Java

Массивы / java.util.Arrays

Лекция #6 B

Пустовалова О.Г.
доцент. каф. мат.мод.
ИММиКН ЮФУ

Содержание

- java.util.Arrays
- Копирование массивов copyOf() copyOfRange()
- Массив в виде строки toString() deepToString()
- Проверка на идентичность equals() deepEquals()
- Поиск в массиве binarySearch()

java.util.Arrays

java.util.Arrays. Основные методы

Класс **java.util.Arrays** предназначен для работы с массивами.

- **copyOf()** – предназначен для копирования массива
- **copyOfRange()** – копирует часть массива
- **toString()** – позволяет получить все элементы в виде одной строки
- **sort()** — сортирует массив методом quick sort
- **binarySearch()** – ищет элемент методом бинарного поиска
- **fill()** – заполняет массив переданным значением (удобно использовать, если нам необходимо значение по умолчанию для массива)
- **equals()** – проверяет на идентичность массивы
- **deepEquals()** – проверяет на идентичность массивы массивов
- **asList()** – возвращает массив как коллекцию

java.util.Arrays. binarySearch

```
static int  binarySearch(char[] a, char key)

static int binarySearch(char[] a, int fromIndex, int toIndex, char

static int binarySearch(double[] a, double key)

static int binarySearch(double[] a, int fromIndex, int toIndex, do

static int binarySearch(float[] a, float key)

static int binarySearch(float[] a, int fromIndex, int toIndex, float

static int binarySearch(int[] a, int key)

static int binarySearch(int[] a, int fromIndex, int toIndex, int key)

...
```

Выполняет поиск указанного значения в указанном массиве с помощью алгоритма бинарного поиска.

java.util.Arrays. binarySearch

```
// initializing unsorted array
Object arr[] = {10,2,22,69};

// sorting array
Arrays.sort(arr);

// entering the value to be searched
int searchVal = 22;

int retVal = Arrays.binarySearch(arr,searchVal);

System.out.println("The index of element 22 is : " + retVal);
```

The index of element 22 is : 2

java.util.Arrays. copyOf

static boolean[] copyOf(**boolean**[] original, **int** newLength)

static byte[] copyOf(**byte**[] original, **int** newLength)

static char[] copyOf(**char**[] original, **int** newLength)

static double[] copyOf(**double**[] original, **int** newLength)

static float[] copyOf(**float**[] original, **int** newLength)

static int[] copyOf(**int**[] original, **int** newLength)

static long[] copyOf(**long**[] original, **int** newLength)

static short[] copyOf(**short**[] original, **int** newLength)

Этот метод копирует указанный массив, усекая или заполняя его нулями (при необходимости), чтобы копия имела заданную длину.

Копирование массивов

Если требуется скопировать все элементы одного массива в другой, для этого следует вызвать метод **copyOf** () из класса **Arrays**

```
import java.util.Arrays;
...

int[] A = { 1, 2, 3, 4, 5};

int[] B = Arrays.copyOf(A, A.length);

// увеличить массив в 2 раза
// новые элементы заполняются нулями
A = Arrays.copyOf(A, 2*A.length);


System.out.println(Arrays.toString(A)) ;
```

[1, 2, 3, 4, 5, 0, 0, 0, 0, 0]

java.util.Arrays. copyOf

```
import java.util.Arrays;
public class Main {
    public static void main(String[] args) {
        // intializing an array arr1
        int[] arr1 = new int[] {1, 2, 3};

        // copying array arr1 to arr2 with newlength as 5
        int[] arr2 = Arrays.copyOf(arr1, 5);
        arr2[3] = 11;
        arr2[4] = 22;

        // printing the array arr2
        System.out.println("Printing new array:");
        for (int i = 0; i < arr2.length; i++) {
            System.out.println(arr2[i]);
        }
    }}
```

1 2 3 11 22

java.util.Arrays. copyOfRange

static boolean[] copyOfRange(**boolean**[] original, **int** from, **int** to)

static byte[] copyOfRange(**byte**[] original, **int** from, **int** to)

static char[] copyOfRange(**char**[] original, **int** from, **int** to)

static double[] copyOfRange(**double**[] original, **int** from, **int** to)

static float[] copyOfRange(**float**[] original, **int** from, **int** to)

static int[] copyOfRange(**int**[] original, **int** from, **int** to)

static long[] copyOfRange(**long**[] original, **int** from, **int** to)

static short[] copyOfRange(**short**[] original, **int** from, **int** to)

Этот метод копирует указанный диапазон указанного массива в новый массив.

java.util.Arrays. copyOfRange

```
import java.util.Arrays;
public class Main {

    public static void main(String[] args) {

        // intializing an array arr1
        long[] arr1 = new long[] {0,1,2,3,4,5};

        // copying array arr1 to arr2 with range of index from 2 to 5
        long[] arr2 = Arrays.copyOfRange(arr1, 2, 4);

        // printing the array arr2
        System.out.println("Printing new array:");
        for (int i = 0; i < arr2.length; i++) {
            System.out.println(arr2[i]);
        }
    }}
```

```
2 3
```

java.util.Arrays. toString

static String toString(**boolean**[] a)

static String toString(**byte**[] a)

static String toString(**char**[] a)

static String toString(**double**[] a)

static String toString(**float**[] a)

static String toString(**int**[] a)

static String toString(**long**[] a)

static String toString(Object[] a)

static String toString(**short**[] a)

