

INPUT & OUTPUT

INPUT

To use Scanner, import Scanner library:

```
import java.Object.Scanner
```

```
Scanner scan = new Scanner(System.in);
```

```
name = scan.nextInt();
```

```
name = scan.nextLine();
```

```
name = scan.nextDouble();
```

OUTPUT

```
System.out.print ( "Hello" );
```

```
System.out.println ( "Hello" ); // New line.
```

COMMENTS

```
// A single line comment.
```

```
/*
```

```
* Multiple
```

```
* line comments.
```

```
*/
```

JAVADOC

```
package-info.java //Need to generate.
```

```
/**
```

```
* Javadoc comments generate a HTML file.
```

```
* @author
```

```
* @version
```

```
*/
```

VARIABLES

PRIMITIVE DATA TYPES

```
int value = null; //Doesn't compile.
```

```
byte 8 bits
```

```
short 16 bits
```

```
int 32 bits
```

```
long 64 bits
```

```
char Textual: (16 bits)
```

```
boolean Logical: true/false
```

```
float (decimal) Floats 15 places (32 bits)
```

```
double (decimal) Floats 15 places (64 bits)
```

FINAL VARIABLES

☞ Permanent Values

```
final double PI = 3.14159265;
```

VARIABLES (cont)

```
final String LANGUAGE = "Java";
```

KEYWORDS

```
abstract final/ly public
```

```
assert float requires
```

```
boolean for return
```

```
break goto short
```

```
byte if static
```

```
case implements strictfp
```

```
catch import super
```

```
char instanceof switch
```

```
class int synchronized
```

```
continue interface this
```

```
const long throw/s
```

```
default module transient
```

```
do native true
```

```
double new try
```

```
else null var
```

```
enum package void
```

```
exports private volatile
```

```
extends protected while
```

```
false
```

STRINGS

REFERENCE DATA TYPE

```
String name = "Always within quotes";
```

```
String words = "123 have no value";
```

STRING CONCATENATION

```
System.out.println("Hello " + variable + ".");
```

```
System.out.print (variable + " Hello World");
```

CONCATENATION RULES

```
number + number = number
```

```
number + String = number + String
```

```
String + number = String
```

Above number changed to a String!

```
String + (number) = String + number
```

STRINGS (cont)

Use () to override String change.

CALCULATIONS

COMPOUND ASSIGNMENT OPERATORS

Syntax to assign arithmetic operation result **back-into** "assignment" variable.

```
+= Add values
```

```
-= Minus values
```

```
*= Multiply values
```

```
/= Divide values
```

```
%= Modulus (remainder)
```

```
++ Adds one
```

```
-- Subtracts one
```

```
x = x + 1; //adds one to x
```

```
x += 1; //adds one to x
```

```
x++; //adds one to x
```

MODULAR DIVISION (%)

REMAINDER

Only works on integers!

```
7 % 2 = remainder 1 8 % 3 = remainder 2
```

```
Find even numbers x % 2 = remainder 0
```

```
Find odd numbers x % 2 = remainder 1
```

MINIMUM & MAXIMUM

```
Integer.MIN_VALUE ↔ Integer.MAX_VALUE
```

```
2147483647 ↔ -2147483648
```

NAMING CONVENTIONS

Package **java, lang, shapes**, etc...

package.name, java.io, etc...

com.example.mypackage, etc...

Lowercase letters.

Separate words with dots (.)

Companies: "reversed Internet domain" starts their package name.

NAMING CONVENTIONS (cont)

Class **Color, Button, Cats**, etc...
CambridgeCitySchools, etc...
 NOUNS: Each word starts with an uppercase letter. ☒ No acronyms.

Method **main(), print(), equals()**, etc...
newMethod(), toString() etc...
 VERB: First word lowercase.
 Adding words: First letter uppercase.

Variable **area, name, size**, etc...
newTriangle, rightSide, etc...
 IDENTIFIER: first word lowercase.
 Adding words; first letter uppercase.
 ☒ No keywords.
 ☒ No special characters to start.
 ☒ Avoid one letter: x, y, z, etc.

Final **PI, BLUE, MATH**, etc...
Constant **MAX_PRIORITY, CITY_ZIP**, etc...
 All uppercase letters: Every word separated with an underscore(_).

TYPE CASTING

(double) 1/3 = 0.3333 Results a double

int value = (int) 4.7 Narrowing Cast

(int) 3.86 = 3 Truncated

WIDENING & NARROWING CASTING

Does not round: Truncates!

byte→short→char→int→long→float→double

Narrowing Casting (manually)

larger → smaller

Widening Casting (*automatic*)

smaller → larger

ROUNDING

int nearestInt = (int) (number + 0.5)

int nearestNegInt = (int) (negNumber - 0.5)



By **KETHEO**
cheatography.com/ketheo/

Published 26th October, 2021.
 Last updated 26th October, 2021.
 Page 2 of 2.

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