



Основы программирования на языке Java

Лекция 2

Цели

- Знакомство с синтаксисом языка Java
- Определение типов данных
- Знакомство с
 - ❖ операторами,
 - ❖ управляющими структурами



Синтаксис языка Java

Идентификаторы - это имена переменных, процедур, функций и т.д.

- Длина идентификатора в Java любая.
- Идентификаторы в Java состоят из так называемых букв Java (Java letters) и арабских цифр 0—9, причем первым символом идентификатора не может быть цифра.
- В число букв Java обязательно входят прописные и строчные латинские буквы, знак подчеркивания `_`, а так же символы национальных алфавитов.
- Служебные слова Java, такие как **class**, **void**, **static**, зарезервированы, их нельзя использовать в качестве идентификаторов своих объектов.
- Не указывайте в именах знак доллара (`$`). Компилятор Java использует его для записи имен вложенных классов.
- В именах лучше не использовать строчную букву `l`, которую легко спутать с единицей, и букву `o`, которую легко принять за ноль.
- Не забывайте о рекомендациях "Code Conventions".

Идентификаторы

В классе `Character`, входящем в состав Java API, есть два метода, проверяющие, пригоден ли данный символ для использования в идентификаторе:

- ❖ **`public static boolean isJavaIdentifierStart (char ch)`** возвращает `true`, если данный символ может быть использован в качестве первого символа идентификатора, `false` – в противном случае.
- ❖ **`public static boolean isJavaIdentifierPart (char ch)`** возвращает `true`, если данный символ может быть использован в качестве символа в идентификаторе (за исключением первого), `false` – в противном случае.

Ключевые слова

- **Зарезервированные ключевые слова** - это специальные идентификаторы, которые в языке Java используются для того, чтобы определять встроенные типы, модификаторы и средства управления выполнением программы.
- Эти ключевые слова совместно с синтаксисом операторов и разделителей входят в описание языка Java.
- Они могут применяться только по назначению, их нельзя использовать в качестве идентификаторов для имен переменных, классов или методов.

Ключевые слова

- В языке Java имеется 59 зарезервированных слов:

abstract	const	loat	int	protected	throw
boolean	continue	for	interface	public	throws
break	default	future	long	rest	transient
byte	do	generic	native	return	true
byvalue	double	goto	new	short	try
case	else	if	null	static	var
cast	extends	implements	operator	super	void
catch	false	import	outer	switch	volatile
char	final	inner	package	synchronized	while
class	finally	instanceof	private	this	

Ключевые слова

Кроме этого, в Java есть **зарезервированные имена методов** (эти методы наследуются каждым классом, их нельзя использовать, за исключением случаев явного переопределения методов класса Objects):

clone	finalize	hashCode	notifyAll	wait
equals	getClass	notify	toString	

Комментарии

- Комментарии очень удобны для чтения и понимания кода, они превращают программу в документ, описывающий ее действия.
- Программу с хорошими комментариями называют **самодокументированной**.

Комментарии вводятся следующим образом:

1. за двумя наклонными чертами подряд `//`, без пробела между ними, начинается комментарий, продолжающийся до конца строки;

```
// System.out.println("My first Java programm!!!!");
```

2. за наклонной чертой и звездочкой `/*` начинается комментарий, который может занимать несколько строк, до звездочки и наклонной черты `*/` (без пробелов между этими знаками);

```
/* public static void main(String[] args) {  
    System.out.println("My first Java programm!!!!");  
}  
*/
```


Комментарии

Комментарии вводятся следующим образом:

3. за наклонной чертой и двумя звездочками подряд, без пробелов, **/**** начинается комментарий, который может занимать несколько строк до звездочки (одной) и наклонной черты ***/** и обрабатываться программой **javadoc**, извлекающая эти комментарии в отдельные файлы формата HTML и создающая гиперссылки между ними.
- В такой комментарий можно вставить указания программе **javadoc**, которые начинаются с символа **@**.

```
/**
 * <p> Title: Вычисление площади фигуры</p> (название)
 * <p> Description: Найти площадь правильного двенадцатиугольника, если его
   сторона равна a.</p> (описание)
 * <p> Copyright: группа ПМ-33 (с) 2008</p> (авторские права)
 * <p> Company: МарГУ </p> (компания, организация или образовательное учреждение)
 * @author Петров В.И. (имя, фамилия автора)
 * @version 1.0 (версия программы)
 */
```

Комментарии



СИМВОЛЫ

- Печатные символы можно записать в апострофах: 'a', 'N', '?'.
- Управляющие символы записываются в апострофах с обратной наклонной чертой:
 - ❖ '\n' — символ перевода строки newline с кодом ASCII 10;
 - ❖ '\r' — символ возврата каретки CR с кодом 13;
 - ❖ '\f' — символ перевода страницы FF с кодом 12;
 - ❖ '\b' — символ возврата на шаг BS с кодом 8;
 - ❖ '\t' — символ горизонтальной табуляции HT с кодом 9;
 - ❖ '\\' — обратная наклонная черта;
 - ❖ '\"' — кавычка;
 - ❖ '\'' — апостроф.

СИМВОЛЫ

- Код любого символа с десятичной кодировкой от 0 до 255 можно задать, записав его не более чем тремя цифрами в **восьмеричной системе** счисления в апострофах после обратной наклонной черты:

'\123' — буква S, **'\346'** — буква Ж в кодировке CP1251.

