Лекция 2

Базовый ввод и вывод в Java

Операции языка Java

Операторы управления Java

Массивы в Java

Структура файла содержащего код Java

1. Описание пакета

package имя_пакета

2. Импорт классов из других пакетов

import имя_пакета.*;
import имя_пакета.Имя_класса;

3. Описание классов

Ввод и вывод в Java

Вывод данных осуществляется через объект System.out класса PrintStream.

Основными методами вывода в этом классе являются:

- print(); //Без перехода на новую строку
- println(); //С переходом на новую строку
- printf(); //Форматный вывод

Ввод и вывод в Java

Ввод данных через консоль в языке Java осуществляется посредством объекта System.in класса InputStream.

Но данный класс содержит фактически единственный метод read. Использование этого метода для ввода данных неудобно, поэтому для этих целей используется класс Scanner из пакета java.util.

Этот класс содержит набор методов для чтения и проверки данных определенного типа из входного потока:

- next() – чтение строки,
- nextLine() – чтение строки до перехода на новую строку,
- nextByte() – чтение числа типа byte,
- nextShort() – чтение числа типа short,
- nextInt() – чтение числа типа int,
- nextLong() – чтение числа типа long,
- nextBoolean() – чтение значения логического типа,
- nextFloat() – чтение числа типа float,
- nextDouble() – чтение числа типа double.

Изменить «локаль» можно методом useLocale(Locale locale)

Операции

Операция присваивания в языке Java имеет следующий формат записи:

переменная = выражение;

Математические операции

В языке Java определены следующие математические операции, применимые над всеми численными типами:

+	Сложение
-	Вычитание
*	Умножение
/	Деление
%	Взятие остатка от деления
++	Инкремент (префиксный и постфиксный)
--	Декремент (префиксный и постфиксный)

Целочисленные операции

В языке Java определены следующие операции над целочисленными значениями (byte, short, char, int):

~	побитовая унарная операция отрицания NOT
&	побитовое AND
|	побитовое OR
^	побитовое XOR
>>	сдвиг вправо
>>>	сдвиг вправо с заполнением нулями
<<	сдвиг влево

Совмещенные операции

В языке Java как и в языке C++ доступны совмещенные операции:

+= -= /= *= %=

&= |= ^= >>= >>>= <<=

Операции сравнения

>	Больше
<	Меньше
>=	Больше или равно
<=	Меньше или равно
==	Равно
!=	Не равно

Булевские логические операции

&	Логическое И
|	Логическое ИЛИ
^	Логическое исключающее ИЛИ
||	Замыкающее ИЛИ
&&	Замыкающее И
!	Логическое унарное НЕ
&=	И с присваиванием
|=	ИЛИ с присваиванием
^=	Исключающее ИЛИ с присваиванием
==	Равно
!=	Не равно
?:	Тернарная операция (условная операция)

Приоритеты операций Java

( )	[ ]	.	
++	--	~	!
*	/	%	
+	-		
>>	>>>	<<	
>	>=	<	<=
==	!=		
&			
^			
|			
&&			
||			
?:			
=	op=		

Операторы управления

Все операторы управления в Java можно разделить на три группы:

- операторы выбора;
- операторы циклов;
- операторы переходов.

Операторы выбора

Оператор условия:
if(логическое_выражение) блок_операторов_1
[**else** блок_операторов_2]

Примеры:

```
if(x>10) x--;
  else x = 0;

if((x>0)&&(x<10)){
  y = x;
  x += 2;
}
```

Операторы выбора

Оператор выбора:

```
switch(выражение){
  case значение1:
    …
    break;
 case значение2:
    …
    break;
  …
  default:
    …
}
```

Выражение должно иметь тип byte, char, short, int.

В java 1.7: String

Тип каждого значения должен быть совместим с типом выражения.

Пример оператора выбора

```
int i = 3;
String str;
switch(i){
  case 1: str = “один”;      break;
  case 2:  str = “два”;      break;
  case 3:  str = “три”;      break;
  case 4:  str = “четыре”;  break;
  case 5:  str = “пять”;      break;
 default: str = “неизвестно”;
}
System.out.println(str);
```

Пример оператора выбора

```
int month = 3;
String season;
switch(month){
  case 12: case 1: case 2:
    season = “зима”;
    break;
 case 3: case 4: case 5:
    season = “весна”;
    break;
 case 6: case 7: case 8:
    season = “лето”;
    break;
 case 9: case 10: case 11:
    season = “осень”;
    break;
  default: season = “неизвестно”;
}
```

Операторы циклов

С предусловием:

```
while(условие){
  //тело цикла
}
```

С постусловием:

```
do{
  //Тело цикла
}while(условие);
```

Операторы циклов

```
for(инициализация;условие;приращение){
  //Тело цикла
}
```

Пример:

```
for(int i=0;i<10;i++){
  System.out.println(i+"^2 = " + (i*i));
}
```

Операторы циклов

Пример цикла for:

```
for(int i=0, b=10; i<b; i++, b--){
  System.out.println(“a = ” + a);
  System.out.println(“b = ” + b);
}
```

Операторы циклов

Оператор **for** в режиме **for-each** (доступен начиная с JDK5):

```
for(тип итерацонной_переменной: коллекция){
  //тело цикла
}
```

