Теория и технология программирования

Программирование на языке Java

Лекция 2. Основные типы и конструкции, ввод-вывод

Глухих Михаил Игоревич, к.т.н., доц.
mailto: glukhikh@mail.ru

Основные конструкции

- Ветвления – аналогично C++
  - `if (...) { ... } else { ... }`
  - `switch (...) {`

```
    case 1:
        ...
        break;
    default:
        ...
        break;
}
```

Допустимые типы ключа в switch

- Целые числа
- Символы
- Элементы перечисления (enum)
- Строки (**только в JDK 7**)

Основные конструкции

- Циклы – аналогично C++
  - `while (...) { ... }`
  - `do { ... } while (...);`
  - `for (int i=0; i<10; i++) { ... }`
- continue – перейти к выполнению следующей итерации цикла
- break – завершить цикл
- goto в Java нет! правда, есть continue с меткой и break с меткой (позже будет пример)

Строки в Java

- Для строк используется тип String:
  - `String s = "Hello, world!";`
- Тип String – ссылочный (reference type), при присваивании строк на самом деле происходит присваивание ссылок:
  - `String s1 = s;`
- Как и любой ссылочный тип, строка может хранить пустую ссылку
  - `String s2 = null;`
- Тип String – неизменяемый (**immutable type**). Это значит, что объекты-строки в памяти не могут меняться. В следующей команде создается новый объект-строка и ссылка перенаправляется на него:
  - `s = "I am glad to see you";`
- См. пример

Операции со строками

- Строки можно складывать друг с другом
  - `String s2 = s + s1;`
- А также с объектами других типов
  - `String s3 = s + 42;`
- Кроме этого, строка – это класс
- У которого есть довольно большое количество методов

Основные методы работы со строками

- s.charAt(3) – определить символ на 3-й позиции
- s.length() – определить длину строки
- s.equals(s1) – сравнение строк на равенство
- s==s1 – сравнение ссылок на равенство
- s.substring(2, 5) – создать подстроку из символов 2, 3, 4 данной строки
- s.indexOf('a') – определить номер первого символа 'a' в строке (-1 если таких нет)
- и так далее (см. JavaDoc)
- Методы не работают, если s – пустая ссылка (возникает NullPointerException)

Строитель строк

- Используется, когда требуется создать строку из многих других строк, например:
  - `String s = s1 + 23 + s2 + 2.5 + s3;`
  - так можно, но неэффективно (будет создано 3 промежуточных строки)!
- Лучше:
  - `StringBuilder sb = new StringBuilder();`
  - `sb.append(s1);`
  - `sb.append(23);`
  - `sb.append(s2).append(2.5).append(s3);`
  - `String s = sb.toString();`

Массивы в Java

- Тип массива в Java обозначается как T[], где T – базовый тип
  - например, int[], float[], double[], String[] – размерность при этом не указывается
- Создается массив следующим образом:
  - `int[] arr = new int[10]; // инициализация нулями`
  - `int[] arr = null; // нулевая ссылка`
  - `// начальные значения заданы`
  - `int[] arr2 = new int[] { 2, 3, 4 };`
- Обращение к элементам происходит как и в C++
  - `arr[0] = arr[1];`
- Массив – ссылочный тип, при присваивании массивов происходит копирование ссылок:
  - `int[] arr3 = arr;`
- См. пример

Границы массивов в Java

- Для любого массива можно определить его размер, обратившись к свойству length
  - `int arrLength = arr.length;`
- Допустимые значения индекса при обращении к массиву – от 0 до arr.length-1
- При обращении по недопустимому индексу возникает ArrayIndexOutOfBoundsException
- См. пример

Перебор элементов массива, цикл for-each

- Старый вариант (цикл for)

