

# Операции

- **Условная операция**

$\langle \text{условие} \rangle ? \langle \text{выражение1} \rangle : \langle \text{выражение2} \rangle$

- Если  $\langle \text{условие} \rangle$  истинно, то результатом будет  $\langle \text{выражение1} \rangle$ , иначе  $\langle \text{выражение2} \rangle$ .
- Например,  $\text{int } x = a < b ? a : b$  вычисляет минимум из  $a$  и  $b$ .

# Операции

- **Операция приведения типов**

Например, пусть метод `f(...)` выдает `long`.

```
int x = (int)f(10);
```

Здесь `(int)` — это операция преобразования типа. Операция преобразования типа обозначается при помощи имени типа, взятого в скобки.

Эта операция применима не только к базовым типам, но и к классам

[старшинство](#)

# Литералы

Литерал в программировании - адресная, числовая или символьная константа, непосредственно включаемая в операторы или команды программы (в отличие от данных, обращение к которым производится посредством их идентификаторов)

# Литералы

Литералы позволяют задать в программе значения для

- ЧИСЛОВЫХ,
- СИМВОЛЬНЫХ
- строковых выражений,
- null-литералов.

# Литералы

**в Java определены следующие виды литералов:**

- целочисленный (integer);
- дробный (floating-point);
- булевский (boolean);
- символьный (character);
- строковый (string);
- null-литерал (null-literal).

# Литералы

**Целочисленные литералы позволяют задавать целочисленные значения в десятичном, восьмеричном и шестнадцатеричном виде.**

`0, 00, 0x0`

`0xaBcDeF, 0xCafe, 0xDEС`

# Литералы

## Дробные литералы

3.14    2.    .5    7e10    3.1E-20

## Логические литералы

`true` и `false`.

# Литералы

**Символьные литералы описывают  
один символ из набора Unicode**

'a' // латинская буква a

' ' // пробел

'\u0041' // латинская буква A

'\u0410' // русская буква А

'\u0391' // греческая буква Α



# Литералы

**Символьный литерал может содержать последовательность, начинающуюся с**

**\**

**\b backspace BS – забой**

**\t horizontal tab HT – табуляция**

**\n linefeed LF – конец строки**

**\f form feed FF – конец страницы**

**\r carriage return CR – возврат каретки**

**\" double quote " – двойная кавычка**

**' single quote ' – одинарная кавычка**

**\\ backslash \ – обратная косая черта**

# Литералы

**Строковые литералы состоят из набора символов и записываются в двойных кавычках.**

**Длина может быть нулевой или сколь угодно большой.**

**Любой символ может быть представлен специальной последовательностью, начинающейся с \**

**"Hello, world!\r\nHello!"**

# Литералы

**Null-литерал может принимать всего одно значение: null**

**Это литерал ссылочного типа, причем эта ссылка никуда не ссылается, объект отсутствует**

**Разделители**

**( ) [ ] { } ; . ,**

# Типы данных

- в Java-программе переменные должны быть описаны до их использования
- Синтаксис:

