

# Лекция 12

Java. Массивы.

# Одномерные массивы

**Одномерный массив** представляется в виде фиксированного числа однотипных элементов, которые располагаются в смежных ячейках памяти. Синтаксис объявления:

```
тип[] имя=new тип[размер] ;
```

Вначале указывается тип элементов массива, а после идентификатора типа следуют пустые квадратные скобки. Далее указывается имя массива, оператор присваивания, инструкция (оператор) **new**, снова тип элементов массива и в квадратных скобках размер массива (количество элементов в массиве). Например, командой

```
int[] nums=new int[20] ;
```

объявляется целочисленный массив **nums** из 20 элементов.

Или можно записать так:

```
int[] nums;  
nums=new int[20] ;
```

# Одномерные массивы

При объявлении переменной массива допускается указывать квадратные скобки либо после идентификатора типа, либо после имени массива. Например, вместо команды

```
int[] nums;  можно использовать команду  
int nums[];
```

Обращение к элементу одномерного массива осуществляется через имя массива с указанием в квадратных скобках индекса элемента.

```
nums[1]=25;
```

*Индексация* элементов массива начинается *с нуля*. Таким образом, ссылка на первый элемент массива **nums** будет иметь вид **nums[0]**. Если в массиве 20 элементов, то последний элемент массива имеет индекс 19, то есть инструкция обращения к элементу выглядит как **nums[19]**. Длину массива можно узнать с помощью свойства **length**:

```
int n=nums.length;
```

Тогда ссылка на последний элемент массива может быть записана как **nums[nums.length-1]**.

# Одномерные массивы

Java используется *автоматическая проверка на предмет выхода за пределы массива*. Поэтому если в коде выполняется обращение к несуществующему элементу массива, то возникает ошибка.

```
nums [90]=25 ;
```

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 90  
    at demo.main(demo.java:6)
```

При объявлении массива для него выделяется память. В Java элементы массива автоматически инициализируются с нулевыми значениями – выделенные ячейки обнуляются, а значения этих обнуленных ячеек интерпретируются в зависимости от типа массива. Но на такую автоматическую инициализацию полагаться не стоит. Разумно инициализировать элементы массива в явном виде. Для этого используют оператор цикла или задают список значений элементов при объявлении массива.

# Одномерные массивы

Например:

```
int[] data={3,8,1,7};
```

Размер массива и значения элементов определяются автоматически в соответствии с количеством элементов в списке значений. В данном случае создается целочисленный массив из четырех элементов со значениями элементов 3, 8, 1 и 7. Или можно записать так:

```
int[] data; // объявление
```

```
data=new int[]{3,8,1,7}; // создание массива из 4 целых чисел,  
// а ссылка на этот массив присваивается переменной data
```

Примеры:

```
double[] x1={1.1, 2.2, 3.3};
```

```
double x2[];
```

```
    x2=new double[]{-1.1, -2.2, -3.3};
```

```
float[] x3={1.1f, 2.2f, 3.3f};
```

```
char[] x4={'1','2','3','4'};
```

```
boolean[] x5={true, false, false, true};
```

# Одномерные массивы

```
public class demo3 {  
    public static void main(String[] args) {  
        int i, n;                // Целочисленные переменные  
        int[] data;              // Объявление массива  
  
        data = new int[] { 3, 8, 1, 7 }; // Первый массив  
        n = data.length;           // Размер массива  
  
        int[] nums = new int[n];    // Второй массив  
        // Заполнение второго массива:  
        for (i = 0; i < nums.length; i++) {  
            nums[i] = 2 * data[i] - 3;  
            System.out.println("nums[" + i + "]=" + nums[i]);  
        }  
    }  
}
```

Результат выполнения:

```
nums[0]=3  
nums[1]=13  
nums[2]=-1  
nums[3]=11
```

# Одномерные массивы

В программе объявляется и инициализируется массив **data** из четырех элементов. Длина массива вычисляется инструкцией **data.length**. Это значение записывается в целочисленную переменную **n**. Далее объявляется еще один целочисленный массив **nums**. Количество элементов в этом массиве определяется значением переменной **n**, поэтому совпадает с размером массива **data**. Заполнение массива **nums** выполняется с помощью оператора цикла. Значения элементов массива **nums** вычисляются на основе значений элементов массива **data**. Для отображения значений элементов массива **nums** использована команда

```
System.out.println("nums["+i+"]="+nums[i]).
```

**Особенность: длина массива задается один раз и изменить ее НЕЛЬЗЯ.**

# Одномерные массивы

Можно выполнять присваивание, т.е. одному массиву присвоить другой массив:

```
import java.util.Scanner;
public class demo {
    public static void main(String[] args) {
        int n;
        Scanner input = new Scanner(System.in);
        System.out.print("Введите целое число: ");
        n = input.nextInt();

        int x1[];
        x1 = new int[n];
        System.out.println("x1:");
        for (int i = 0; i < x1.length; i++)
            System.out.print(" " + x1[i]);
        System.out.println();

        System.out.println("new x1:");
        for (int i = 0; i < x1.length; i++) x1[i] = i * 2;
        for (int i = 0; i < x1.length; i++)
            System.out.print(" " + x1[i]);
        System.out.println();
    }
}
```



# Одномерные массивы

```
//int[] x2 = x1;  
int[] x2 = new int[x1.length];  
for (int i = 0; i < x2.length; i++)  
    x2[i] = x1[i];
```

```
x2[1] = 777;  
System.out.println("x2:");  
for (int i = 0; i < x2.length; i++)  
    System.out.print(" " + x2[i]);  
System.out.println();
```

```
System.out.println("new x1:");  
for (int i = 0; i < x1.length; i++)  
    System.out.print(" " + x1[i]);
```

```
}
```

```
}
```

```
Введите целое число: 10  
x1:  
0 0 0 0 0 0 0 0 0 0  
new x1:  
0 2 4 6 8 10 12 14 16 18  
x2:  
0 777 4 6 8 10 12 14 16 18  
new x1:  
0 777 4 6 8 10 12 14 16 18
```

```
Введите целое число: 10  
x1:  
0 0 0 0 0 0 0 0 0 0  
new x1:  
0 2 4 6 8 10 12 14 16 18  
x2:  
0 777 4 6 8 10 12 14 16 18  
new x1:  
0 2 4 6 8 10 12 14 16 18
```

# Одномерные массивы

Вывод, при выполнении присваивания

**x2 = x1;**

**x2** будет ссылаться на ту же область в памяти, куда указывает **x1!!!**

# Одномерные массивы, создание своего класса

Создадим свой класс – массив:

```
class MyArray{
    private int n;      // размер
    private int []arr;
                        // метод создания
    void create(int n) {
        this.n=n;
        arr=new int[n];
    }
    //метод заполнения случайными числами
    void rand() {
        for(int i=0; i<n; i++)
            arr[i]=(int) (Math.random()*100);
    }
    // метод печати массива
    void show() {
        for(int i=0;i<n;i++)
            System.out.print(arr[i]+" ");
    }
}
```

# Одномерные массивы, создание своего класса

```
// индекс максимального
    int ind_max() {
        int imax=0;
        for(int i=0;i<n;i++)
            if(arr[i]>arr[imax]) imax=i;
        return imax;
    }
// печать одного элемента массива с индексом ind
    void show_ind(int ind) {
        System.out.print(arr[ind]);
    }
} // конец класса
```

# Одномерные массивы, создание своего класса

```
public class Demo {  
    public static void main(String[] args) {  
        MyArray obj=new MyArray(); // создание объекта  
        obj.create(7);           // создание массива из 7 элем.  
        obj.rand();              // заполнение массива  
        obj.show();              // печать массива  
  
        // поиск максимального и его печать  
        int t=obj.ind_max();  
        System.out.println("\nНомер максимального="+t+" ");  
        System.out.print("\nМаксимальный=");  
        obj.show_ind(t);  
    }  
}
```

Завершено: 100% (1/1) [1/1]

60 15 15 63 52 86 28  
Номер максимального=5  
  
Максимальный=86

Завершено: 100% (1/1) [1/1]

17 85 8 2 80 56 11  
Номер максимального=1  
  
Максимальный=85

Завершено: 100% (1/1) [1/1]

26 43 35 65 47 1 34  
Номер максимального=3  
  
Максимальный=65

# Одномерные массивы – встроенный класс

Для работы с массивами в **Java** есть класс `java.util.Arrays`.

С массивами чаще всего проделывают следующие операции:

- заполнение элементами (инициализация),
- извлечение элемента (по номеру),
- сортировка
- поиск.

# Одномерные массивы – встроенный класс

`void sort(int[] myArray, int fromIndex, int toIndex)` - метод сортирует массив целых чисел или его часть по возрастанию.

`int binarySearch(int[] myArray, int fromIndex, int toIndex, int key)` - метод ищет элемент **key** в уже отсортированном массиве **myArray** или подмассиве, начиная с **fromIndex** и до **toIndex**. Если элемент найден, метод возвращает его индекс, если нет – то  $< 0$ .

Метод `String toString(int[] myArray)` преобразовывает массив в строку.

Дело в том, что в **Java** массивы не переопределяют `toString()`. Это значит, что при попытке вывести целый массив (а не по элементам) на экран непосредственно (`System.out.println(myArray)`), вы получите имя класса и шестнадцатеричный хэш-код массива.

# Одномерные массивы – встроенный класс

Рассмотрим на примере:

```
import java.util.Arrays;
public class Demo {
    public static void main(String[] args) {
        //объявляем и инициализируем массив
        int[] array = {1, 5, -4, 3, -7, 10, -12};
        System.out.println(array); // вывод массива на экран без
        //метода toString - получаем 16-ричное число
        System.out.println("правильная печать:\n" +
                           Arrays.toString(array));
        Arrays.sort(array, 0, 4); //сортируем массив от 0 до 4
        System.out.println("После сортировки от 0-го до 4-го");
        System.out.println(Arrays.toString(array)); //ВЫВОД

        Arrays.sort(array, 0, array.length); //сортируем весь
        System.out.println("После сортировки");
        System.out.println(Arrays.toString(array)); //ВЫВОД
```



# Одномерные массивы – встроенный класс

```
// ищем key - число 5 в отсортированном массиве
int key = Arrays.binarySearch(array, 5);
//метод binarySearch возвращает индекс элемента-искомого
// числа в отсортированном массиве
if (key>=0)
    System.out.println("Индекс искомого элемента=« + key);
    //распечатываем индекс искомого числа
else System.out.println("нет такого элемента");

//ищем key - число 5 в отсортированном массиве от 0 до 4
key = Arrays.binarySearch(array,0,4,5);
if (key>=0)
    System.out.println("Индекс искомого элемента в
                        диапазоне от 0-го до 4-го =" +key);
else System.out.println("нет такого элемента в диапазоне
                        от 0-го до 4-го");
}
}
```

```
[I@1eb44e46
правильная печать:
[1, 5, -4, 3, -7, 10, -12]
После сортировки от 0-го до 4-го
[-4, 1, 3, 5, -7, 10, -12]
После сортировки
[-12, -7, -4, 1, 3, 5, 10]
Индекс искомого элемента=5
нет такого элемента в диапазоне от 0-го до 4-го
```

# Главное о массивах

Главные характеристики массива:

- тип помещённых в него данных, имя и длина. Последнее решается при инициализации (выделении памяти под массив), первые два параметра определяются при объявлении массива.
- Размер массива (количество ячеек) нужно определять в **int**.
- Изменить длину массива после его создания нельзя.
- Доступ к элементу массива можно получить по его индексу.
- В массивах элементы нумеруются с нуля.
- После создания массив заполнен значениями по умолчанию.

# Двумерные массивы

Синтаксис объявления двумерного массива:

```
тип[][] имя=new тип[размер][размер];
```

или

```
тип[][] имя;
```

```
имя=new тип[размер][размер];
```

Хотя двумерный массив и удобно представлять как таблицу, но самом деле двумерный массив в **Java** — это одномерный массив, элементами которого являются переменные массива. Каждая такая переменная ссылается на одномерный массив.

Пример:

```
double[][] data;
```

```
data=new double[3][4];
```

Создается двумерный массив с элементами типа **double**. В массиве 3 строки и 4 столбца, а ссылка на массив записывается в переменную **data**.

# Двумерные массивы

Обращение к элементам двумерного массива выполняется в следующем формате: указывается имя массива, в квадратных скобках – первый индекс элемента, в других квадратных скобках – второй индекс элемента массива. Индексация по всем размерностям начинается с нуля. Например, ссылка `data[0][3]` является обращением к элементу массива `data` с индексами 0 и 3, и это элемент в первой строке и в четвертом столбце.

`data[0][3] = -23.3;`

Для инициализации двумерного массива используют вложенные операторы цикла или список, состоящий из списков значений. Каждый такой внутренний список определяет значения элементов массива в строке. Примеры инициализации двумерного массива с помощью списка:

```
double[][] data={{0.1, 0.2, 0.3},{0.4, 0.5, 0.6}};  
int nums[][]={{1, 2, 3},{4, 5}};
```

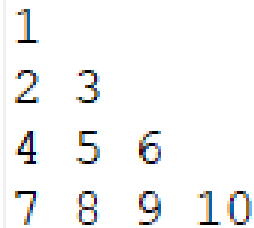
# Двумерные массивы - пример1

