

Типы данных

Целочисленные

- byte
- short
- int
- long
- char (также является целочисленным типом)

Дробные

- float
- double

Булевые

- boolean

Переменные

- ИМЯ;
- ТИП;
- значение.

Примеры

- `int a;`
- `int b = 0, c = 3+2;`
- `int d = b+c;`
- `int e = a = 5;`

final

- Ключевое слово `final` указывают перед типом переменной. Тогда ее необходимо сразу инициализировать и уже больше никогда не менять ее значение. Таким образом, `final`-переменные становятся чем-то вроде констант, но на самом деле некоторые инициализаторы могут вычисляться только во время исполнения программы, генерируя различные значения.
- `final double pi=3.1415;`

Примитивные vs Ссылочные ТИПЫ

- `int a=5;`
- `int b=a;`
- `a=3;`
- `print(b);`

Примитивные vs Ссылочные типы (2)

```
class Point {  
    int x, y;  
}
```

```
Point p1 = new Point(3,5);
```

```
Point p2=p1;
```

```
p1.x=7;
```

```
print(p2.x);
```

Примитивные vs Ссылочные типы

(3)

- `Point p1 = new Point(3,5);`
- `Point p2=p1;`
- `p1 = new Point(7,9);`
- `print(p2.x);`

Целочисленные типы данных

Название типа	Длина (байты)	Область значений
byte	1	-128 .. 127
short	2	-32.768 .. 32.767
int	4	-2.147.483.648 .. 2.147.483.647
long	8	-9.223.372.036.854.775.808 .. 9.223.372.036.854.775.807 (примерно 10^{19})
char	2	'\u0000' .. '\uffff', или 0 .. 65.535

Допустимые операции

- операции сравнения (возвращают булево значение)
 - $<$, $<=$, $>$, $>=$
 - $==$, $!=$
- числовые операции (возвращают числовое значение)
 - унарные операции $+$ и $-$
 - арифметические операции $+$, $-$, $*$, $/$, $\%$
 - операции инкремента и декремента (в префиксной и постфиксной форме): $++$ и $--$
 - операции битового сдвига $<<$, $>>$, $>>>$
 - битовые операции \sim , $\&$, $|$, \wedge
- оператор с условием $?:$
- оператор приведения типов
- оператор конкатенации со строкой $+$

Дробные типы

Название типа	Длина (байты)	Область значений
float	4	3.40282347e+38f ; 1.40239846e-45f
double	8	1.79769313486231570e+308 ; 4.94065645841246544e-324

Допустимые операции

- операции сравнения (возвращают булево значение)
 - $<$, $<=$, $>$, $>=$
 - $==$, $!=$
- числовые операции (возвращают числовое значение)
 - унарные операции $+$ и $-$
 - арифметические операции $+$, $-$, $*$, $/$, $\%$
 - операции инкремента и декремента (в префиксной и постфиксной форме): $++$ и $--$
- оператор с условием $?$:
- оператор приведения типов
- оператор конкатенации со строкой $+$

Специальные значения дробного типа

- положительная и отрицательная бесконечности (positive/negative infinity);
- значение "не число", Not-a-Number, сокращенно NaN ;
- положительный и отрицательный нули.

Специальные значения дробного типа (2)

- Положительную и отрицательную бесконечности можно получить следующим образом:
- `1f/0f` // положительная бесконечность, тип `float`
- `-1d/0d` // отрицательная бесконечность, тип `double`

Специальные значения дробного типа (3)

- Значение NaN можно получить, например, в результате следующих действий:
- $0.0/0.0$ // деление ноль на ноль
- $(1.0/0.0)*0.0$ // умножение бесконечности на ноль

Специальные значения дробного типа (4)

- Величины положительный и отрицательный ноль записываются очевидным образом:
- `0.0` // дробный литерал со значением положительного нуля
- `+0.0` // унарная операция `+`, ее значение - положительный ноль
- `-0.0` // унарная операция `-`, ее значение - отрицательный ноль

Булев тип

- Два возможных значения – true и false.

Допустимые операции

- операции сравнения (возвращают булево значение)
 - ==, !=
- логические операции (возвращают булево значение)
 - !
 - &, |, ^
 - &&, ||
- оператор с условием ?:
- оператор конкатенации со строкой +

Ссылочные типы

- Выражение ссылочного типа имеет значение либо `null`, либо ссылку, указывающую на некоторый объект в виртуальной памяти JVM.

Допустимые операции

- обращение к полям и методам объекта
- оператор `instanceof` (возвращает булево значение)
- операции сравнения `==` и `!=` (возвращают булево значение)
- оператор приведения типов
- оператор с условием `?:`
- оператор конкатенации со строкой `+`

instanceof

- Используя оператор instanceof, можно узнать, от какого класса произошел объект. Этот оператор имеет два аргумента. Слева указывается ссылка на объект, а справа – имя типа, на совместимость с которым проверяется объект.

Пример

- `class Parent { }`
- `class Child extends Parent { }`
- `class Child2 extends Parent { }`
- `Parent p=new Child();`
- `print(p instanceof Child);`
- `print(p instanceof Child2);`