Строки

Символьная строка Java представляет собой последовательность символов в Unicode. стандартной библиотеке Java содержится класс **String**. Каждая символьная строка, заключенная в кавычки, представляет собой экземпляр класса **String**.

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

        String e = ""; // пустая строка
        String greeting = "Hello";
        // использовать конструктор (строка ниже) не рекомендуется
        String qqq = new String("qwerty");

        // substring
        String s1 = "GoodBye!";
        String s2 = "Day";
        String s = s1.substring(0, 4) + s2;
        System.out.println(s);

        // сложение строк
        String s3 = "I have " + 10 + " apples";
        System.out.println(s3);

        // объединение строк join
        String s4 = String.join("*", "5", "10");
        System.out.println(s4);

        //В Java нельзя изменять отдельные символы в строке,
        // поэтому в документации объекты типа String называются
        // неизменяемыми.
        String s5 = "Hello!";
        s5 = s5.substring(0, 3) + "p!";
        System.out.println(s5);

        // equals - сравнение строк
        //Для проверки символьных строк на равенство нельзя
        // применять операцию ==.
        // Эта операция  определяет, хранятся ли обе строки
        // в одной и той же области памяти.
        String s6 = "Hello!";
        System.out.println("Hello!".equals(s6));
        System.out.println("hello!".equalsIgnoreCase(s6));

        System.out.println("иногда требуется проверить - не является ли
символьная строка ни пустой, ни нулевой:");
        String s7 = null;
        System.out.println(s7 != null && s7.length() != 0);

    }
}
```

```
GoodDay
I have 10 apples
5*10
Help!
true
true
иногда требуется проверить - не является ли символьная строка ни пустой, ни
нулевой:
false
```

Регулярные выражения

Регулярное выражение – это шаблон для поиска строки в тексте. В Java исходным представлением этого шаблона всегда является строка, то есть объект класса String.

Метасимволы для поиска символьных классов

Метасимвол	Назначение
\d	цифровой символ
\D	нецифровой символ
\s	символ пробела
\S	непробельный символ
\w	буквенно-цифровой символ или знак подчёркивания
\W	любой символ, кроме буквенного, цифрового или знака подчёркивания
.	любой символ

```
        System.out.println("1234567".matches("\\d{5,}"));

        System.out.println("1234".matches("\\d{5,}"));
```

```
true
false
```

Метасимволы для поиска символов редактирования текста

Метасимвол	Назначение
\t	символ табуляции
\n	символ новой строки
\r	символ возврата каретки
\f	переход на новую страницу
\u 0085	символ следующей строки
\u 2028	символ разделения строк
\u 2029	символ разделения абзацев

```
        System.out.println("\t 1234".matches("\t\\s\\d{3,}"));

        System.out.println("\t ABCD".matches("\t\\s\\D{3,5}"));
```

```
true
true
```

Метасимволы для группировки символов

Метасимвол	Назначение
[абв]	любой из перечисленных (а,б, или в)
[^абв]	любой, кроме перечисленных (не а,б, в)
[a-zA-Z]	слияние диапазонов (латинские символы от a до z без учета регистра)
[a-d[m-p]]	объединение символов (от a до d и от m до p)
[a-z&&[def]]	пересечение символов (символы d,e,f)
[a-z&&[^bc]]	вычитание символов (символы a, d-z)

```
System.out.println("aBCd".matches("[a-zA-Z]{3,}"));

System.out.println("abc 123".matches("[a-z0-9]{3,}\\s[a-z0-9]{3,}"));
```

```
true
true
```

Метасимволы для обозначения количества символов

Метасимвол / квантификатор	Назначение
?	один или отсутствует
*****	ноль или более раз
+	один или более раз
{n}	n раз
{n,}	n раз и более
{n,m}	не менее n раз и не более m раз

```
System.out.println("A".matches("[a-zA-Z]?"));

System.out.println(" 7".matches("[a-z]*\\s+[1-7]*"));
```

```
true
true
```

```
System.out.println("097".matches("\\t?\\s?[0-9]+"));

System.out.println(" 097".matches("\\t?\\s?[0-9]+"));

System.out.println("\t 097".matches("\\t?\\s?[0-9]+"));
```

```
true
true
true
```

Метасимволы для поиска совпадений границ строк

Метасимвол	Назначение
^	начало строки
$	конец строки
\b	граница слова
\B	не граница слова
\A	начало ввода
\G	конец предыдущего совпадения
\Z	конец ввода
\z	конец ввода

```
System.out.println("Axyz".matches("^[A-C]{1,}xyz"));

System.out.println("ABCxyz".matches("\\b[A-C]{1,}xyz"));

System.out.println("Axyz".matches("\\b[A-C]{1,}\\B[xyz]+"));
```

```
true
true
true
```

Примеры с группировкой регулярных выражений и выбором a || b – a или b