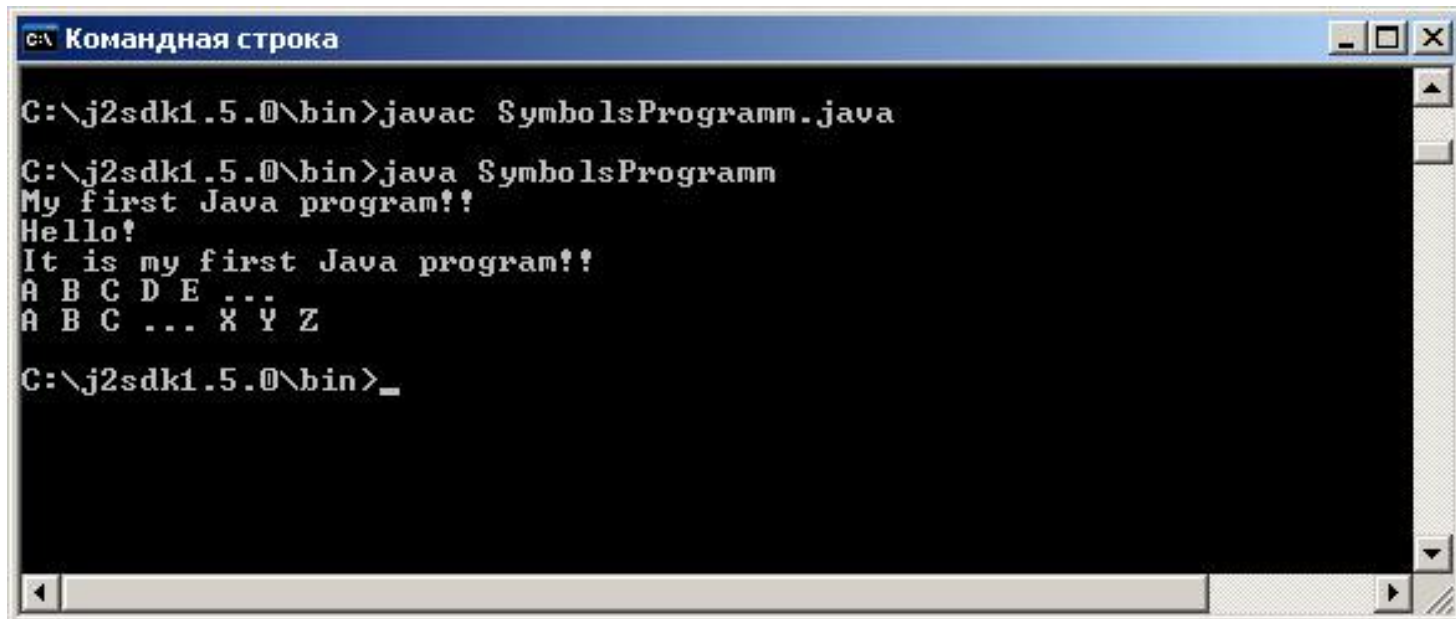
- Код любого символа в кодировке **Unicode** набирается в апострофах после обратной наклонной черты и латинской буквы **u** ровно четырьмя шестнадцатеричными цифрами:

'\u0053' — буква S, **'\u0416'** — буква Ж.

- Символы Unicode состоят из 16 бит, благодаря чему обеспечивается поддержка букв, входящих в большинство языков мира.

СИМВОЛЫ

```
class SymbolsProgramm {  
    public static void main(String[] args) {  
        System.out.println("My first Java program!!");  
        System.out.println("Hello! \n It is my first Java program!!");  
        System.out.println("\101 \102 \103 \104 \105 ...");  
        System.out.println("\u0041 \u0042 \u0043 ... \u0058 \u0059 \u005a ");  
    }  
}
```