**<Тип><переменная-1>,<переменная-2>...;**

**int style, number; // 4 байта**

**char answer; //2 байта**

**double amount, inter = 3.31456; // 4 байта**

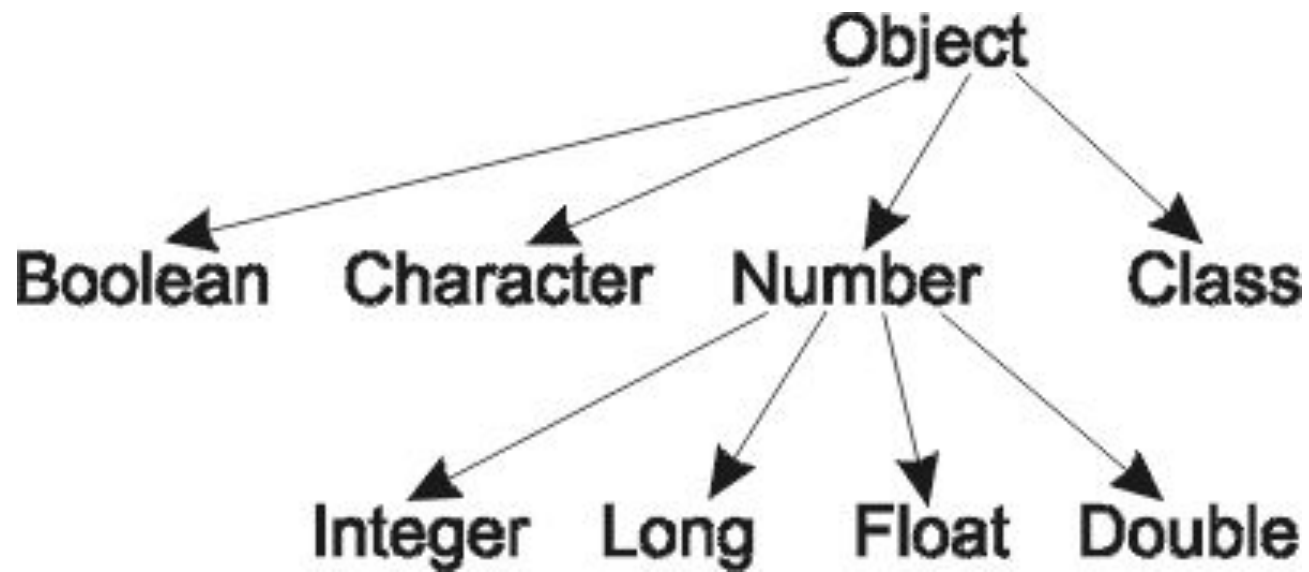
**boolean b1=false; // 1байт**

**final double pi=3.1415; // 8 байтов**

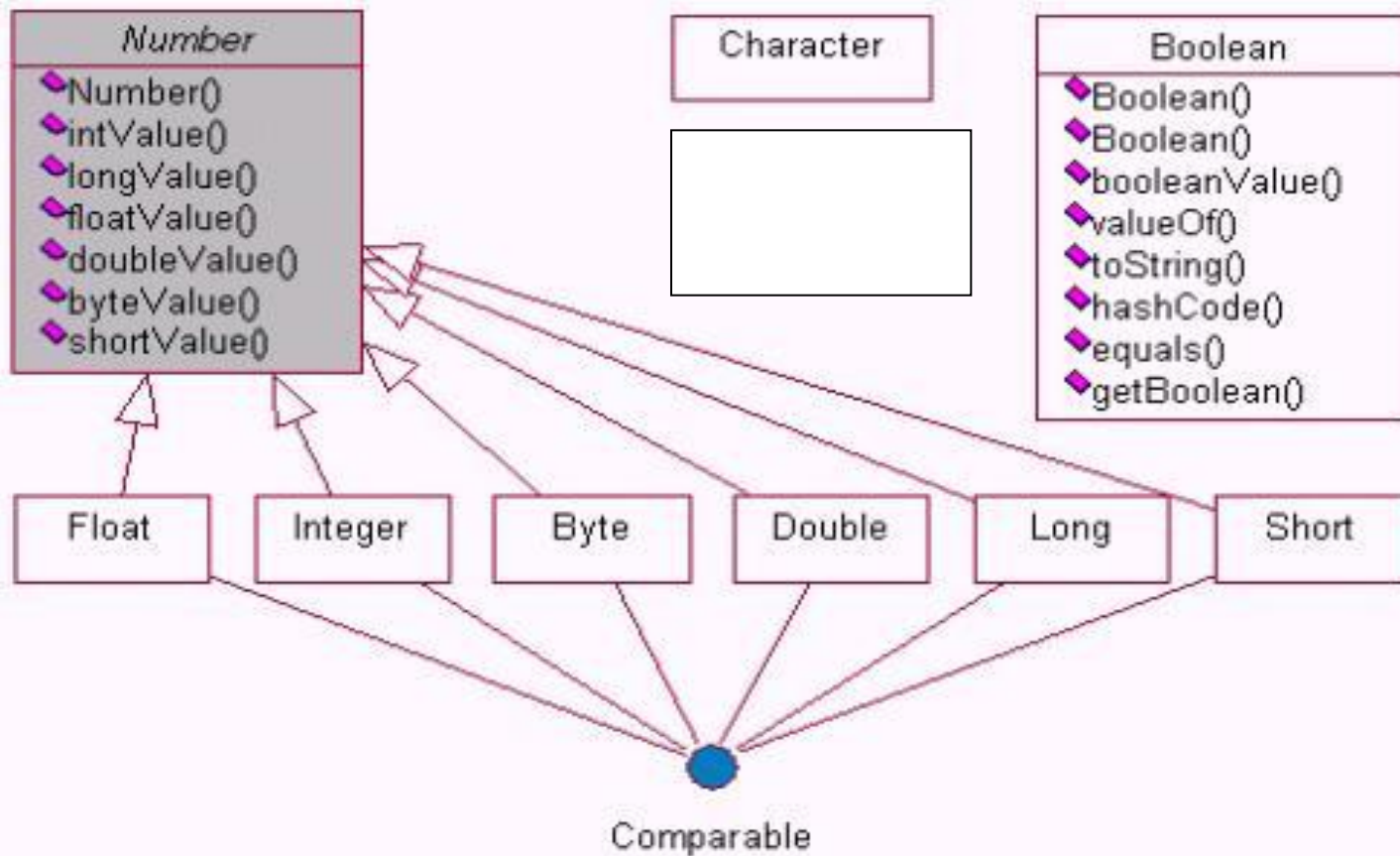
# Типы данных

- Кроме базовых типов данных широко используются соответствующие классы (wrapper – classes): **Boolean, Character, Integer, Byte, Short, Long, Float, Double.**
- Объекты этих классов могут хранить те же значения, что и соответствующие им базовые типы

# Типы данных



# Типы данных



# Типы данных

```
if (value >= Float.MIN_VALUE && value <=
    Float.MAX_VALUE) .....
```

Создание объекта класса **Integer**

```
key = 123;
```

```
Integer keyObj = new Integer(key);
```



# Операторы присваивания

- **class BasicMath {**  
**public static void main(String args[])**  
**{**  
**int a = 1 + 1;**  
**int b = a \* 3;**  
**int c = b / 4;**  
**int d = b - a \* Math.sqrt(4.0);**  
**int e = -d; ...**

# Операторы присваивания

```
}  
} //a = 2 b = 6 c = 1 d = 8 e = -4
```

В арифметических выражениях автоматически выполняются расширяющие преобразования типа `byte` `short` `int` `long` `float` `double`

Для сужающих преобразований необходимо производить явное преобразование вида (тип) значение.

Например: `byte b=(byte)35;`

# Операторы присваивания

```
class IncDec  
{  
  public static void main(String args[])  
  {  
    int a = 1;  
    int b = 2;  
    int c = ++b;  
  }  
}
```

# Операторы присваивания

```
int d = a++;
```

```
c++; ...
```

```
} //a = 2 b = 3 c = 4 d = 1
```

```
}
```

# Условные операторы

***if** (логическое выражение) оператор1;  
[ **else** оператор2;]*

- Раздел **else** необязателен.
- На месте любого из операторов может стоять составной оператор (несколько операторов, заключенных в фигурные скобки).
- *Логическое выражение* — это любое выражение, возвращающее значение типа `boolean`

# Условный оператор

1. <оператор-0> ;

