

Массивы и строки в Java

Алексей Владыкин

26 сентября 2012

1 Массивы

2 Строки

- Работа с кодировками
- Регулярные выражения

1 Массивы

2 Строки

- Работа с кодировками
- Регулярные выражения

Объявление

- Массив обозначается квадратными скобками

```
int [] numbers;  
String [] args;  
boolean bits [];  
char [] letters, digits;  
float rates [], maxRate;
```

Создание

- Массив создается оператором `new`
- Все элементы массива инициализируются нулями
- Размер массива фиксируется в момент создания

```
int [] numbers = new int [100];  
String [] args = new String [1];  
boolean [] bits = new boolean [0];
```

Инициализация

- Можно перечислить значения всех элементов при создании массива

```
int[] numbers = new int[] {1, 2, 3, 4, 5};
boolean bits = new boolean[] {true, true, false};

// this works only in variable declaration
char[] digits = {
    '0', '1', '2', '3', '4',
    '5', '6', '7', '8', '9'};
```

Индексация

- Элементы индексируются с нуля
- Длина массива доступна как `array.length`
- При выходе за границы массива бросается исключение

```
int [] numbers = {1, 2, 3, 4, 5};  
// numbers.length -> 5  
// numbers[0] -> 1  
// numbers[1] -> 2  
// numbers[4] -> 5  
// numbers[5] -> ArrayIndexOutOfBoundsException
```

Многомерные массивы

- Многомерный массив — это массив массивов

```
int [][] matrix0;  
int [][] matrix1 = new int [2][2];  
int [][] matrix2 = {{1, 2}, {3, 4}};  
int [] row = matrix2[0]  
  
// matrix2[1][1] -> 4  
// row[0] -> 1
```

Многомерные массивы

- Разрешены ступенчатые массивы

```
int [][] triangle = {  
    {1, 2, 3, 4, 5},  
    {6, 7, 8, 9},  
    {10, 11, 12},  
    {13, 14},  
    {15}};  
  
// triangle.length -> 5  
// triangle[0].length -> 5  
// triangle[4].length -> 1
```

Представление в памяти

- Одномерный массив занимает непрерывный участок памяти
- Двумерный массив занимает $n + 1$ участок в памяти, где n — первая размерность

```
int [][] a = new int [10] [1000];  
int [][] b = new int [1000] [10];
```

java.util.Arrays

- copyOf, copyOfRange
- equals, deepEquals
- fill
- sort
- binarySearch
- toString, deepToString

- java.lang.System.arraycopy

Varargs

- Специальный синтаксис для массива аргументов
- Поддерживается с Java 5

```
int max(int [] numbers);  
// usage: max(new int [] {1, 2, 3, 4});  
  
int max(int... numbers);  
// usage: max(1, 2, 3, 4);
```

1 Массивы

2 Строки

- Работа с кодировками
- Регулярные выражения

- Последовательность символов произвольной длины
- Класс `java.lang.String`
- Не то же, что массив символов
- Никаких нулевых символов в конце, длина хранится отдельно

Создание строк

- Строковые литералы

```
String zeros = "\u0000\u0000";  
String hello = "Hello";  
String specialChars = "\r\n\t\"";  
String unicodeEscapes = "\u0101\u2134\u03ff";
```

- Создание из массива символов

```
char[] charArray = {'a', 'b', 'c'};  
String string = new String(charArray);
```

Доступ к содержимому строки

- `int` `length()`
- `char` `charAt(int index)`
- `char []` `toCharArray()`
- `String` `substring(int beginIndex)`
`String` `substring(int beginIndex, int endIndex)`

Сравнение строк

- Оператор `==` сравнивает ссылки, а не содержимое строки
- `boolean equals(Object anObject)`
`boolean equalsIgnoreCase(String anotherString)`
- `int compareTo(String anotherString)`
`int compareToIgnoreCase(String anotherString)`

Операции

- Строки неизменяемы
- `boolean` `startsWith(String prefix)`
- `boolean` `endsWith(String suffix)`
- `int` `indexOf(String str)`
- `int` `lastIndexOf(String str)`

Операции

- Строки неизменяемы
- `String trim()`
- `String replace(char oldChar, char newChar)`
- `String toLowerCase()`
- `String toUpperCase()`

Конкатенация строк

- `String concat(String str)`
- Оператор `+`

```
String helloWorld = "Hello" + " World!";
```

- `java.lang.StringBuilder`

```
StringBuilder buf = new StringBuilder();  
buf.append("Hello");  
buf.append(" World");  
buf.append('!');  
String result = buf.toString();
```

- JVM использует для строк кодировку UTF-16 (каждый символ занимает один или два char'a)
- Кодировка строк в памяти не зависит от платформы или локали
- Можно конвертировать данные в другие кодировки

Преобразование кодировок

- `byte[] getBytes(String charsetName)`

```
String str = "test";  
byte[] ascii = str.getBytes("US-ASCII");  
// ascii -> {116, 101, 115, 116}
```

- `String(byte bytes[], String charsetName)`

```
byte[] ascii = {116, 101, 115, 116};  
String str = new String(ascii, "US-ASCII");  
// str -> "test"
```

- Способ задания шаблонов строк для поиска и замены
- Регулярные выражения поддерживаются в стандартной библиотеке Java
- `boolean matches(String regex)`
- `String[] split(String regex)`
- `String replaceAll(String regex, String replacement)`
- `String replaceFirst(String regex, String replacement)`

Язык регулярных выражений

<code>x</code>	конкретный символ
<code>[a-z]</code>	диапазон символов
<code>[^a-z]</code>	любой символ вне диапазона
<code>.</code>	любой символ
<code>re+</code>	одно или более повторений
<code>re*</code>	ноль или более повторений
<code>re?</code>	ноль или одно повторение
<code>(re)</code>	группировка
<code>re1 re2</code>	выбор

Пример

```
String str = "a, b, c,d, e";  
String[] items = str.split(", *");  
  
// str.split(", [\t ]*");  
// str.split("\\s*, \\s*");
```

Пример

```
String str = "abracadabra";  
String regex = "(ab|ac|ad|r.?)+";  
  
// str.matches(regex) -> true
```

Пример

```
String str = "(aa)(bb)(cccc)";  
String regex = "\\(([^)]*)\\)";  
String result = str.replaceAll(regex, "$1");  
// result -> "aabbcccc"
```

Что сегодня узнали

- Как создавать и использовать массивы
- Как создавать и использовать строки
- Как конвертировать строки из одной кодировки в другую
- Что такое регулярные выражения и какие задачи можно решать с их помощью