Пример:

```
int arr[] = new int[10];
…
for(int a: arr){
  System.out.println(a);
}
```

Операторы переходов

В языке Java присутствуют три оператора переходов:

- break
- continue
- return

Оператор break

Применение:

- в операторе **switch**
- в операторах циклов
- переход по метке

Оператор break

```
for(int i = 0; i<100; i++) {
  //выход из цикла если i равно 10
  if(i == 10) break;
  System.out.println("i: "+ i);
}
System.out.println("Цикл завершен.");
```

Оператор break

```
for(int i = 0; i<3; i++) {
    System.out.print("проход" + i + ". ");
    for(int j = 0; j<100; j++) {
        if(j == 10) break; //выход из цикла,
                           //если j  равно 10
        System.out.print(j +" ");
    }
    System.out.println();
}
System.out.println("Циклы завершены.");
```

Оператор break

break метка;

```
outer: for(int i=0; i<3; i++) {
    System.out.print("Проход" + i + ": Н);
    for(int j = 0; j<100; j++) {
        //выход из обоих циклов
        if(j == 10) break outer;
        System.out.print(j + " ");
    }
    System.out.println("Не будет выведено!");
}
System.out.println("Циклы завершены.");
```

Оператор break

```
one: for(int i = 0; i<3; i++) {
     System.out.print("Pass " + i + ". ");
}

for(int j = 0; j<100; j++) {
    if(j == 10) break one; //ОШИБКА!
    System.out.print(j + " ");
}
```

Оператор continue

```
for(int i = 0; i<10; i++) {
    System.out.print(i + " ");
    if(i%2 == 0) continue;
    System.out.println("");
}
```

0 1
2 3
4 5
6 7
8 9

Оператор continue

```
outer: for (int i = 0; i<10; i++) {
    for (int j = 0; j<10; j++){
        if(j>i) {
            System.out.println();
            continue outer;
        }
        System.out.print(" " + (i * j));
    }
}
System.out.println();
```

Оператор return

Оператор return предназначен для явного возврата из метода.

return;
return значение;

```
public static void main(String args[]) {
    boolean t = true;
    System.out.println("До выполнения возврата.");
    if(t) return; // возврат к вызывающему объекту
    System.out.println(
      "Этот оператор выполняться не будет.");
}
```

Массивы в Java

Массивы в Java являются классами, которые задаются в неявной форме.

Описание одномерных массивов в Java:

тип [] имя [=инициализация];

тип имя[] [=инициализация];

Примеры:

```
int[] arr1 = {1,2,3,4,5}, arr2 = {6,7,8,9,0};
int mas[] = {1,2,3,4,5,6,7,8,9,0}, val = 100;
```

Массивы в Java

Если массив объявлен без инициализации, то его необходимо создать используя оператор **new**:

```
int [] arr;
arr = new int[10];
for(int i=0;i<arr.length;i++) arr[i] = i;
```

Оператор циклов для коллекций

Оператор **for** в режиме **for-each** (доступен начиная с JDK5):

```
for(тип итерационной_переменной: коллекция){
  //тело цикла
}
```

Пример:

```
int arr[] = new int[10];
…
for(int a: arr){
  System.out.println(a);
}
```

Оператор циклов для коллекций

При получении данных из коллекции (элемента из массива) создается его копия, поэтому следующий фрагмент программы никакого влияния на массив оказывать не будет:

```
for(int a: arr){
  a = 100;
}
```

Двумерные массивы

Объявление прямоугольных матриц с инициализацией:

int [][] arr = {{1,2,3,4,5},{6,7,8,9,0}};

или

int arr [][] = {{1,2,3,4,5},{6,7,8,9,0}};

Вывод массива:

```
for(int i=0;i<2;i++){
  for(int j=0;j<5;j++)
    System.out.print(arr[i][j] + " ");
  System.out.println();
}
```

Двумерные массивы

Создание и использование прямоугольной матрицы:

```
int arr[][];

arr = new int[2][5];
for(int i=0;i<2;i++){
  for(int j=0;j<5;j++) arr[i][j] = i+j;
}
for(int i=0;i<2;i++){
  for(int j=0;j<5;j++)
    System.out.print(arr[i][j]+" ");
  System.out.println();
}
```

Двумерные массивы

Объявление непрямоугольных матриц с инициализацией:

```
int [][] arr =
          {{1,2,3,4,5,6,7},{8,9,0}};

for(int i=0;i<arr.length;i++){
    for(int j=0;j<arr[i].length;j++)
        System.out.print(arr[i][j]+" ");
    System.out.println();
}
```

Двумерные матрицы

Создание и использование непрямоугольной матрицы:

```
int [][] arr ;

arr = new int[3][];
for(int i=0;i<3;i++){
  arr[i] = new int[i+2];
  for(int j=0;j<arr[i].length;j++)
    arr[i][j] = i+j;
}
for(int i=0;i<arr.length;i++){
  for(int j=0;j<arr[i].length;j++)
    System.out.print(arr[i][j]+" ");
  System.out.println();
}
```

Двумерные массивы

Использование циклов для коллекций:

```
int [][] arr ;
…
for(int[] mas: arr){
    for(int a: mas)
      System.out.print(a+" ");
    System.out.println();
}
```