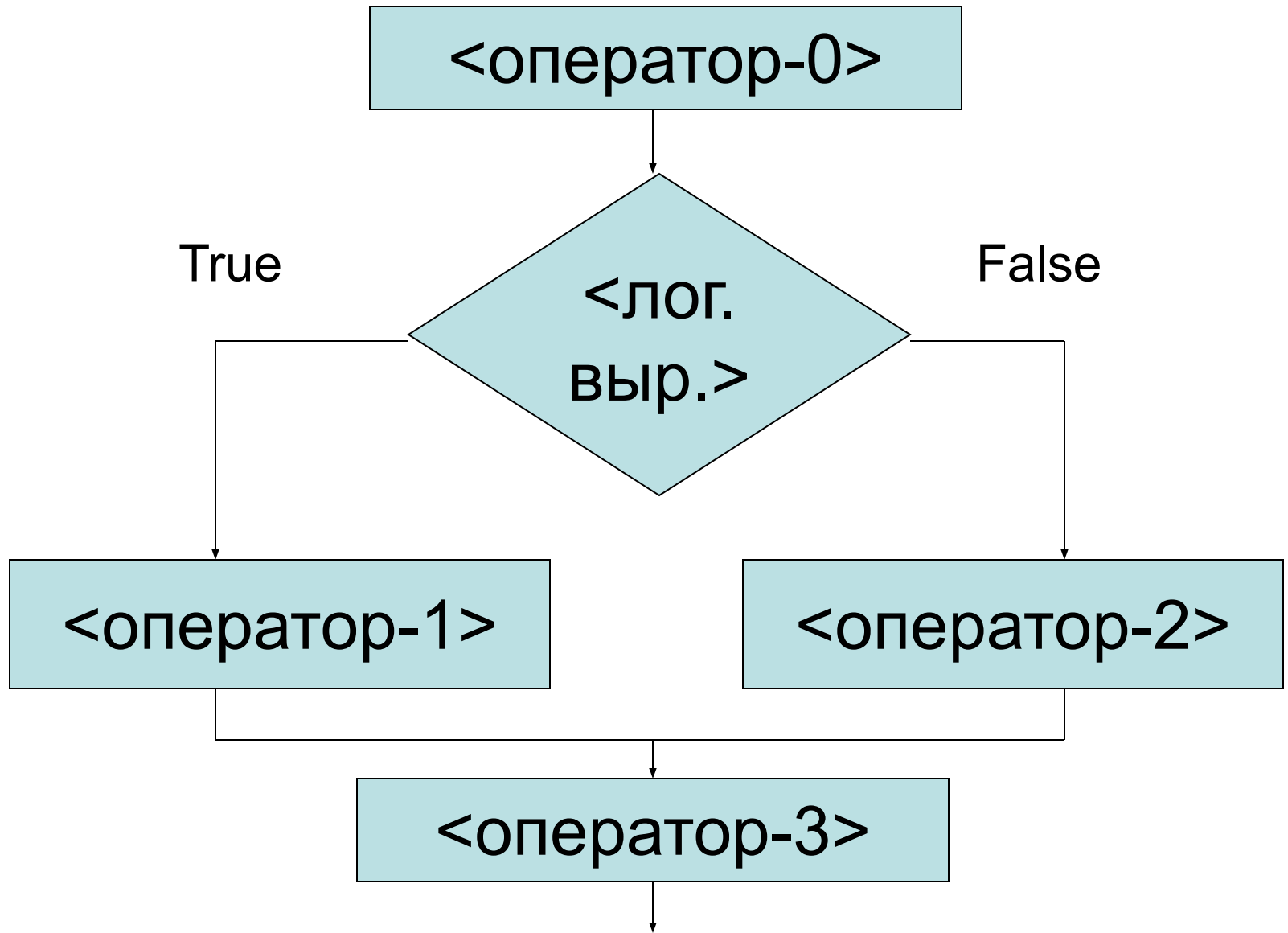
**if** <логическое выражение>

<оператор-1>;

**else** <оператор-2>;

<оператор-3>;

