.println(newStr);
```

Сравнение строк

boolean equals(Object anObject) — проверяет идентичность строк. Возвращает true только в том случае, если в строках представлена одинаковая последовательность символов одной величены.

Пример 14

```
String str = "Я помню ЧУДНОЕ мгновенье";
String str2 = "я помню чудное мгновенье";
//строки не идентичны
System.out.println(str.equals(str2)); //false
//строки идентичны после перевода первой строки
//в нижний регистр
System.out.println(str.toLowerCase().equals(str2)); // true
```

int compareTo(String anotherString) — так же проверяет идентичность строк, однако, в отличии от метода equals возвращает:

- нулевое значение, если строки равны,
- целое отрицательное число, если первая строка предшествует второй

- целое положительное число, если первая строка следует за второй

Данный метод предназначен для упорядочивания строк. Он позволяет сравнить строки между собой и определить предшествующую строку. Для того, чтобы реализовать такое сравнение метод сравнивает числовые значения букв.

Для сравнения строк без учета регистра символов используется функция int compareToIgnoreCase(String str)

System.out.println("маша".compareToIgnoreCase("Миша")); //-8

Как мы видим, при сравнивании «маша» с «Миша» мы снова получаем отрицательное значение, то есть «маша» предшествует имени «Миша».

Пример 16. Использование printf для форматирования в Java

Метод printf() принадлежит классу PrintStream, который отвечает за вывод информации. Уже знакомые нам методы print() и println() также являются методами класса PrintStream.

Метод printf определен следующим образом:

printf(String format, Object... args)

Первый аргумент format это строка, определяющая шаблон, согласно которому будет происходить форматирование. Для ее записи существуют определенные правила.

В предыдущем примере формат был «%4d», где d означает вывод десятичного целого числа, а 4 — означает то, что если количество знаков в числе меньше, чем 4, то оно будет спереди дополнено пробелами на недостающее (до 4-х) количество знаков (тем самым подвинуто вправо).

Для наглядности приведем еще один пример, который выводит столбиком несколько чисел.

System.out.printf("%6d%n%6d%n%6d%n%6d%n%6d%n%6d",666666, 55555, 4444, 333, 22, 1);

На консоль будет выведено:

```
666666
 55555
  4444
   333
    22
     1
```

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое строка?
2. Как создается строка?
3. Как выглядит массив строк?
4. Что такое конкатенация?
5. Как происходит слияние строк?

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Задание 1.

Вариант 1. Напишите метод, который принимает в качестве параметра любую строку, например “I like Java!!!”.

Вариант 2. Распечатать последний символ строки. Используем метод String.charAt()

Вариант 3. Проверить, заканчивается ли ваша строка подстрокой “!!!”. Используем метод String.endsWith().

Вариант 4. Проверить, начинается ли ваша строка подстрокой “I like”. Используем метод String.startsWith().

Вариант 5. Проверить, содержит ли ваша строка подстроку “Java”. Используем метод String.contains().

Вариант 6. Найти позицию подстроки “Java” в строке “I like Java!!!”.

Вариант 7. Заменить все символы “a” на “o”.

Вариант 8. Преобразуйте строку к верхнему регистру.

Вариант 9. Преобразуйте строку к нижнему регистру.

Вариант 10. Вырезать строку Java с помощью метода String.substring().

Вариант 11. Замените символ “=” на слово “равно”. Используйте методы StringBuilder.replace().

Задание 2.

Даны строки разной длины (длина - четное число), необходимо вернуть ее два средних знака: "string" → "ri", "code" → "od", "Practice"→"ct".

Задание 3.

Создать класс Employee, у которого есть переменные класса - fullname, salary. Создать массив содержащий несколько объектов этого типа. Создать класс Report, который будет содержать статический метод generateReport, в котором выводится информация о зарплате всех сотрудников. Используйте форматировании строк. Пусть salary будет выровнено по правому краю, десятичное значение имеет 2 знака после запятой. Выделить под фамилию 15 символов.

Задание 4.

Вариант 1. Дана строка “Versions: Java 5, Java 6, Java 7, Java 8, Java 12.” Найти все подстроки "Java X" и распечатать их.

Вариант 2. Дана строка 'aAXa aeffa aGha aza ax23a a3sSa'. Найти и вывести строки следующего вида: по краям стоят буквы 'a', а между ними - маленькие латинские буквы, не затронув остальных.

Вариант 3. Дана строка 'aAXa aeffa aGha aza ax23a a3sSa'. Найти и вывести строки следующего вида: по краям стоят буквы 'a', а между ними - маленькие и большие латинские буквы, не затронув остальных.

Вариант 4. Дана строка 'aAXa aeffa aGha aza ax23a a3sSa'. Найти и вывести строки следующего вида: по краям стоят буквы 'a', а между ними - маленькие латинские буквы и цифры, не затронув остальных.

Вариант 5. Дана строка 'ааа ббб ёёё ззз ййй ААА БББ ЁЁЁ ЗЗЗ ЙЙЙ'. Найти и вывести все слова по шаблону: любая кириллическая буква любое количество раз.

Вариант 6. Дана строка 'аеееа аееа аеа аха ахха аххха'. Найти и вывести строки следующего вида: по краям стоят буквы 'а', а между ними - или буква 'е' любое количество раз или по краям стоят буквы 'а', а между ними - буква 'х' любое количество раз.

Вариант 7. Дана строка 'аеееа аееа аеа аха ахха аххха'. Найти и вывести строки следующего вида: по краям стоят буквы 'а', а между ними - или буква 'е' два раза или буква 'х' любое количество раз.

Вариант 8. Предложение состоит из нескольких слов, разделенных пробелами. Например: "One two three раз два три one1 two2 123 ". Найти количество слов, содержащих только символы латинского алфавита.

Задание 5.

Предложение состоит из нескольких слов, например: "Если есть хвосты по дз, начните с 1 не сданного задания. 123 324 111 4554". Среди слов, состоящих только из цифр, найти слово-палиндром.

ДОМАШНЕЕ ЗАДАНИЕ

Индивидуальное задание

1. ЛИТЕРАТУРА

И. Н. Блинов В. С. Романчик, Java, Четыре четверти, 2020

Преподаватель А.С.Кибисова

Рассмотрено на заседании цикловой комиссии программного обеспечения информационных технологий
Протокол № ____ от «___» _____ 2021
Председатель ЦК _____В.Ю.Михалевич