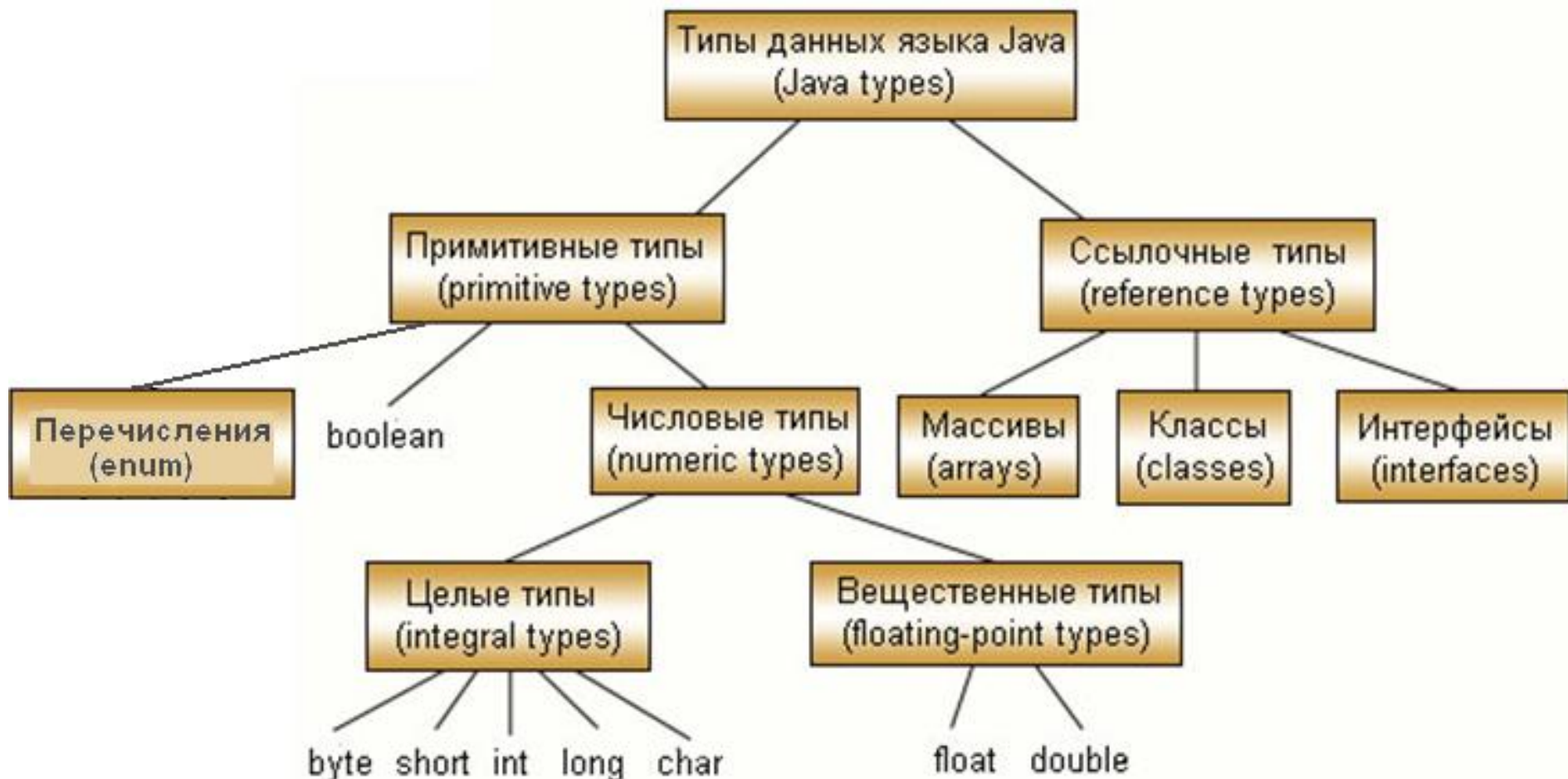


```
c:\ Командная строка  
  
C:\j2sdk1.5.0\bin>javac SymbolsProgramm.java  
  
C:\j2sdk1.5.0\bin>java SymbolsProgramm  
My first Java program!!  
Hello!  
It is my first Java program!!  
A B C D E ...  
A B C ... X Y Z  
  
C:\j2sdk1.5.0\bin>_
```

Основы языка Java

- Типы данных
- Переменные
- Операторы
- Управляющие структуры

Типы данных



Типы данных

- **Примитивными типами** называются такие, для которых данные содержатся в одной ячейке памяти, и эта ячейка не имеет подъячеек.
- **Ссылочными типами** называются такие, для которых в ячейке памяти (*ссылочной переменной*) содержатся не сами данные, а только адреса этих данных, то есть *ссылки* на них.
- При присваивании в ссылочную переменную заносится новый адрес, а не сами данные. Но непосредственного доступа к адресу, хранящемуся в ссылочных переменных, нет.
- Это сделано для обеспечения безопасности работы с данными — как с точки зрения устранения непреднамеренных ошибок.

Примитивные типы данных

Логический тип данных

`boolean a = true, b = false, c;`

■ Логические операции:

- ❖ отрицание (NOT) ! (обозначается восклицательным знаком);
- ❖ конъюнкция (AND) & (амперсанд);
- ❖ дизъюнкция (OR) | (вертикальная черта);
- ❖ исключающее ИЛИ (XOR) ^ (каре).

a	b	!a	a&b	a b	a^b
true	true	false	true	true	false
true	false	false	false	true	true
false	true	true	false	true	true
false	false	true	false	false	false

Примитивные типы данных

Целочисленный тип данных

byte a1 = 50, b1 = -99, b3;

short a2 = 0, b2 = 1;

int a3 = -100, b3 = 100, c3 = 9999, d3;

long a4 = 50, b4 = 2147483648L;

char ch1 = 'A', ch2 = '?', newLine = '\n';

Тип	Разрядность (байт)	Диапазон
byte	1	от -128 до 127
short	2	от -32768 до 32767
int	4	от -2147483648 до 2147483647
long	8	от -9223372036854775808 до 9223372036854775807
char	2	от '\u0000' до '\uFFFF' , в десятичной форме от 0 до 65535

Примитивные типы данных

Вещественный тип данных

`float a, x=2, y = 3.14F;` // обратите внимание на F, так как по умолчанию все литералы double

`double c, d= 16.2305, pi = 3.14159265358979323846;`

Тип	Разрядность (байт)	Диапазон	Точность
float	4	$3,4e-38 < x < 3,4e38$	7—8 цифр
double	8	$1,7e-308 < x < 1,7e308$	17 цифр

К обычным вещественным числам добавляются еще три значения:

1. **POSITIVE_INFINITY** (положительная бесконечность, выражаемая константой и возникающая при переполнении положительного значения)
2. **NEGATIVE_INFINITY** (отрицательная бесконечность)
3. **NaN** ("Не число", записываемое константой (Not a Number) и возникающее при делении вещественного числа на нуль или умножении нуля на бесконечность).

