Java

This cheat sheet is a crash course for Java beginners and help review the basic syntax of the Java language.

Getting Started

Hello.java

```
public class Hello {
  // main method
  public static void main(String[] args)
  {
    // Output: Hello, world!
    System.out.println("Hello, world!");
  }
}
```

Compiling and running

```
$ javac Hello.java
$ java Hello
Hello, world!
```

Variables

```
int num = 5;
float floatNum = 5.99f;
char letter = 'D';
boolean bool = true;
String site = "quickref.me";
```

Primitive Data Types

Data Type	Size	Default	Range
byte	1 byte	0	-128 to 127

Data Type	Size	Default	Range
short	2 byte	0	-2^{15} to $2^{15}-1$
int	4 byte	0	-2^{31} to $2^{31}-1$
long	8 byte	0	-2^{63} to $2^{63}-1$
float	4 byte	0.0f	N/A
double	8 byte	0.0d	N/A
char	2 byte	\u0000	0 to 65535

Strings

```
String first = "John";
String last = "Doe";
String name = first + " " + last;
System.out.println(name);
```

See: Strings

Loops

```
String word = "QuickRef";
for (char c: word.toCharArray()) {
  System.out.print(c + "-");
}
// Outputs: Q-u-i-c-k-R-e-f-
```

See: Loops

Arrays

```
char[] chars = new char[10];
chars[0] = 'a'
chars[1] = 'b'

String[] letters = {"A", "B", "C"};
int[] mylist = {100, 200};
boolean[] answers = {true, false};
```

See: Arrays

Swap

```
int a = 1;
int b = 2;
System.out.println(a + " " + b); // 1 2

int temp = a;
a = b;
b = temp;
System.out.println(a + " " + b); // 2 1
```

Type Casting

```
// Widening
// byte<short<int<long<float<double
int i = 10;
long l = i;               // 10

// Narrowing
double d = 10.02;
long l = (long)d;         // 10

String.valueOf(10);       // "10"
Integer.parseInt("10");   // 10
Double.parseDouble("10"); // 10.0
```

Conditionals

```
int j = 10;

if (j == 10) {
  System.out.println("I get printed");
} else if (j > 10) {
  System.out.println("I don't");
} else {
  System.out.println("I also don't");
}
```

See: Conditionals

User Input

```
Scanner in = new Scanner(System.in);
String str = in.nextLine();
```

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

int num = in.nextInt();
System.out.println(num);
```

Java Strings

Basic

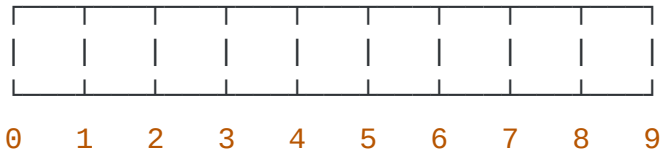
```
String str1 = "value";
String str2 = new String("value");
String str3 = String.valueOf(123);
```

Concatenation

```
String s = 3 + "str" + 3;      // 3str3
String s = 3 + 3 + "str";      // 6str
String s = "3" + 3 + "str";    // 33str
String s = "3" + "3" + "23";   // 3323
String s = "" + 3 + 3 + "23"; // 3323
String s = 3 + 3 + 23;         // 29
```

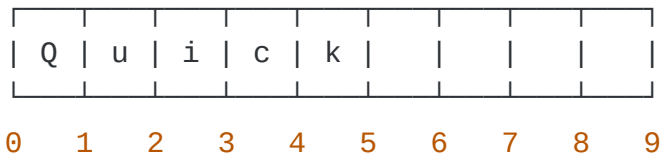
StringBuilder

StringBuilder sb = new StringBuilder(10);

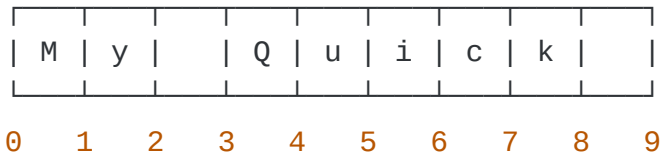


sb.append("QuickRef");

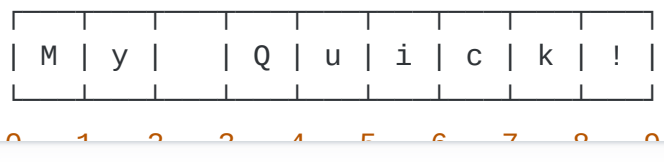
sb.delete(5, 9);



sb.insert(0, "My ");



sb.append("!");



Comparison

