Обработка строк

Класс String – это самый частоиспользуемый класс в Java, он предназначен для хранения набора (массива) символов.

Состояние объектов класса String невозможно изменить после создания объекта (объекты класса являются неизменяемыми).

Внутреннее устройство класса String

```
public final class String{
    private final char value[]; // Массив символов.
    private final int offset; // Смещение от начала массива.
    private final int count; // Количество символов.
    private int hash; // Хэш-код строки.
}
```

Создание объекта строки

Создать объект строка в Java можно с помощью оператора new или строкового литерала (символы, ограниченные двойными кавычками).

```
public class Task {
    public static void main (String[] args)
    {
        String str1 = "Java";
        String str2 = new String(new char[]{'J', 'A', 'V','A'});

        System.out.println(str1);
        System.out.print(str2);
    }
}
```

Объект строки создается в специально отведенном месте памяти, называемом ***пул строк****. При повторном использовании литерала, новый объект не создается, а в переменную присваивается ссылка на ранее созданный объект.*

При создании объекта строки оператором ***new всегда создается новый объект.***

Статические методы класса String

format – метод возвращает строку, в которой спецификаторы формата заменены на значения параметров в методе.

valueOf – возвращает строковое представление значений примитивных типов.

```
public class Task {
    public static void main (String[] args)
    {
        double a = 1.6;
        int b = 7;
        double result = a+b;
        System.out.print(String.format("Число %f + %d = %f", a,b,result));

        String value = String.valueOf(3);
        System.out.println(value); // 3.
        System.out.println(value+7); // 37.
        System.out.println(3+7); // 10.
    }
}
```

Спецификаторы формата

Спецификатор	Выполняемое форматирование
%a	Шестнадцатеричное значение с плавающей точкой
%b	Логическое (булево) значение аргумента
%c	**Символьное представление аргумента**
%d	**Десятичное целое значение аргумента**
%h	Хэш-код аргумента
%e	Экспоненциальное представление аргумента
%f	**Десятичное значение с плавающей точкой**
%g	Выбирает более короткое представление из двух: %e или %f
%o	Восьмеричное целое значение аргумента
%n	**Вставка символа новой строки**
%s	**Строковое представление аргумента**
%t	Время и дата
%x	Шестнадцатеричное целое значение аргумента
%%	**Вставка знака %**

Escape-последовательности

Escape-последовательность	Описание
\n	Новая строка
\t	Табуляция
\b	Символ «Backspace»
\r	Возврат каретки
\f	Перевод формата
\’	Одинарная кавычка
\”	Двойная кавычка
\\	Слеш влево

Все методы класса String не изменяют строку, в которой они вызываются, а возвращают ссылку на новый объект класса String.

Методы класса String

charAt – возвращает символ из строки, находящийся по индексу.

```
String str = "Java";
char c = str.charAt(2);
System.out.println(c); // v.
```

concat – возвращает конкатенацию (объединение двух строк в одну).

```
String str1 = "Java";
String str2 = "Hello";
str2 = str2.concat(str1); // HelloJava.
```

Методы класса String

length – возвращает количество символов в строке.

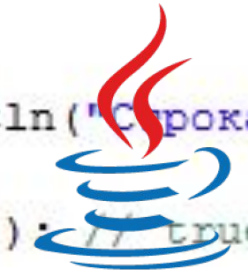
```
String str1 = "Java";
System.out.println(str1.length()); // 4
```

isEmpty – возвращает истину, если строка не содержит символов, иначе ложь. Работает быстрее, чем length.

```
String str1 = "";
if(str1.length() == 0) System.out.println("Строка пуста");

String str2 = null;
if(str2 == null) System.out.println("Строка пуста");

System.out.println(str1.isEmpty()); // true.
```



Методы класса String

contains – возвращает истину, если строка содержит хотя бы одно совпадение со сравниваемой строкой.

```
String str = "Java";
System.out.println(str.contains("v")); // true.
System.out.println(str.contains("j")); // false.
```

Методы класса String

startsWith – возвращает истину, если строка начинается с искомого символа или строки.

endsWith – возвращает истину, если строка заканчивается на искомый символ или строку.

```
String str = "Java.exe";
System.out.println(str.startsWith("Ja")); // true.
System.out.println(str.endsWith(".exe")); // true.
System.out.println(str.endsWith("ja")); // false.
```

Методы класса String

trim – возвращает строку с удаленными начальными и конечными пробелами.

toLowerCase – возвращает строку, в которой все заглавные символы исходной строки заменены на строчные.

toUpperCase – возвращает строку, в которой все строчные символы исходной строки заменены на заглавные.