Перечисления (enum)

- **Перечисляемый тип** (англ. *enumeration*) — в тип данных, чьё множество значений представляет собой ограниченный список идентификаторов.
- Например, опишем с помощью **enum** тип данных для хранения времени года:

```
enum Season { WINTER, SPRING, SUMMER, AUTUMN } ;
```

- Его использование:

```
season = Season.SPRING;
```

```
if (season == Season.SPRING)
```

```
    season = Season.SUMMER;
```

```
System.out.println(season);
```

Ссылочные типы данных

Массивы

- **Массив** - это совокупность переменных одного типа, хранящихся в смежных ячейках оперативной памяти.
- Массивы в языке Java относятся к ссылочным типам и описываются своеобразно, но характерно для ссылочных типов.

Объявление массивов

- Существуют три способа объявления массивов:

- ❖ `datatype identifier [];`
- ❖ `datatype identifier [] = new datatype[size];`
- ❖ `datatype identifier [] = {value1,value2,...valueN};`

`float c[];`

`double b [] = new double [4];`

`int ar [] = {1, 2, 3, 4, 5};`

- Массивы в Java имеют некоторые свойства объектов. Так, для получения размерности массива можно обратиться к его свойству `length`:

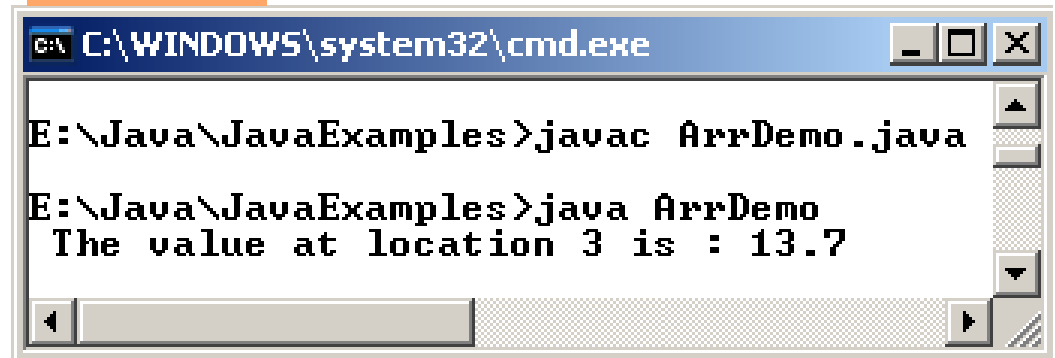
`int[] list = new int [10];`

`int n=list.length;`

Пример – Одномерный массив

```
class ArrDemo
{
    public static void main(String [] arg)
    {
        double nums[] = {10.1, 11.3, 12.5, 13.7, 14.9};
        System.out.println(" The value at location 3 is : " +nums[3]);
    }
}
```

Вывод



```
C:\WINDOWS\system32\cmd.exe

E:\Java\JavaExamples>javac ArrDemo.java

E:\Java\JavaExamples>java ArrDemo
The value at location 3 is : 13.7
```

Многомерный массив

- Элементами массивов в Java могут быть снова массивы.

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

что эквивалентно

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

```
int[][] d = new int[3] [4];
```

```
int[][] inds = {{1, 2, 3}, {4, 5, 6}};
```


Классы в языке Java

Класс - языковая конструкция, определяющая поля данных объектов данного класса (instance variables) и их поведение (methods).

- Синтаксис определения класса:

```
class Classname
{
    var_datatype variablename;
    :

    met_datatype methodname(parameter_list)
    :
}
```

Пример класса

Customer

CustomerName

Address

Email

Phone

Accept Customer Details

Display Customer Details



Интерфейсы

- **Интерфейс** - это набор абстрактных методов, которые не содержат никакого кода.
- Интерфейсы похожи на классы, но в отличие от последних у интерфейсов нет переменных представителей, а в объявлениях методов отсутствует реализация.
- Интерфейсы Java созданы для поддержки динамического выбора (resolution) методов во время выполнения программы.
- Класс может иметь любое количество интерфейсов.

Интерфейсы

Общая форма интерфейса:

```
interface имя {  
    тип_результата имя_метода1 (список параметров);  
    тип имя_переменной = значение;  
}
```

```
interface Callback {  
    void callback(int param);  
}
```

Преобразование типа

- При преобразовании типа (type casting) тип данных преобразовывается в необходимый другой тип данных.