# Условный оператор

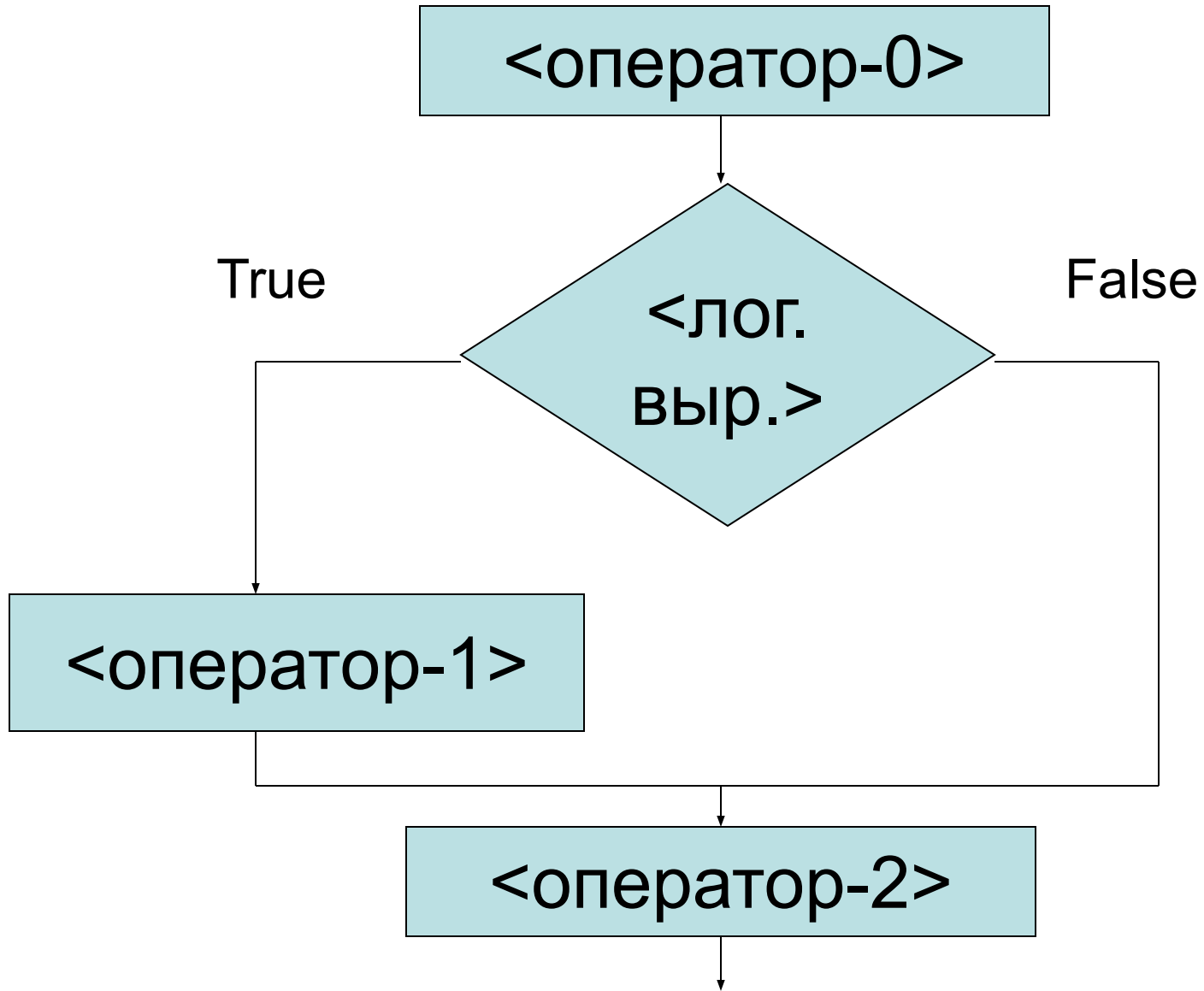


# Условный оператор

1. <оператор-0> ;  
    **if** <логическое выражение>  
        <оператор-1>;  
    <оператор-2>



# Условный оператор



# Условные операторы

1. **if** (a>b)

    y = a;

**else** y = b;

2. int x = 5;

**if**( x < 4)

{ System.out.println("Меньше 4"); }

**else if** (x > 4)

{ System.out.println("Больше 4"); }

**else if** (x == 4) { System.out.println("Равно 4"); }

**else**{ System.out.println("Другое значение");

}

# Условные операторы

```
public class IfElseDemo {  
    public static void main(String[] args) {  
        int testscore = 76;  
        char grade;  
        if (testscore >= 90) { grade = 'A'; }  
        else if (testscore >= 80) {grade = 'B'; }  
        else if (testscore >= 70) { grade = 'C'; }  
        else if (testscore >= 60) { grade = 'D'; }  
        else { grade = 'F'; }  
        System.out.println("Grade = " + grade);  
    }  
}
```