Методы класса String

indexOf – возвращает индекс символа, с которого найдено первое совпадение с искомой строкой или символом. Поиск начинается с начала строки. Если совпадение не найдено, возвращает –1.

```
String str = "Java.exe";
int index1 = str.indexOf('.');
System.out.println(index1); // 4.
int index2 = str.indexOf("e");
System.out.println(index2); // 5.
```

Методы класса String

lastIndexOf – возвращает индекс символа, с которого найдено первое совпадение с искомой строкой или символом. Поиск начинается с конца сроки. Если совпадение не найдено, возвращает –1.

```
String str = "Java.exe";
int index1 = str.lastIndexOf("T");
System.out.println(index1); // -1.
int index2 = str.lastIndexOf("e");
System.out.println(index2); // 7.
```

Методы класса String

substring – возвращает часть строки из исходной.

```
String str = "Java.exe";
String substr1 = str.substring(2);
System.out.println(substr1); // va.exe.

String substr2 = str.substring(3,5);
System.out.println(substr2); // a.

String substr3 = str.substring(10,16);
System.out.println(substr3); // StringIndexOutOfBoundsException.
```

Методы класса String

replace – возвращает строку, заменяя в исходной строке символ или набор символов на другой символ или набор символов.

```
String str = "Java.exe";
String replStr1 = str.replace('a', 'A');
System.out.println(replStr1); // JAvA.exe

String replStr2 = str.replace("exe", "java");
System.out.println(replStr2); // JAvA.java
```

Методы класса String

split – позволяет разбить строку на подстроки по определенному разделителю. Разделитель – какой-нибудь символ или набор символов передается в качестве параметра в метод.

```
String text = "split - позволяет разбить строку на подстроки по определенному разделителю";
String[] words = text.split(" "); // Разделитель - пробел.
for(String word : words){
    System.out.println(word);
}
```

Задача 1

Подсчитать общее количество знаков «+», «-» и «», входящих во вводимую с клавиатуры строку.*

```
import java.util.Scanner;

public class Task {
    public static void main (String[] args)
    {
        System.out.print("Введите строку: ");
        Scanner scan = new Scanner(System.in);
        String text = scan.next();

        int count = 0;
        for(String c : text.split("")) {
            if(c.contains("+") || c.contains("-") || c.contains("*")) {
                count++;
            }
        }

        System.out.print(count);

    }
}
```

Задача 2

Определить количество вхождений заданной подстроки в строку. Ввод строки и подстроки организовать с клавиатуры.

Задача 2

```java
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println("Введите строку: ");
        String str = scan.nextLine();

        System.out.println("Введите подстроку: ");
        String subStr = scan.nextLine();

        String[] words = str.split( regex: " "); // Массив слок (разбивка предложения на слова).
        int count = 0; // Кол-во вхождений заданной подстроки в строку.

        for (String word : words) { // Перебор по массиву слов.
            if (word.contains(subStr)) { // Если подстрока содержится в исходной строке, то увеличиваем счетчик.
                count++;
            }
        }
        System.out.print("Количество вхождений заданной подстроки в строку: " + count);
    }
}
```

Задача 3

Дана строка, которая содержит имена пользователей, разделенные запятой – "Login1,LOgin2,login3,loGin4".

Необходимо разбить эту строку на массив строк (чтобы отдельно были логины), и перевести их все в нижний регистр.

Задача 4

Разработать программу, проверяющую, является ли введенное с клавиатуры слово (строка) палиндромом (читается одинаково в обе стороны).

Задача 4

```java
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        System.out.println("Введите строку: ");
        String str = scan.nextLine();
        boolean fl = false;

        for (int i = 0; i < str.length() / 2; i++) { // Просматриваем половину строки.
            // Сравниваем символ сначала строки и с конца.
            if (str.charAt(i) != str.charAt(str.length() - i - 1)) fl = false;
            else fl = true;
        }
        if (fl == true) System.out.print("Является палиндромом");
        else System.out.print("Не является палиндромом");
    }
}
```

Сравнение строк

Для сравнения строк используются методы equalsIgnoreCase() (без учета регистра) и equals() (с учетом регистра).

```
String str1 = "JaVa";
String str2 = "java";

System.out.println(str1.equals(str2)); // false.
System.out.println(str1.equalsIgnoreCase(str2)); // true.
```

Сравнение строк

Метод regionMatches() сравнивает отдельные подстроки в рамках двух строк.

```
boolean regionMatches(int toffset, String other, int oofset, int len)
boolean regionMatches(boolean ignoreCase, int toffset, String other, int oofset, int len)
```

***ignoreCase**: надо ли игнорировать регистр символов при сравнении.*

***toffset**: начальный индекс в вызывающей строке, с которого начнется сравнение.*

***other**: строка, с которой сравнивается вызывающая.*

***oofset**: начальный индекс в сравниваемой строке, с которого начнется сравнение.*

***len**: количество сравниваемых символов в обеих строках.*

Сравнение строк

Методы int compareTo(String str) и int compareToIgnoreCase(String str) сравнивают строки и позволяют узнать больше ли одна строка, чем другая или нет.

```
String str1 = "hello"; String str2 = "world"; String str3 = "hell";

System.out.println(str1.compareTo(str2)); // -15 - str1 меньше чем strt2
System.out.println(str1.compareTo(str3)); // 1 - str1 больше чем str3
```

Сравнение строк

При применении ***оператора сравнения*** *(==) для переменных* ***ссылочного типа*** *происходит сравнение* ***ссылок на объект***.

В связи с этим для сравнение ***идентичности*** *разных строк, следует использовать метод* ***equals()***, *который* **сравнивает строки на эквивалентность**.

Домашнее задание по строкам

Задача 1. *Перевернуть каждое четное слово в строке.*

Задача 2. *Подсчитать общее количество символов '+' и '-' и заменить каждый символ ';' на ',' и '.'.*

Задача 3. *Подсчитать количество букв в третьем слове.*