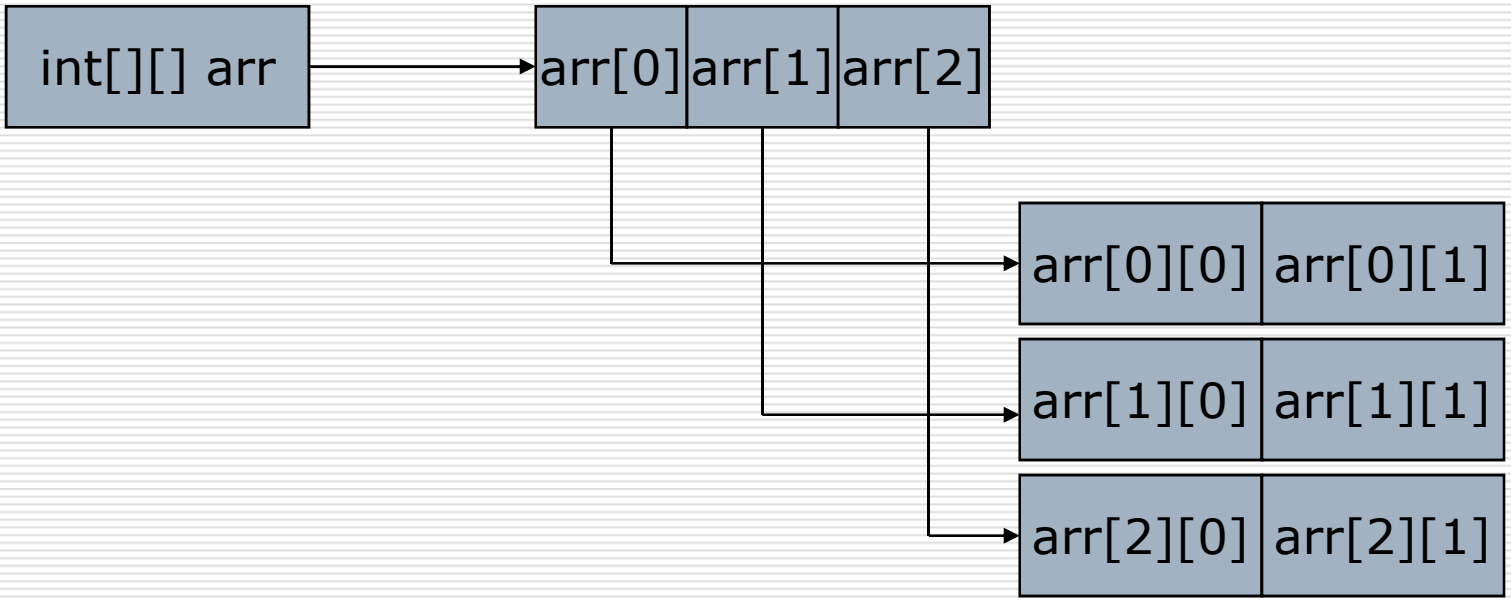
```
// Обратите внимание, что функция статическая
// Обратите внимание, как массив передается в функцию
static int getArraySum(int[] arr) {
    int sum=0;
    for (int i=0; i<arr.length; i++)
        sum += arr[i];
    return sum;
}
```

- Новый вариант (цикл for-each)

```
static int getArraySum(int[] arr) {
    int sum=0;
    for (int elem: arr)
        sum += elem;
    return sum;
}
```

Многомерные массивы

- В Java многомерные массивы являются ступенчатыми (фактически массив массивов)



Многомерные массивы

- Например
  - `int[][] arr2d = new int[3][];`
  - `for (int i=0; i<3; i++)`
  - `    arr2d[i] = new int[2];`
- Или просто
  - `arr2d = new int[3][2];`

Пример – поиск -1 в двумерном массиве, break с меткой

```
static void findMinusOne(int[][] arr2d) {
L1:
   for (int i=0; i<arr2d.length; i++) {
      for (int j=0; j<arr2d[i].length; j++) {
         if (arr2d[i][j]==-1) {
            System.out.println("i="+i+" j="+j);
            break L1; // break с меткой
         }
      }
   }
}
```