Этот метод возвращает строковое представление содержимого указанного массива.

java.util.Arrays. toString

```
import java.util.Arrays;
public class Main {

    public static void main(String[] args) {

// initializing int array
    int[] i1 = new int[] { 1, 2, 3 };

    // let us print all the elements available in list
    System.out.println("The array is:");
    for (int number : i1) {
        System.out.println("Number = " + number);
    }

    System.out.println("The string representation of array is:");
    System.out.println(Arrays.toString(i1));
}}
```

```
1
2
3
```

```
[ 1 2 3 ]
```

java.util.Arrays. Вывод многомерных массивов. deepToString()

Метод **deepToString**() позволяет выводить многомерные массивы

```
String[][] s = {
        {"a(1,1)", "a(1,2)"},
        {"a(2,1)", "a(2,2)"},
        };
System.out.println(Arrays.deepToString(s));
```

[[a(1,1), a(1,2)], [a(2,1), a(2,2)]]

java.util.Arrays. deepToString

```
int[][][] a3 ={{{1, 2, 3}, {4 ,5, 6}},
               {{10, 20, 30 }, {40, 50, 60}}};
```

System.*out*.println(Arrays.*deepToString*(a3));

[[[1, 2, 3], [4, 5, 6]], [[10, 20, 30], [40, 50, 60]]]

System.*out*.println(a3[0][1][0]); → **4**

System.*out*.println(a3[1][0][0]); → **10**

java.util.Arrays. hashCode

В классе Object, который является родительским классом для объектов java, определен метод **hashCode**(), позволяющий получить уникальный целый номер для данного объекта.

Функция hashCode() объекта Object возвращает целое число int, размер которого равен 4-м байтам и значение которого располагается в диапазоне от **-2 147 483 648** до **2 147 483 647**.

java.util.Arrays. hashCode

static int hashCode(**boolean**[] a)

static int hashCode(**byte**[] a)

static int hashCode(**char**[] a)

static int hashCode(**double**[] a)

static int hashCode(**float**[] a)

static int hashCode(**int**[] a)

static int hashCode(**long**[] a)

static int hashCode(Object[] a)

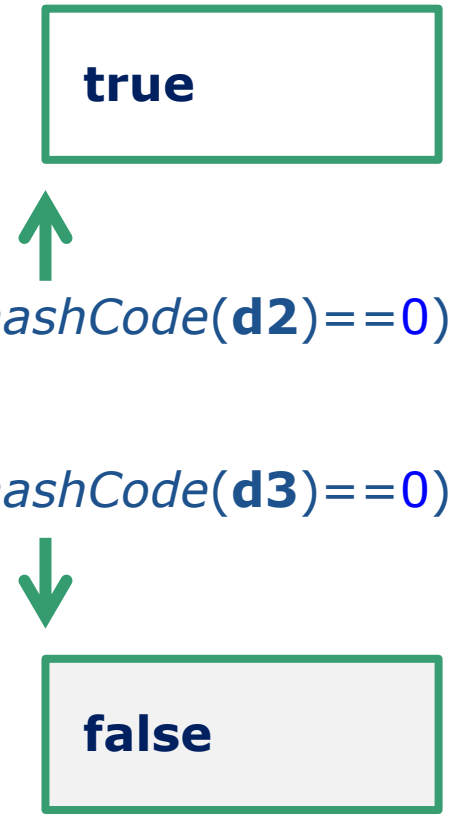
static int hashCode(**short**[] a)

Возвращает хэш-код, основанный на содержимом указанного массива.

Хэш-код объекта, hashCode

```
int d1[]={10};
int d2[]={10};
int d3[]={11};
System.out.println(Arrays.hashCode(d1) - Arrays.hashCode(d2)==0);

System.out.println(Arrays.hashCode(d1) - Arrays.hashCode(d3)==0);
```



Если хеш-коды разные, то и входные объекты разные.

java.util.Arrays. deepHashCode

```
int[][] a1 ={{1, 2}, {4 ,5}};
```

```
System.out.println(Arrays.deepHashCode(a1));
```
→ **32865**

```
System.out.println(Arrays.hashCode(a1));
```
→ **-1008904932**

Этот метод возвращает хэш-код, основанный на" глубоком содержимом " указанного массива.

java.util.Arrays. equals

static boolean equals(**boolean**[] a, **boolean**[] a2)

static boolean equals(**byte**[] a, **byte**[] a2)

static boolean equals(**char**[] a, **char**[] a2)

static boolean equals(**double**[] a, **double**[] a2)

static boolean equals(**float**[] a, **float**[] a2)

static boolean equals(**int**[] a, **int**[] a2)

static boolean equals(**long**[] a, **long**[] a2)

static boolean equals(Object[] a, Object[] a2)

static boolean equals(**short**[] a, **short**[] a2)