- Пример

```
float c = 34.89675f;
```

```
int b = (int)c + 10;
```

```
int a = 100;
```

```
byte b = (byte) a;
```

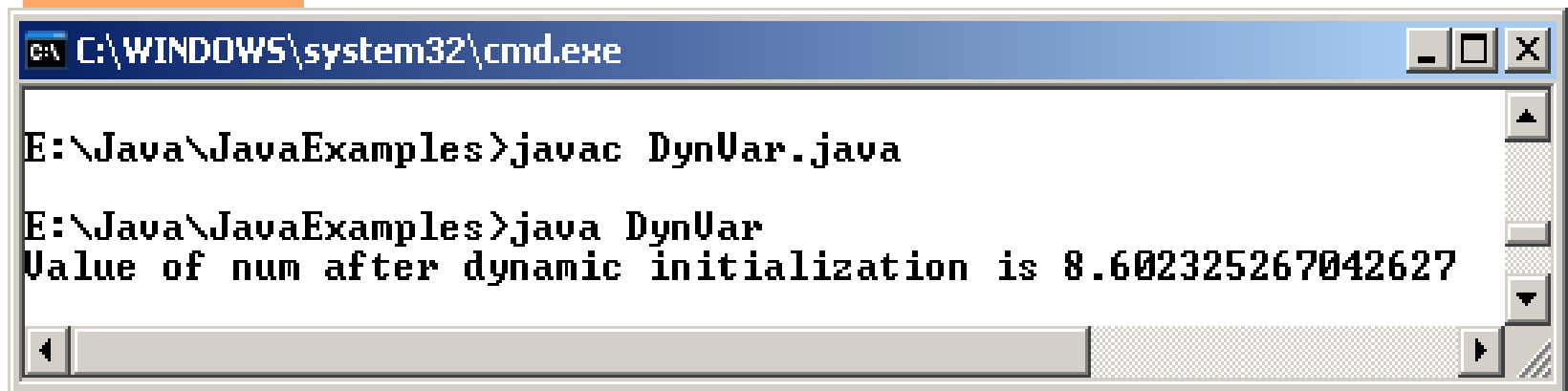
Переменные

- **Переменная** – это именованная ячейка памяти, содержимое которой может изменяться.
- Три компонента объявления переменной:
 - ❖ Тип данных
 - ❖ Имя
 - ❖ Присваиваемое начальное значение (необязательно)
- Синтаксис
`datatype identifier [=value][, identifier [=value]...];`

Пример

```
class DynVar
{
    public static void main(String [] args)
    {
        double len = 5.0, wide = 7.0;
        double num = Math.sqrt(len * len + wide * wide);
        System.out.println("Value of num after dynamic initialization is " + num);
    }
}
```

Вывод



The screenshot shows a Windows command prompt window with the title bar "C:\WINDOWS\system32\cmd.exe". The command prompt displays the following text:

```
E:\Java\JavaExamples>javac DynVar.java

E:\Java\JavaExamples>java DynVar
Value of num after dynamic initialization is 8.602325267042627
```

The output shows the successful compilation of `DynVar.java` and the execution of `DynVar`, which prints the value of `num` after dynamic initialization.

Операторы

- Арифметические операторы
- Битовые операторы
- Операторы отношений
- Логические операторы
- Условные операторы
- Операторы присваивания

