

# Ultimate Java Quick Reference - CodeWithHarry

1. <code>// [comment]</code> Single line comment.	10. <code>Private</code> Can only be changed by a method.	64-bit number with decimals.
2. <code>/* [comment] */</code> Multi line comment.	11. <code>int</code> Can store numbers from $2^{-31}$ to $2^{31}$ .	19. <code>float</code> 32-bit number with decimals.
3. <code>public</code> This can be imported publically.	12. fields are attributes	20. <code>protected</code> Can only be accessed by other code in the package.
4. <code>import [object].*</code> Imports everything in object.	13. <code>boolean</code> Can have true or false as the value.	21. <code>Scanner</code> This lets you get user input.
5. <code>static</code> Going to be shared by every [object].	14. <code>{ }</code> These are used to start and end a function, class, etc.	22. <code>new [object constructor]</code> This will let you create a new object.
6. <code>final</code> Cannot be changed; common to be defined with all uppercase.	15. <code>byte</code> These can store from -127 - 128.	23. <code>System.in</code> This lets you get data from the keyboard.
7. <code>double</code> Integer with numbers that can have decimals.	16. <code>long</code> Can store numbers from $2^{127}$ to $2^{-127}$ .	24. <code>public [class]()</code> This will be the constructor, you use it to create new objects.
8. <code>;</code> Put after every command.	17. <code>char</code> Just lets you put in one chracter.	25. <code>super()</code> This will create the superclass (the class it's inheriting).
9. <code>String</code> Just a string of characters.	18. <code>double</code>	

## Ultimate Java Quick Reference - CodeWithHarry

26. extends [class] Makes the object a subclass of [object], [object] must be a superclass.	35. public static void main(String[] args) This is your main function and your project will start in here.	44. < This means less than.
27. ++ Will increment the amount.	36. System.out.print([text]) This prints stuff but there is no line break. (/n)	45. > This means greater than.
28. -- Will decrement the amount.	37. \n Called a line break; will print a new line.	46. >= This means greater than or equal to.
29. += [amount] Increment by [amount]	38. \t This will print a tab.	47. [inputVarHere].hasNextLine() This will return if there is a next line in the input.
30. -= [amount] Decrement by [amount]	39. if ([condition]) This will make it so if [condition] is true then it'll keep going.	48. this Refer to the class that you are in.
31. *= [amount] Multiply by [amount]	40. && This means and.	49. [caller].next[datatype]() This will get the [datatype] that you somehow inputted.
32. /= [amount] Divide by [amount]	41. ! This means not.	50. Create getters and setters This will create the get methods and set methods for every checked variable.
33. System.out.println([text]) Will print something to the output console.	42.    This means or.	51. [caller].hasNext[datatype]()
34. + Can be used for concatenation. (ex. "6" + [var_here])	43. == This means equal to.	

# Ultimate Java Quick Reference - CodeWithHarry

This will return if it has the correct datatype within the input.

## 52. overloading

If you have different parameters you can call them whatever way you want.

## 53. parameters

These are the inputs of your function.

## 54. ([datatype])[variable]

This will convert [variable] into [datatype]. Also known as casting.

## 55. Math.random()

Generate an extremely precise string of numbers between 0 and 1.

## 56. Primitives

Just the basic data types which are not objects.

## 57. [x].toString()

Will convert [x] into a string.

## 58.

[number].parse[numbertype]([string])

This will parse [number] into the [numbertype] with [string].

## 59. ^

Return true if there is one true and one false.

## 60. !=

Not equal too. (NEQ)

## 61. ([condition]) ? [amount] : [var]

This will be like a shortcut way to an if statement.

## 62. switch([variable])

This will do stuff with specific cases. (e.g. switch(hi){ case 2: (do stuff)})

## 63. case [value]:

This will do stuff if the case is the case.

## 64. break

Put that when you want to leave the loop/switch; should be at end of case.

## 65. default [value]:

This will do stuff if none of the cases in the switch statement was made.

## 66. for ([number]; [condition]; [operation])

This will start at [number] and then do [operation] until [condition] is met.

## 67. continue

This will just go back to the enclosing loop before reaching other code.

## 68. while ([condition])

This will basically do something while [condition] is true.

## 69. void

This means no return type.

## 70. return

This will return something when you call it to where it was called from .

## 71. do { } while ([condition])

Guarantees it will execute once even if [condition] isn't met.

## 72. printf("%[type] stuff here bah bla", [variable here])

This will let you use [variable here] with %s being where.

## Ultimate Java Quick Reference - CodeWithHarry

73. `System.out.printf([text])`

Another way to print? //  
didn't quite get but ok then

This will get how long  
something is, text, amount  
of indexes in array, etc.

74. `[type] [returntype]  
[name]([parameters]) {`

This is a way to create a  
method.

80. `Arrays.copyOf([array],  
indexes);`

This will copy the array and  
how many indexes into  
another array.

75. `[type][[indexes]]`

This will create an array  
with [indexes] amount of  
indexes; default infinite.

81. `Arrays.toString([array])`

Convert the whole array  
into one huge string.

76. `int[] something = new  
int[20];`

This will just make an array  
of ints with 20 ints in it.

82.  
`Arrays.binarySearch([array],  
[object])`

This will search for [object]  
in [array].

77. `for ([object]  
[nameOfObject] :  
[arrayOfObject]) {`

This will iterate through all  
of the arrayOfObject with  
object in use incrementing  
by 1 until done.

78. `[object][[1]][[2]][[3]]  
[name] = {[value] [value]  
[value] \n [value] [value]  
[value]}`

[1] is how many down in  
array, [2] how many accross  
in array, [3] how many  
groups

79. `.length`