```
System.out.println("ABC".matches("[A-C]{1,}||[1-3]{1,}"));

System.out.println("111".matches("[A-C]{1,}||[1-3]{1,}"));
```

```
true
true
```

[A-D&&[^BC]]– от A до D , исключая BC

```
System.out.println("ABCD".matches("[A-D&&[^BC]]{3,}"));

System.out.println("AADD".matches("[A-D&&[^BC]]{3,}"));
```

```
false
true
```

?– необязательное выражение

```
System.out.println("1.3".matches("^\\d+(\\.\\d+)?"));

System.out.println("1".matches("^\\d+(\\.\\d+)?"));
```

```
false
true
```

Задания на составление регулярных выражений

1. буква 'a', любой символ, буква 'b'.
2. буква 'a', буква 'b' любое количество раз, буква 'a'.
3. буква 'a', буква 'b' один раз или ни одного, буква 'a'.
4. по краям слова стоят буквы **'a'**.
5. только цифры любое количество и русские буквы любое количество.
6. цвет в шестнадцатеричном формате
7. ip-адрес
8. имя файла в Java
9. имя переменной в Java
10. фамилия и инициалы

Чтобы создать регулярное выражение Java, нужно:

1. написать его в виде строки с учётом синтаксиса регулярных выражений;
2. скомпилировать эту строку в регулярное выражение;

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;

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

        // строка
        String s = "xxxzzzyyy";

        // скомпилированное регулярное выражение
        Pattern p1 = Pattern.compile("[x-z]+");

        //В классе Pattern также определен метод matcher(String input),
        // который в качестве параметра принимает строку,
        // где надо проводить поиск, и возвращает объект Matcher:
        Matcher matcher = p1.matcher(s);

        //у объекта Matcher вызывается метод matches()
        // для поиска соответствий шаблону
        System.out.println(matcher.matches());

    }
}
```

```
true
```

Методы replaceFirst и replaceAll

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;

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

        String REGEX = "dog";
        String INPUT = "The dog says meow. " + "All dogs say meow.";
        String REPLACE = "cat";

        Pattern p = Pattern.compile(REGEX);

        Matcher m = p.matcher(INPUT);
        INPUT = m.replaceAll(REPLACE);
        System.out.println(INPUT);    }}
```

```
The cat says meow. All cats say meow.
```

Основные операции со строками

В языке Java класс String содержит свыше 50 методов.

Извлечение символов и подстрок

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

```
String str = "Hello world!";
int start = 6;
int end = 11;
char[] dst=new char[end - start];
str.getChars(start, end, dst, 0);
System.out.println(dst); // world
```

Сравнение строк

```
String str1 = "Hello";
String str2 = "hello";

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

```
//метод сравнивает 3 символа с 6-го индекса первой строки ("wor")
// и 3 символа со 2-го индекса второй строки ("wor")
String str1 = "Hello world";
String str2 = "I work";
boolean result = str1.regionMatches(6, str2, 2, 3);
System.out.println(result); // true
```

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

//Если возвращаемое значение больше 0, то первая строка больше второй,
// если меньше нуля, то, наоборот, вторая больше первой.
// Если строки равны, то возвращается 0.

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

Поиск в строке

```
String str = "Hello world";
int index1 = str.indexOf('l'); // 2
int index2 = str.indexOf("wo"); //6
int index3 = str.lastIndexOf('l'); //9
```

Начинается с подстроки и заканчивается подстрокой

```
String str = "myfile.exe";
boolean start = str.startsWith("my"); //true
boolean end = str.endsWith("exe"); //true
```

Замена в строке

```
String str = "Hello world";
String replStr1 = str.replace('l', 'd'); // Heddo wordd
String replStr2 = str.replace("Hello", "Bye"); // Bye world
```

Обрезка строки

```
String str = "  hello world  ";
str = str.trim(); // hello world
```

Выделение подстроки

```
String str = "Hello world";
String substr1 = str.substring(6); // world
String substr2 = str.substring(3,5); //lo
```

Изменение регистра

```
String str = "Hello World";
System.out.println(str.toLowerCase()); // hello world
System.out.println(str.toUpperCase()); // HELLO WORLD
```

Разбиение на подстроки по определенному разделителю

```
String text = "FIFA will never regret it";
String[] words = text.split(" ");
for(String word : words){
    System.out.println(word);
```

Задания

1. Задана строка, состоящая из нескольких слов. Вывести на консоль слова из строки, состоящие из 4 букв.
2. Заменить все символы латинского алфавита в слове на соответствующие символы русского алфавита.
3. Ввести n слов с консоли. Найти количество слов, содержащих только символы латинского алфавита.
4. Ввести n слов с консоли. Найти количество слов, содержащих только цифры.
5. Ввести n слов с консоли. Среди слов, состоящих только из цифр, найти слово-палиндром.