Программирование на Java
Тема 1.2 Введение в Java

Массивы

- **Описание массива:**

1. **Объявление (declaration)**

```
double[] a, b; // Создаются ссылки на
                    // массивы типа double
 int i = 0, ar[], k = -1;
```

2. **Определение (installation)**

```
a = new int[5];
b = new double[100];
ar = new int[50];
```

3. **Инициализация (initialization)**

```
 a[0] = 0.01; a[1] = -3.4; a[2] = 2.89;
a[3] = 4.5; a[4] = -6.7;
for (int i = 0; i < 100; i++) b[i] = 1.0 /i;
for (int i = 0; i < 50; i++) ar[i] = 2 * i + 1;
```

Массивы

- *Массив* — это совокупность переменных одного типа, хранящихся в смежных ячейках оперативной памяти.
- Массивы в языке Java относятся к ссылочным типам и их описание характерно для ссылочных типов. Описание производится в три этапа.
- Первый этап — *объявление* (declaration). На этом этапе определяется только переменная типа *ссылка* (reference) *на массив,* содержащая тип массива. Для этого записывается имя типа элементов массива, квадратными скобками указывается, что объявляется ссылка на массив, а не простая переменная, и перечисляются имена переменных типа ссылка. Можно поставить квадратные скобки и непосредственно после имени. Это удобно делать среди определений обычных переменных.
- Второй этап — *определение* (installation). На этом этапе указывается количество элементов массива, называемое его *длиной,* выделяется место для массива в оперативной памяти, переменная-ссылка получает адрес массива. Все эти действия производятся еще одной операцией языка Java — операцией new *тип,* выделяющей участок в оперативной памяти для объекта указанного в операции типа и возвращающей в качестве результата адрес этого участка.
- Индексы элементов массивов всегда начинаются с 0. Индексы можно задавать любыми целочисленными выражениями, кроме типа long.Исполняющая система Java следит за тем, чтобы значения этих выражений не выходили за границы длины массива.

Массивы

- **Описание массива:**

```
// совмещение объявления и определения массива
   int[] ar = new int[5];
  double[] a = new double[5], b = new double[100];

// совмещение объявления, определения  и
// инициализации массива

  int[] ar = new int[]{5, -12, 0, 17, 0};
  double[] a = {0.01, -3.4, 2.89, 4.5, -6.7};

// создание безымянного массива
  System.out.println(new char[] {'H', 'e', 'l', 'l',
'o'});
                        a = b;
                        arr = null;
```

Массивы

- Первые два этапа можно совместить.
- Можно сразу задать и начальные значения, записав их в фигурных скобках через запятую в виде констант или константных выражений. При этом даже необязательно указывать количество элементов массива, оно будет равно количеству начальных значений;
- Можно даже создать безымянный массив, сразу же используя результат операции new.
- Ссылка на массив не является частью описанного массива, ее можно перебросить на другой массив того же типа операцией присваивания. Например, после присваивания a = b обе ссылки a и b указывают на один и тот же массив из 100 вещественных переменных типа double и содержат один и тот же адрес.
- Ссылке можно присвоить "пустое" значение null, не указывающее ни на какой адрес оперативной памяти.
- Кроме простой операции присваивания, со ссылками можно производить еще только сравнения на равенство, например, a = b, и неравенство, a != b. При этом сопоставляются адреса, содержащиеся в ссылках, мы можем узнать, не ссылаются ли они на один и тот же массив.
- **Массивы в Java всегда определяются динамически, хотя ссылки на них задаются статически.**

Массивы

- **Длина массива:**

```
double aMin = a[0], aMax = aMin;
for (int i = 1; i < a.length; i++)
 { if (a[i] < aMin) aMin = a[i];
   if (a[i] > aMax) aMax = a[i];
  }
```

- **Последний элемент массива:** `a[a.length - 1],`

Массивы

- Кроме ссылки на массив, для каждого массива автоматически определяется целая константа с именем length. Она равна длине массива. Для каждого массива имя этой константы уточняется именем массива через точку. Так, после наших определений, константа a.length равна 5, константа b. length равна 100, a ar. length равна 50.
- Элементы массива — это обыкновенные переменные своего типа, с ними можно производить все операции, допустимые для этого типа.
- **Массив символов в Java не является строкой, даже если он заканчивается нуль-символом ' \uOOOO'.**

Массивы

- **Пример:**

```
// замена отрицательных элементов массива  на максимальный
public class FindReplace
{
 public static void main(String[] args)
   {
      int a[] = new int[] {5,10,0,-5,16,-2};

      int max = a[0];
       //поиск max-элемента
       for(int i = 0; i < a.length; i++)
         if(max < a[i]) max = a[i];
       for(int i = 0; i < a.length; i++)
        { //замена
         if( a[i] < 0 )  a[i] = max;
         System.out.println("a[" + i + "]= " + a[i]);
        }
   }
}
```

```
a[0]= 5
a[1]= 10
a[2]= 0
a[3]= 16
a[4]= 16
a[5]= 16
```

