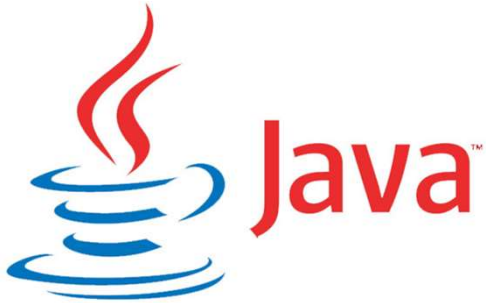
Тема 3.
Одномерные массивы



Массивы

Массив – группа элементов одного типа, имеющих одно имя и различающихся по номеру элемента внутри массива – индексу.

Массивы

Характеристики массивов

Массив в Java является ссылочным типом, имеет ряд атрибутов и методов, облегчающих работу с ними и предоставляющих дополнительные возможности.

1. Мерность
   - *Одномерные массивы (векторы)*
   - *Двухмерные массивы (матрицы)*
   - *Многомерные массивы*
2. Размер (количество элементов) каждого измерения

Массивы

Принципы работы

- Объявление переменной-ссылки на массив (декларация)
- Создание объекта-массива (выделение памяти)
- Инициализация (присвоение начальных значений)
- Обработка (обращение к элементам)

Массивы

Если переменные предназначены для хранения одиночного значения, то массив представляет собой набор однотипных значений.

Объявление массива похоже на объявление переменной.

Объявить массив можно двумя способами:

- int[] nums;
- String str[];

Массивы

Объявление

- Синтаксис объявления массива:

```
Тип[] имя;
```
или
```
Тип имя[];
```

```
int[] nums;
```
или
```
int nums[];
```

- Если переменная объявлена, но еще не проинициализирована, выделение памяти под массив не производится и размер массива указать нельзя.
- Оба объявления вернут одинаковый результат. Разницы нет.

Массивы

Создание (выделение) памяти

- Выделение памяти под массив происходит с помощью ключевого слова **new**:

```
int[] iData = new int[10];
```

```
int[] iData = new int[]{1, 2, 3, 4, 12, 9};
```

- В данном случае будет создан массив из 10 целых чисел.

Массивы

Значения элементов по умолчанию

В отличие от локальных переменных, элементы массивов примитивных типов инициализируются значениями по умолчанию:

- Числовые элементы – нулями;
- Символьные – значением ‘\o’;
- Логические - значением false;
- Массивы объектов – значением null.

Массивы

Обработка

- Массивы обрабатываются не целиком, а поэлементно;
- Доступ к элементу массива осуществляется по его индексу (порядковому номеру);
- Как правило, доступ к элементам массива осуществляется в цикле;
- Нумерация элементов массива **ВСЕГДА** начинается с **НУЛЯ**;
- Следовательно, конечный элемент массива из N элементов имеет номер (индекс) **N - 1**.

Массивы

Инициализация

- При создании переменной-ссылки на массив можно явно произвести его инициализацию, что приведет к созданию массива, выделению необходимого объема памяти и размещению в ней заданных значений;
- В примере ниже массив array будет состоять из 10-ти элементов и занимать в памяти 40 байт

```
Int[] array = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
```

Массивы

Инициализация

- В Java любая инициализация переменных происходит на этапе выполнения, а не компиляции. Поэтому для инициализации можно использовать не только литеральные константы, но и переменные или значения выражений.
- **Важно!** Локальные переменные, в отличии от элементов массива, не инициализируются по умолчанию. Во избежание ошибок при компиляции они должны быть инициализированы явно.

```
int a; int b;
a = a + b; // ошибка: должны быть проинициализированы
```

Массивы

Типовые алгоритмы обработки

- Присвоение начальных значений или генерация значений элементов случайным образом;
- Поиск элемента массива и его индекса: максимальный, минимальный, заданный;
- Обработка значений: вычисление суммы, разности, произведения, среднего арифметического и т.п.;
- Сортировка элементов массива (упорядочение по возрастанию или убыванию);
- Перестановка элементов массива.

Массивы

Добавление элемента в массив

Как мы уже видели, массивы имеют **фиксированный размер**.

Итак, чтобы добавить элемент, сначала нам нужно объявить новый массив, который больше старого массива, и скопировать элементы из старого массива во вновь созданный массив. После этого мы можем добавить новый элемент в этот вновь созданный массив.

Массивы

Пример обработки - подсчет среднего арифметического температуры:

```
int temper[] = {25,28,31,26,33,30,32,24,30,32};

double avg;
int sum = 0;
int n = temper.length;
for (int i = 0; i < n; i++) {
    sum += temper[i];
}

avg = (double)sum / n;
```

Массивы

Проход по всем элементам

В Java есть специальная форма цикла for, которая упрощает полный перебор всех элементов массива или коллекции:

```
for(Type Var_Name : Array_name){
        loop_body;
}
```

Например:

```
for(int a : array){
    sum += a;
}
```

Массивы

Проход по всем элементам

- В некоторых других языках (Perl, PHP и прочие) подобный цикл записывается как for each («для каждого элемента»);
- Отсутствие счетчика делает применение этого вида цикла ограниченным.

Массивы

Свойство length

Для прохода по всем элементам массива можно использовать цикл for со счетчиком, используя в качестве верхней границы свойство объекта-массива **length**:

```
for(int i = 0; i < array.length; i++){
        sum += array[i];
}
```

Использование свойства length делает программу более универсальной и не зависящей от конкретного значения размера массива, поэтому его использование предпочтительно.

Практика

Задача: задан массив

int[] mas = {25, 47, 34, 18, 96, 33, 28, 55, 87, 13}

Найти максимальное и минимальное значения и вывести на экран.