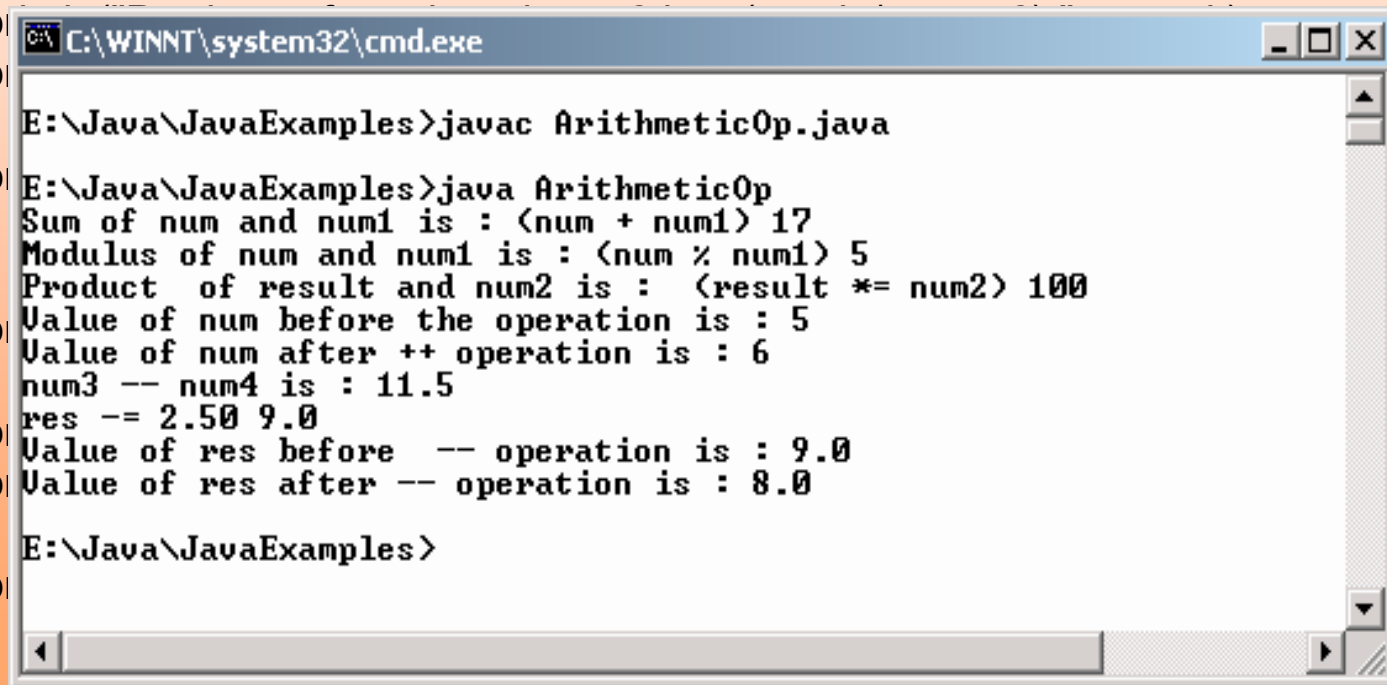
Арифметические операторы

К арифметическим операциям относятся:

- сложение **+** (плюс);
- вычитание **-** (дефис);
- умножение ***** (звездочка);
- деление **/** (наклонная черта — слэш);
- взятие остатка от деления (деление по модулю) **%** (процент);
- инкремент (увеличение на единицу) **++** ;
- декремент (уменьшение на единицу) **--**

Пример

```
class ArithmeticOp {  
    public static void main ( String [] arg){  
        int num = 5, num1 = 12, num2 = 20, result;  
        result = num + num1;  
        System.out.println("Sum of num and num1 is : (num + num1) " + result);  
        result = num % num1;  
        System.out.print ВЫВОД is of num and num1 is : (num % num1) " + result);  
        result *= num2;  
        System.out.p  
        System.out.p  
        num ++;  
        System.out.p  
        double num3  
        res = num3 -  
        System.out.p  
        res -= 2.50;  
        System.out.p  
        System.out.p  
        res--;  
        System.out.p
```



```
C:\WINNT\system32\cmd.exe  
E:\Java\JavaExamples>javac ArithmeticOp.java  
E:\Java\JavaExamples>java ArithmeticOp  
Sum of num and num1 is : (num + num1) 17  
Modulus of num and num1 is : (num % num1) 5  
Product of result and num2 is : (result *= num2) 100  
Value of num before the operation is : 5  
Value of num after ++ operation is : 6  
num3 -- num4 is : 11.5  
res -= 2.50 9.0  
Value of res before -- operation is : 9.0  
Value of res after -- operation is : 8.0  
E:\Java\JavaExamples>
```

Операторы отношений

- Операторы отношений проверяют отношение между двумя операндами.
- Результат выражения, в котором используются операторы отношений, является логическим (boolean) типом (то есть, либо true, либо false).
- Операторы отношений используются в управляющих структурах.

Операторы отношений

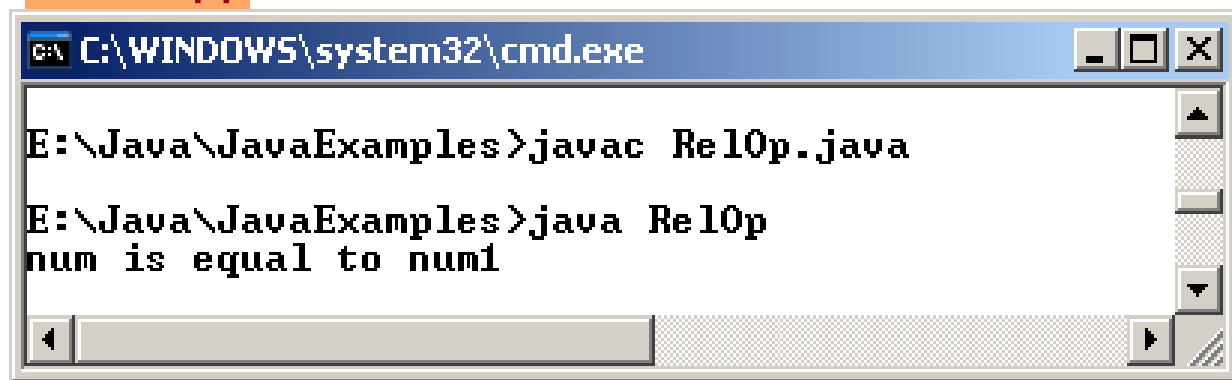
В языке Java шесть обычных операторов сравнения:

- больше > ;
- меньше < ;
- больше или равно >= ;
- меньше или равно <= ;
- равно == ;
- не равно != .

Пример

```
class RelOp
{
    public static void main(String [] args)
    {
        float num = 10.0F;
        double num1 = 10.0;
        if (num == num1)
            System.out.println ("num is equal to num1");
        else
            System.out.println ("num is not equal to num1");
    }
}
```

Вывод



The screenshot shows a Windows command prompt window with the title bar "C:\WINDOWS\system32\cmd.exe". The command prompt is open at the directory "E:\Java\JavaExamples". The user has entered the command "javac RelOp.java" to compile the program. The output of the compilation is not visible. The user has then entered the command "java RelOp" to run the program. The output of the program is "num is equal to num1".

```
C:\WINDOWS\system32\cmd.exe

E:\Java\JavaExamples>javac RelOp.java

E:\Java\JavaExamples>java RelOp
num is equal to num1
```

Битовые операторы

- Битовый оператор позволяет работать с отдельными битами целочисленного базового типа данных.
- Эти операторы воздействуют на отдельные биты своих операндов.
- Битовые операторы применяют Булеву (логическую) алгебру к соответствующим битам в двух операндах, чтобы получить результат.