Операции над массивами

- Определены внутри класса java.util.Arrays (данный класс содержит только функции, причем все они статические)
  - `import java.util.Arrays;`
- Arrays.equals(arr1, arr2) – сравнение на поэлементное равенство
  - arr1==arr2 – сравнение ссылок на равенство
- Arrays.binarySearch(arr, elem) – бинарный поиск элемента
- Arrays.copyOf(arr, length) – создать копию массива заданной длины
- Arrays.fill(arr, elem) – заполнение элементом
- Arrays.sort(arr) – сортировка
- И так далее! См. JavaDoc

Числовые классы

- Классы-обкладки
  - Integer, Long, Float, Double, Boolean, ...
- Многоразрядные варианты
  - Находятся в пакете java.math
  - BigInteger, BigDecimal

Ввод-вывод в Java

- Классы ввода-вывода в Java находятся в пакете java.io
  - `import java.io.*;`
  - `// или, если нужен конкретный класс`
  - `import java.io.PrintStream;`
- Что такое System.in и System.out?
  - это объекты классов InputStream и PrintStream
  - как и все классы – ссылочный тип

Создание нового потока вывода

```
// файловый поток
PrintStream fout = new
    PrintStream("out.txt");
// или с указанием кодировки
PrintStream fout866 = new
    PrintStream("out.txt", "CP866");
// или консольный с
// указанием кодировки
PrintStream out866 = new PrintStream(
    System.out, true, "CP866");
// См. пример
```

Основные возможности PrintStream

- print(var) – вывод var в поток (в случае System.out на консоль)
- println(var) – вывод var в поток с переводом строки
- printf(fmt, ...) или format(fmt, ...) – форматированный вывод, аналог функции printf в языке C
- close() – закрытие потока

Создание нового потока ввода

- InputStreamReader reader = new InputStreamReader(System.in, "CP866");
- // или
- FileInputStream fstream = new FileInputStream("in.txt");
- InputStreamReader freader = new InputStreamReader(fstream, "CP1251");

Основные возможности InputStream/InputStreamReader

```
char[] buf = new char[50];
int symNum = in.read(buf);
// или
int symNum = in.read(buf, 0, 50);
// и все!
// А как же чтение чисел,
// например?
```

Разбор введенных данных

```
// Преобразование в строку
String str = new String(buf);
// Длина строки будет совпадать с
// длиной массива
// Урезание лишних символов
str = str.substring(0, symNum);
// Удаление пробелов в начале
// и в конце
str = str.trim();
```

Разбор получившейся строки

```
// Разбить строку по пробелам
// 12 34 56 => { "12", "34", "56" }
String[] splitted;
try {
   // Аргумент split – регулярное выражение
   splitted = str.split(" ");
} catch (PatternSyntaxException e) {
   // Если разбиение не получилось
   out866.println("Неверный формат ввода!");
   return;
}
```

Разбор подстрок

```
// Преобразование к целым
for (String substr: splitted) {
    int num = 0;
    try {
        num = Integer.parseInt(substr);
        out866.println("Введено число: " + num);
    } catch (NumberFormatException e) {
        out866.println("Введено НЕ число: " +
    substr);
    }
    // ...
}
```

Другой вариант разбора строки - StringTokenizer

```
StringTokenizer tokenizer = new
    StringTokenizer(str, " \t");
// В строке указываются разделители
while (tokenizer.hasMoreElements()) {
     String substr =
         tokenizer.nextToken();
     // ...
}
```

Третий вариант разбора строки – Scanner

```
Scanner sc = new Scanner(str);
// Можно вместо строки указать поток
while (sc.hasNextInt()) {
    int num = sc.nextInt();
    // ...
}
```

Итоги

- Рассмотрены основные конструкции
- Рассмотрены массивы, строки и числовые типы
- Рассмотрены основные способы ввода и вывода