Массивы

Выход за границы массива

- Во время выполнения приложения виртуальная машина Java отслеживает выход за границы массива;
- Если приложение пытается выйти за пределы массива, генерируется исключение:

java.lang.ArrayIndexOutOfBoundsException

Массивы

Копирование массивов

- Если присвоить одной переменной-ссылке на массив другую переменную-ссылку на массив, то будет скопирован только адрес массива:

```
int[] a = new int[3];
int[] b = a;
```

- Если изменить массив b, то это скажется и на массиве a, так как эти переменные-ссылки указывают на один и тот же массив;
- Скопировать значения элементов массива можно в цикле;

Массивы

Копирование массивов

- Имеется также системный метод копирования массивов:

  System.arraycopy(a, indexA, b, indexB, count);

- Из a в b копируется count элементов, начиная с индекса indexA массива a. Они размещаются в массиве b, начиная с индекска indexB.

Массивы

Методы обработки массивов

- Используется класс Arrays из пакета java.util (т.е. данный пакет необходимо импортировать: ***import java.util.*;***)
- Arrays.fill(array, values) – заполняет массив одинаковыми значениями values;
- Arrays.equals(a, b) – сравнивает два массива по элементам.

Сравнивать a == b нельзя, так как будут сравниваться адреса массивов, а не значения!

- Arrays.sort(a) – сортирует массив и др.

Класс Scanner

Для ввода данных используется класс Scanner из библиотеки пакетов Java.

В классе есть методы для чтения очередного символа заданного типа со стандартного потока ввода, а также для проверки существования такого символа:

- nextInt()
- nextLine()
- …
- hasNextInt()
- …

Класс Scanner

Пример проверки вводимого с клавиатуры числа:

```
public class A {

	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		System.out.println("Введите целое число");
		if(sc.hasNextInt()){
			int a=sc.nextInt();
			System.out.println("Вы ввели "+a);
		}else{
			System.out.println("Вы ввели не целое число");
		}

	}

}
```

Класс Scanner

Пример считывания целого числа из потока ввода:

```
public class A {

	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		System.out.println("Введите число");
		int a = sc.nextInt();
		System.out.println("Вы ввели "+a);

	}

}
```

Практика

Задача: организовать ввод массива из потока ввода (клавиатуры) с использованием класса Scanner. Размер массива и его элементы ввести с клавиатуры с использованием класса Scanner.

Класс Random

Класс Random представляет собой генератор псевдослучайных чисел.

В классе есть методы для генерации очередного числа определенного типа:

- nextInt()
- nextDouble()
- …
- nextInt(int i)

Практика

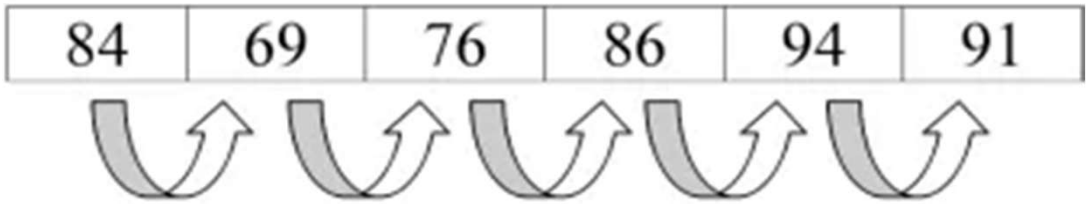
Задача: организовать генерацию массива с использованием класса Random. Размер массива ввести с клавиатуры с использованием класса Scanner.

Пузырьковая сортировка

Или **сортировка простыми обменами**. Бессмертная классика жанра.

Принцип действий прост: обходим массив от начала до конца, попутно меняя местами неотсортированные соседние элементы. В результате первого прохода на последнее место «всплывёт» максимальный элемент.

Теперь снова обходим не отсортированную часть массива. Второй по величине элемент окажется на предпоследнем месте.



Пузырьковая сортировка

```
for(int i=0; i < k.length; i++)
  for(int j = 0; j < k.length - 1 - i; j++)
    if (k[j] > k[j+1]){

      int tmp  = k[j];
      k[j] = k[j + 1];
      k[j + 1] = tmp;
    }
```

Шейкерная сортировка

А что будет если в пузырьковой сортировке не только в конец задвигать максимумы, а ещё и в начало перебрасывать минимумы ?

Получится **шейкерная сортировка**.

Она же **сортировка перемешиванием**, она же **коктейльная сортировка**. Начинается процесс как в «пузырьке»: выдавливаем максимум на самые задворки. После этого разворачиваемся на 1800 и идём в обратную сторону, при этом уже перекатывая в начало не максимум, а минимум.

Шейкерная сортировка

```
int temp;
int leftSide = 0;
int rightSide = array.length - 1;

do {
    for (int i = leftSide; i < rightSide; i++) { // Сортир. влево ищем МАКС. знач.
        if (array[i] > array[i + 1]) {
            temp = array[i];
            array[i] = array[i + 1];
            array[i + 1] = temp;
        }
    }
    rightSide--; // Уменьшаем количество проходов
    for (int i = rightSide; i > leftSide; i--) { // Теперь идём в обратную сторону ищем МИН.
        if (array[i] < array[i - 1]) {
            temp = array[i];
            array[i] = array[i - 1];
            array[i - 1] = temp;
        }
    }
    leftSide++; // Уменьшаем количество проходов т.к. мы шли с конца исходя из условия
    // последнего for, что бы уменьшить кол-во проходок6 нужно инкрементировать счётчик
} while (leftSide < rightSide); // Априори наше условие при котором буду происходить проходки
```