Описание многомерных массивов

1. **Объявление:**

```
char[][] c; или char c[]c[]; или char c[][];
```

2. **Определение:**

```
c = new char[3][]; //c.length равна 3
c[0] = new char[2]; //c[0].length равна 2
c[1] = new char[4]; //c[1].length равна 4
c[2] = new char[3]; //c[2].length равна 3
```

3. **Инициализация:**

```
c [0][0] = 'a', c[0][1] = 'r',
c[1][0] = 'r',c[1][1] = 'a', c[1][2] = 'y' и т.д.
//сокращенная форма описания двумерного массива
int[][] d = new int[3][4];
int[][] inds = {{1, 2, 3}, {4, 5, 6}};
```

Массивы

- Элементами массивов в Java могут быть снова массивы. Например, c — массив, состоящий из трех элементов-массивов.
- **Двумерный массив в Java не обязан быть прямоугольным.**

Пример: Вывод таблицы умножения Пифагора

```
public class Pith
{ public static final int N = 10; // !!! N - константа
  public static void main (String [] args)
  { int pithagor_table[][]=new int[N][N];
     for (int i=1; i<N; i++)
       {  for (int j=1; j<N; j++)
        { pithagor_table[i][j]=i*j;
          System.out.format("%3d",pithagor_table[i][j]);
         }
         System.out.println();
        }
   }
 }
```

```
  1   2   3   4   5   6   7   8   9
  2   4   6   8  10  12  14  16  18
  3   6   9  12  15  18  21  24  27
  4   8  12  16  20  24  28  32  36
  5  10  15  20  25  30  35  40  45
  6  12  18  24  30  36  42  48  54
  7  14  21  28  35  42  49  56  63
  8  16  24  32  40  48  56  64  72
  9  18  27  36  45  54  63  72  81
```

- **Методы класса Arrays**
  - **Методы сортировки:**

```
static void sort(type[] a)   // type может быть byte, short, int, long,
                             // char, float, double или тип Object
static void sort(type[] a, int from, int to)
static void sort(Object[] a, Comparator c)
static void sort(Object[] a, int from, int to, Comparator c)
```

  - **Методы бинарного поиска:**

```
static int binarySearch(type[] a, type element)
static int binarySearch(Object[] a, Object element,
Comparator c).
```

  - **Методы заполнения массива:**

```
static void fill(type[], type value)
static void fill(type[], int from, int to, type value)
```

  - **Методы сравнения массивов:**

```
static boolean equals(type[] al, type[] a2)
```

Массивы

- В классе Arrays из пакета java.util собрано множество методов для работы с массивами. Их можно разделить на четыре группы.
- Восемнадцать статических методов сортируют массивы с разными типами числовых элементов в порядке возрастания чисел или просто объекты в их естественном порядке.
- Восемь из них имеют простой вид static void sort(type[] a), где type может быть один из семи примитивных типов byte, short, int, long, char, float, double или тип Object .
- Восемь методов с теми же типами сортируют часть массива от индекса from включительно до индекса to исключительно: static void sort(type[] a, int from, int to).
- Оставшиеся два метода сортировки упорядочивают массив или его часть с элементами типа Object по правилу, заданному объектом с, реализующим интерфейс Comparator.
- После сортировки можно организовать бинарный поиск в массиве одним из девяти статических методов поиска. Восемь методов имеют вид static int binarySearch(type[] a, type element), где type — один из тех же восьми типов. Девятый метод поиска имеет вид static int binarySearch(Object[] a, Object element, Comparator c). Он отыскивает элемент element в массиве, отсортированном в порядке, заданном объектом с.
- Методы поиска возвращают индекс найденного элемента массива. Если элемент не найден, то возвращается отрицательное число
- Восемнадцать статических методов заполняют массив или часть массива указанным значением value.
- Наконец, девять статических логических методов сравнивают массивы.
- Массивы считаются равными, и возвращается true , если они имеют одинаковую длину и равны элементы массивов с одинаковыми индексами.

Применение методов класса Arrays

```
import java.util.*;
class ArraysTest
{ public static void main(String[] args)
  { int[] a = {34, -45, 12, 67, -24, 45, 36, -56};
    Arrays.sort(a);
    for (int i = 0; i < a.length; i++)
      System.out.print(a[i] + " ");
    System.out.println();
    Arrays.fill(a, Arrays.binarySearch(a, 12), a.length,
  0);
    for (int i = 0; i < a.length; i++)
        System.out.print(a[i] + " ");
     System.out.println();
   }
}
```

```
-56 -45 -24 12 34 36 45 67
-56 -45 -24 0 0 0 0 0
```

Контрольные вопросы

1. **Одномерные массивы: объявление, определение, инициализация и примеры использования.**
2. **Двумерные массивы: объявление, определение, инициализация и примеры использования.**