Логические операторы

- Логические операторы работают с операндами типа **boolean**.
- Некоторые логические операторы:
 - ❖ **&** - логическое И
 - ❖ **|** - логическое ИЛИ
 - ❖ **^** - логическое исключающее ИЛИ
 - ❖ **!** – логическое отрицание
 - ❖ **||** - укороченное ИЛИ
 - ❖ **&&** - укороченное И

Логические операторы

- Логическое И(&) и Логическое ИЛИ (|) оценивают операнды по отдельности и затем считается полное выражение – True или False.
- В Короткой схеме И (&&), если левая сторона оператора ложна, правая не оценивается.
- В Короткой схеме ИЛИ (| |) если левая сторона истинна, то правая не оценивается.

Условные операторы

- Условный оператор является единственным в своём роде, поскольку это тернарный или тройной оператор, выражение которого содержит три операнда.
- Он может заменять некоторые типы конструкций if-then-else.
- **Пример:**

```
Category = (Age < 18) ? "Ребенок" : "Взрослый" ;
```

Операторы присваивания

- **Оператор присваивания** – это одиночный символ равенства =, который присваивает значение переменной.
- Можно одновременно выполнять присваивание значения нескольким переменным.
- Другими словами, этот оператор позволяет создать цепочку присваиваний.
- Кроме простого оператора присваивания есть еще 11 составных (compound assignment operators):

+=	-=	*=	/=	%=	&=	=	^=	<<=	>>=	>>>=
----	----	----	----	----	----	---	----	-----	-----	------

Приоритет операторов

- Скобки: () и []
 - Унарные операторы: +, -, ++, --, !
 - Арифметические операторы и операторы сдвига : *, /, %, +, -, >>, <<
 - Операторы отношений: >, >=, <, <=, ==, !=
 - Логические и битовые операторы: &, ^, |, &&, ||,
 - Условный оператор и операторы присваивания: ?=, =, *=, /=, +=, -=
-
- Скобки используются для изменения порядка выполнения операторов в выражении.
 - Любая часть выражения, заключённая в скобки, выполняется в первую очередь.

Управляющие структуры Java

Набор управляющих конструкций языка Java включает:

- блок **{}**;
- операторы присваивания;
- условный оператор **if**;
- оператор варианта **switch**;
- три оператора цикла **while**, **do-while**, **for**;
- операторы перехода **break**, **continue** и **return**;
- пустой оператор — просто точка с запятой.

Управляющие конструкции

■ Принятие решений

- ❖ Конструкция if-else
- ❖ Конструкция switch-case

■ Циклы

- ❖ Цикл while
- ❖ Цикл do-while
- ❖ Цикл for

Конструкция if-else

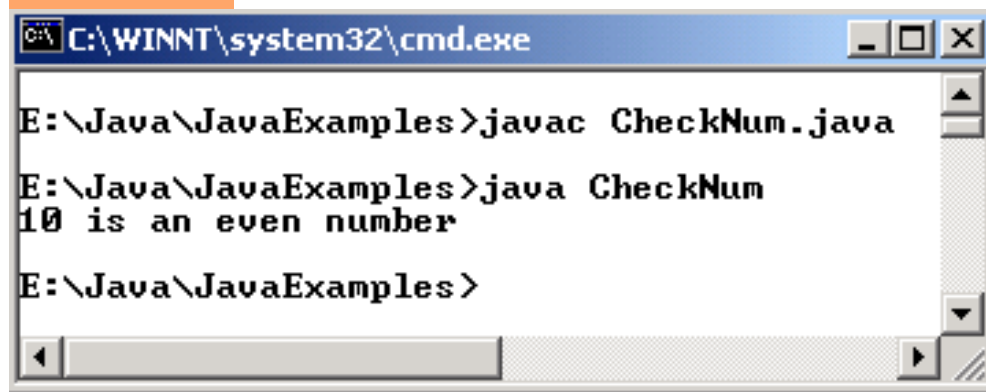
- Конструкция `if-else` проверяет результат условного выражения (`condition`) и выполняет соответствующие действия на основании полученного результата.
- Может быть использована для направления выполнения программы по двум различным путям.
- Формат конструкции `if-else` очень прост. Он приведён ниже:

```
if (condition)
{
    action1;
}
else
{
    action2;
}
```

Пример

```
class CheckNum
{
    public static void main(String [] args)
    {
        int num = 10;
        if (num % 2 == 0)
            System.out.println(num + " is an even number");
        else
            System.out.println(num + " is an odd number");
    }
}
```

Вывод



The screenshot shows a Windows command prompt window with the title bar "C:\WINNT\system32\cmd.exe". The command history is as follows:

```
E:\Java\JavaExamples>javac CheckNum.java
E:\Java\JavaExamples>java CheckNum
10 is an even number
E:\Java\JavaExamples>
```

Конструкция `switch – case`

- Конструкция `switch – case` может быть использована вместо конструкции `if-else-if`.
- Применяется в ситуациях, когда результатом вычисляемого выражения могут быть многочисленные альтернативные значения.
- Использование конструкции `switch-case` ведёт к упрощению кода и к улучшению производительности.

