

Java Data Types

byte/short/int/long	2 ⁸ (8/ 16/ 32/ 64)
float [32bit] / double [64bit]	1.0f / 1.0d, 1.0
char [16bit]	"U", "±"
String	"Hello World"

Literals

2 ⇒ int	2.0F ⇒ float
2L ⇒ long	2.0 ; 2.0D ⇒ double
"u" ⇒ String	'u' ⇒ char
true ⇒ boolean	false ⇒ boolean

number literal modifiers are case-insensitive

Inc- / Decrement

a++	++a
a--	--a
→ return a++ / a--	→ return a

Assignment shortcuts: x op= y

Exp: x += 2;

Variables

```
int i = 5, j;
int a = 3, b = a + 1;
final int c; //=> Constant
c = 22; // may be init after
```

Comments

```
// Single Line
/* Multi Line */
/** Docu String */
```

Array

```
int[] a; //Declaration
a = new int[5] //Dimension
int[] a = new int[5];
int[] b = {10, 20, 30};
int[][] matrix = new int[2][3];
```

Init Values: 0, "\0", false, null

Array Methods

```
int[] a;
a.length; //length of array
```

ArrayList

```
ArrayList<Double> nums = new
ArrayList<>();
nums.add(2.3);
nums.size() == 1
double a = nums.get(0);
```

Like a list in python

Type Casting

```
int a, b;
a = (double) b * a //b => double
a = b * (double) a //a => double
a = (double) (a * b)
/* (a * b) will be calculated
with int logic then
type-casted to double */
```

Strings

1. Strings are a class in Java.
2. Concating is possible without type-cast
String a = "Num: " + 5;
3. Parsing:
int i = Integer.parseInt("22");
float f = Float.parseFloat("1.3");
boolean b = Boolean.parseBoolean("True");

String Functions

```
s.equals(String s2) -> bool
s.toLowerCase()
s.toUpperCase()
s.replace(char old, char new)
s.replace(String old, String new)
s.indexOf(String s) //-1 if not available
s.lastIndexOf(String s)
s.split(String delimiter) -> String[]
```

all functions return the modified String.
They don't modify the original String.

Bitwise Operations

int	boolean	
	!	NOT
&	&&	AND
		OR
^	^	XOR

Biwise Shifts

~	Complement
<<	Shift left
>>	Shift right
>>>	Shift right Zero fill

Arithmetic Operator Results (+ - * / %)

Both Arguments

byte, short, int ⇒ int

One, or Both Arguments

long ⇒ long

float ⇒ float

double ⇒ double

Examples: byte * byte ⇒ int;

double + float ⇒ double

Java Operations Order

1. Members () [] .
2. Multiplikation * / %
3. Addition + -
4. XOR ^
5. Logical AND &&
6. Logical OR ||

a++, a-- counts as 3. Addition

```
int a = 5, b = 8;
```

```
int c = a * b++ //c is 40;
```

Java Naming Convention

Constants:

MAX, PI, MIN_TIME

Variables, Methods, Packages:

xVal, int1Arr, date, showDate

Classes:

Date, DatabaseHelper



File IO

```
File file = new
File("text.txt");
file.exists();
file.createNewFile();
file.delete();
BufferedReader = new BufferedR-
eader(file);
while ((line = input.readLine())
!= null) {
//statements with line
}
```

There are other classes, but just use
BufferedReader

abstract / Interface

abstract Method

```
public abstract fun();
```

abstract Class

```
public abstract class Test{}
```

Interface

Like abstract class, but with only abstract functions. You don't need abstract for these

Abstract Classes and Methods are without implementation.

You use implements for Interfaces

UI

```
// Pop-Up Box:
import javax.swing.*;
String out = JOptionPane.showIn-
putDialog("Inp: ");
```

Regular Expressions (Regex)

```
Pattern p =
Pattern.compile("\\w+");
Matcher m = p.matcher("abc");
while (m.find()) {
m.group()
}
//true if whole string matches
m.matches() -> bool
```

Bolierplate

```
//Syntax for a standalone
application in Java:
class <classname>
{
public static void main(S-
tring args[]) {
statements;
}
}
```

Bolierplate cont.

Steps to run the above application:

1. Type the program in IntelliJ or notepad. Save the file with a .java extension.
2. The file name should be the same as the class, which has the main method.
3. To compile the program, using javac compiler, type the following on the command line:
Syntax: javac <filename.java>
Example: javac abc.java
4. After compilation, run the program using the Java interpreter.
Syntax: java <filename> (without the .java extension)
Example: java abc
5. The program output will be displayed on the command line

Java Statements

If Statement

```
if ( expression ) {
statements
} else if ( expression ) {
statements
} else {
statements
}
```

While Loop

```
while ( expression ) {
```

Java Statements (cont)

```
statements
}
Do-While Loop
do {
statements
} while ( expression );
For Loop
for ( int i = 0; i < max; ++i) {
statements
}
For Each Loop
for ( vartype var : collection )
{
statements
}
Switch Statement
switch ( expression ) {
case value:
statements
break;
case value2:
statements
break;
default:
statements
}
Exception Handling
try {
statements;
} catch (ExceptionType e1) {
statements;
} catch (Exception e2) {
catch-all statements;
} finally {
statements;
}
```



IntelliJ Emmets

```
psvm    public static void main(String[] args) {}
```

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



By **Welten**
cheatography.com/welten/

Published 7th July, 2021.
Last updated 12th July, 2021.
Page 3 of 3.

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
<http://crosswordcheats.com>