Возвращает **true**, если два указанных массива значений равны друг другу.

java.util.Arrays. equals

Поэлементное сравнение массива **следует** выполнять с помощью метода **Arrays.equals**(a,a1).

```
int[] a1 = {1,2,3};
int[] a2 = {1,2,3};
System.out.println(Arrays.equals(a1, a2));
```

→ **true**

```
System.out.println(a1.equals(a2));
System.out.println(a1==a2);
```

→ **false**
false

java.util.Arrays. deepEquals

```
int[][] a1 ={{1, 2}, {4 ,5}};
int[][] a2 ={{1, 2}, {4 ,5}};
```

```
System.out.println(Arrays.deepEquals(a1,a2));
```

→ **true**

```
System.out.println(Arrays.equals(a1,a2));
```

→ **false**

Этот метод возвращает true, если два указанных массива глубоко равны друг другу.

java.util.Arrays. fill

```
static void fill(byte[] a, byte val)
static void fill(byte[] a, int fromIndex, int toIndex, byte val)
static void fill(char[] a, char val)
static void fill(char[] a, int fromIndex, int toIndex, char val)
static void fill(double[] a, double val)
static void fill(double[] a, int fromIndex, int toIndex, double val)
static void fill(float[] a, float val)
static void fill(float[] a, int fromIndex, int toIndex, float val)
static void fill(int[] a, int val)
static void fill(int[] a, int fromIndex, int toIndex, int val)
static void fill(long[] a, int fromIndex, int toIndex, long val)
static void fill(long[] a, long val)
static void fill(Object[] a, int fromIndex, int toIndex, Object val)
static void fill(Object[] a, Object val)
static void fill(short[] a, int fromIndex, int toIndex, short val)
static void fill(short[] a, short val)
```

Присваивает указанное значение каждому элементу *указанного диапазона* указанного массива.

java.util.Arrays. Заполнение массива. fill

Метод **fill**() дублирует одно заданное значение в каждом элементе массива.

```
Arrays.fill(arr, "0");
System.out.println(Arrays.toString(arr));

Arrays.fill(arr, 2, 3, "Yes");
System.out.println(Arrays.toString(arr));
```

[0, 0, 0, 0]

[0, 0, Yes, 0]

java.util.Arrays. sort

static void sort(**byte**[] a)

Sorts the specified array into ascending numerical order.

static void sort(**byte**[] a, **int** fromIndex, **int** toIndex)

Sorts the specified range of the array into ascending order.

static void sort(**char**[] a)

Sorts the specified array into ascending numerical order.

static void sort(**char**[] a, **int** fromIndex, **int** toIndex)

Sorts the specified range of the array into ascending order.

static void sort(**double**[] a)

Sorts the specified array into ascending numerical order.

static void sort(**double**[] a, **int** fromIndex, **int** toIndex)

Sorts the specified range of the array into ascending order.

static void sort(**float**[] a)

Sorts the specified array into ascending numerical order.

static void sort(**float**[] a, **int** fromIndex, **int** toIndex)

Sorts the specified range of the array into ascending order.

Сортирует указанный массив по возрастанию.

java.util.Arrays. sort

static void sort(**int**[] a)

Sorts the specified array into ascending numerical order.

static void sort(**int**[] a, **int** fromIndex, **int** toIndex)

Sorts the specified range of the array into ascending order.

static void sort(**long**[] a)

Sorts the specified array into ascending numerical order.

static void sort(**long**[] a, **int** fromIndex, **int** toIndex)

Sorts the specified range of the array into ascending order.

static void sort(Object[] a)

Sorts the specified array of objects into ascending order, according to the natural ordering of its elements.

static void sort(Object[] a, **int** fromIndex, **int** toIndex)

Sorts the specified range of the specified array of objects into ascending order, according to the natural ordering of its elements.

static void sort(**short**[] a)

Sorts the specified array into ascending numerical order.

Сортирует указанный массив по возрастанию.

java.util.Arrays. Сортировка массива

Сортировка (упорядочение по значениям) массива a производится методами **Arrays.sort**(a) и Arrays.sort(a,index1,index2).

```
import java.util.Arrays;
public class Main {
    public static void main(String[] args) {
        // элементы массива в произвольном порядке
        int[] numbers = new int[]{1, 7, 3, 5, 2, 6, 4};

        Arrays.sort(numbers);

// в цикле выводим все элементы массива по порядку
        for (int i = 0; i < numbers.length; i++)
            System.out.println(numbers[i]);
    }}
```

java.util.Arrays. Arrays.asList

Возвращает список фиксированного размера, заполненный указанным массивом.

```
import java.util.Arrays;
import java.util.List;

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

        // create an array of strings
        String a[] = new String[]{"abc","klm","xyz","pqr"};

        List list1 = Arrays.asList(a);

        // printing the list
        System.out.println("The list is:" + list1);
    }}
```

The list is:[abc, klm, xyz, pqr]

Спасибо за внимание!