```
public class Demo {  
    public static void main(String[] args) {  
        int i, j, n = 3, val = 1;  
        // Создание двумерного массива:  
        int[][] nums = new int[n - 1][n];  
  
        for (i = 0; i < n - 1; i++) {  
            for (j = 0; j < n; j++) {  
                nums[i][j] = val++; // Заполнение элементов  
                System.out.print(nums[i][j] + " "); //и печать  
            }  
            System.out.println(); // Переход к новой строке  
        }  
    }  
}
```

1	2	3
4	5	6

# Двумерные массивы - пример2

```
public class Demo {  
    public static void main(String[] args) {  
        int i, j, val = 1;  
        // Создание массива (второй размер не указан):  
        int[][] nums = new int[4][];  
        // Цикл для создания треугольного массива:  
        for (i = 0; i < nums.length; i++)  
            nums[i] = new int[i + 1]; // Создание строки в массиве  
        // Заполнение массива:  
        for (i = 0; i < nums.length; i++) {  
            for (j = 0; j < nums[i].length; j++) {  
                nums[i][j] = val++; // Значение элемента массива  
                System.out.print(nums[i][j] + " "); // Отображение  
            }  
            System.out.println(); // Переход к новой строке  
        }  
    }  
}
```



```
1  
2 3  
4 5 6  
7 8 9 10
```

# Двумерные массивы - пример3 (транспонирование)

```
import static java.lang.Math.random;  
public class Demo {  
    public static void main(String[] args) {  
        int n = 3, m=4;  
        int[][] A = new int[n][m]; // исходный массив  
        int[][] B = new int[m][n]; // результ. массив  
        int i, j;  
        System.out.println("Матрица до транспонирования:");  
        // Заполнение матрицы случайными числами:  
        for (i = 0; i < n; i++) {  
            for (j = 0; j < m; j++) {  
                A[i][j] = (int) (20 * random());  
                System.out.print(A[i][j] + " "); // и печать  
            }  
            System.out.println(); // Переход к новой строке  
        }  
    }  
}
```

$$A^T = \begin{pmatrix} 2 & -3 \\ 4 & 8 \\ 5 & 7 \end{pmatrix}^T = \begin{pmatrix} 2 & 4 & 5 \\ -3 & 8 & 7 \end{pmatrix}$$

# Двумерные массивы - пример3 (транспонирование)

```
// Транспонирование матрицы:
for (i = 0; i < m; i++)
    for (j = 0; j < n; j++)
        B[i][j] = A[j][i];

// Отображение результата:
System.out.println("Результат");
for (i = 0; i < m; i++) {
    for (j = 0; j < n; j++)
        System.out.print(B[i][j] + " ");
    System.out.println(); // Переход к новой строке
}
}
```

Исходная матрица:

3	15	2	10
18	14	8	12
11	14	1	17

Результат

3	18	11
15	14	14
2	8	1
10	12	17



# Двумерные массивы, создание своего класса

Напишем свой класс для работы с двумерными массивами:

```
class MyMatrix {  
    private int n; // Размер матрицы  
    private int[][] matrix; // Ссылка на массив  
  
    // Метод для создания матрицы:  
    void create(int n) {  
        this.n = n;  
        matrix = new int[n][n];  
    }  
  
    // Метод для заполнения матрицы случайными числами  
    void rand() {  
        int i, j;  
        for (i = 0; i < n; i++)  
            for (j = 0; j < n; j++)  
                matrix[i][j] = (int) (Math.random() * 10);  
    }  
}
```

# Двумерные массивы, создание своего класса

// Метод для вычисления следа матрицы

```
int trace() {  
    int i, s = 0;  
    for (i = 0; i < n; i++)  
        s += matrix[i][i];  
    return s;  
}
```

// Метод для отображения матрицы

```
void show() {  
    int i, j;  
    for (i = 0; i < n; i++) {  
        for (j = 0; j < n; j++)  
            System.out.print(matrix[i][j] + " ");  
        System.out.println();  
    }  
}
```

```
}
```

# Двумерные массивы, создание своего класса

```
public class Demo {  
    public static void main(String[] args) {  
  
        MyMatrix obj = new MyMatrix(); // Создание объекта  
        obj.create(3); // Создание матрицы  
        obj.show(); // Печать матрицы  
        obj.rand(); // Заполнение случайными числами  
        System.out.println("Случайные числа:");  
        obj.show(); // Печать матрицы  
  
        // Вычисление следа матрицы:  
        System.out.println("След матрицы = " + obj.trace());  
    }  
}
```

```
0 0 0  
0 0 0  
0 0 0  
Случайные числа:  
3 9 0  
5 6 8  
5 9 2  
След матрицы = 11
```

```
0 0 0  
0 0 0  
0 0 0  
Случайные числа:  
9 3 7  
1 6 5  
7 8 5  
След матрицы = 20
```