Пример

```
switch (целВыр){  
    case констВыр1: оператор1;  
    case констВыр2: оператор2 ;  
    . . . . .  
    case констВыр N: операторN;  
    default: операторOef ;  
}
```

Цикл `while`

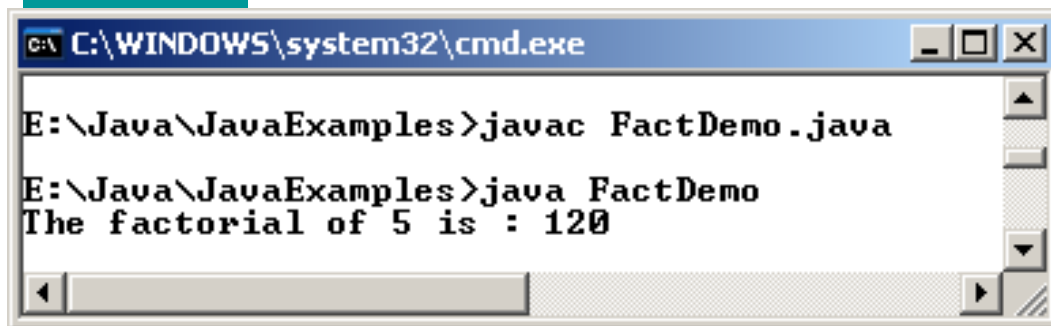
- Цикл `while` используется в ситуациях, когда необходимо многократное повторное выполнение некоторого набора операций, если заданное начальное условие является истинным (`True`).
- Количество повторений выполнения цикла не определено заранее, но зависит от начального условия (`condition`).
- Синтаксис:

```
while (condition)
{
    action statements;
    .
    .
    . }
```

Пример

```
class FactDemo
{
    public static void main(String [] args)
    {
        int num = 5, fact = 1;
        while (num >= 1)
        {
            fact *= num;
            num--;
        }
        System.out.println("The factorial of 5 is : " + fact);
    }
}
```

Вывод



The screenshot shows a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe". The prompt is at "E:\Java\JavaExamples>". The user has entered "javac FactDemo.java" and the prompt has moved to the next line. Then, the user has entered "java FactDemo" and the output "The factorial of 5 is : 120" has been displayed. The window has standard Windows controls (minimize, maximize, close) in the top right corner and a scrollbar at the bottom.

```
C:\WINDOWS\system32\cmd.exe
E:\Java\JavaExamples>javac FactDemo.java
E:\Java\JavaExamples>java FactDemo
The factorial of 5 is : 120
```

Цикл do – while

- Цикл `do-while` выполняет определённые операции до тех пор, пока заданное условие (`condition`) является истинным (`True`).
- Этот цикл похож на цикл `while`, за исключением того, что цикл `do-while` выполняется по меньшей мере один раз, даже если заданное условие `condition` является ложным (`False`).
- Синтаксис:

```
do
{
    action statements;
.
.
} while (condition);
```

Пример

```
class DoWhileDemo {  
    public static void main(String [] args){  
        int count = 1, sum = 0;  
        do  
        {  
            sum += count;  
            count++;  
        }while (count <= 100);  
  
        System.out.println("The sum of first 100 numbers is : " +  
sum);  
    }  
}
```

ВЫВОД

The sum of first 100 numbers is : 5050

Цикл for

- Все циклы обладают некоторыми общими свойствами: переменная-счётчик цикла, которая инициализируется перед началом цикла, условное выражение, которое проверяет переменную-счётчик, и инструкция, которая изменяет значение переменной-счётчика.
- Цикл `for` предоставляет компактный формат записи всех этих свойств.

Синтаксис:

```
for (initialization statements; condition; increment / decrement  
    statements)  
{  
    action statements;  
    .  
    .  
}
```

Пример

```
class ForDemo
{
    public static void main(String [] args)
    {
        int count = 1, sum = 0;
        for (count = 1; count <= 10; count += 2)
        {
            sum += count;
        }
        System.out.println("The sum of first 5 odd numbers is : " + sum);
    }
}
```

Вывод

The sum of first 5 odd numbers is : 25

Инструкции перехода

- Существуют три инструкции перехода (jump):
 - ❖ `break`
 - ❖ `continue`
 - ❖ `return`
- Инструкцию `break` используют в случаях:
 - ❖ Завершение выполнения последовательности команд в конструкции `switch`.
 - ❖ Для выхода из цикла.

Пример

```
class BrDemoAppl
{
    public static void main(String [] args)
    {
        for (int count = 1; count <= 100; count++)
        {
            if (count == 10)
                break;
            System.out.println("The value of num is : " + count);
        }
        System.out.println("The loop is over");
    }
}
```

Вывод

```
The value of num is : 1
The value of num is : 2
The value of num is : 3
The value of num is : 4
The value of num is : 5
The value of num is : 6
The value of num is : 7
The value of num is : 8
The value of num is : 9
The loop is over
```

Вопросы

- Какие виды комментариев в Java вы знаете?
- Обязателен ли метод `main()` в java-приложении?
- Перечислите типы данных языка Java.
- Какие операторы вы знаете?
- Перечислите управляющие конструкции.
- Какие операторы переходы вы знаете?