```
String s1 = new String("QuickRef");
String s2 = new String("QuickRef");

s1 == s2          // false
s1.equals(s2)     // true

"AB".equalsIgnoreCase("ab")  // true
```

Manipulation

```
String str = "Abcd";

str.toUpperCase();     // ABCD
str.toLowerCase();     // abcd
str.concat("#");       // Abcd#
str.replace("b", "-"); // A-cd

"  abc ".trim();       // abc
"ab".toCharArray();    // {'a', 'b'}
```

Information

```
String str = "abcd";

str.charAt(2);        // c
str.indexOf("a")      // 0
str.indexOf("z")      // -1
str.length();         // 4
str.toString();       // abcd
str.substring(2);     // cd
str.substring(2,3);   // c
str.contains("c");    // true
str.endsWith("d");    // true
str.startsWith("a"); // true
```

Immutable

```
String str = "hello";
str.concat("world");

// Outputs: hello
System.out.println(str);
```

```
String str = "hello";
String concat = str.concat("world");

// Outputs: helloworld
System.out.println(concat);
```

Once created cannot be modified, any modification creates a new String

Java Arrays

Declare

```
int[] a1;
int[] a2 = {1, 2, 3};
int[] a3 = new int[]{1, 2, 3};

int[] a4 = new int[3];
a4[0] = 1;
```

```
a4[2] = 2;
```

Modify

```
int[] a = {1, 2, 3};
System.out.println(a[0]); // 1

a[0] = 9;
System.out.println(a[0]); // 9

System.out.println(a.length); // 3
```

Loop (Read & Modify)

```
int[] arr = {1, 2, 3};
for (int i=0; i < arr.length; i++) {
    arr[i] = arr[i] * 2;
    System.out.print(arr[i] + " ");
}
// Outputs: 2 4 6
```

Loop (Read)

```
String[] arr = {"a", "b", "c"};
for (int a: arr) {
    System.out.print(a + " ");
}
// Outputs: a b c
```

Multidimensional Arrays

```
int[][] matrix = { {1, 2, 3}, {4, 5} };

int x = matrix[1][0];  // 4
// [[1, 2, 3], [4, 5]]
Arrays.deepToString(matrix)

for (int i = 0; i < a.length; ++i) {
  for(int j = 0; j < a[i].length; ++j) {
    System.out.println(a[i][j]);
  }
}
// Outputs: 1 2 3 4 5 6 7
```

Sort

```
char[] chars = {'b', 'a', 'c'};
Arrays.sort(chars);

// [a, b, c]
Arrays.toString(chars);
```

Java Conditionals

Operators

+	-	*	/
%	=	++	--
!			
==	!=	>	>=
<	<=		
&&	||	?:	
instanceof			
~	<<	>>	>>>
&	^	|	

If else

```
int k = 15;
if (k > 20) {
  System.out.println(1);
} else if (k > 10) {
  System.out.println(2);
} else {
```

```
  System.out.println(3);
```

Switch

```
int month = 3;
String str;
switch (month) {
  case 1:
    str = "January";
    break;
  case 2:
    str = "February";
    break;
  case 3:
    str = "March";
    break;
  default:
    str = "Some other month";
    break;
}

// Outputs: Result March
System.out.println("Result " + str);
```

Ternary operator

```
int a = 10;
int b = 20;
int max = (a > b) ? a : b;

// Outputs: 20
System.out.println(max);
```

Java Loops

For Loop

```
for (int i = 0; i < 10; i++) {
  System.out.print(i);
```

```
}
// Outputs: 0123456789
```

```
for (int i = 0,j = 0; i < 3; i++,j--) {
  System.out.print(j + "|" + i + " ");
```

Enhanced For Loop

```
int[] numbers = {1,2,3,4,5};

for (int number: numbers) {
  System.out.print(number);
}
// Outputs: 12345
```

Used to loop around array's or List's

While Loop

```
int count = 0;

while (count < 5) {
  System.out.print(count);
  count++;
}
// Outputs: 01234
```

Do While Loop

```
int count = 0;

do {
  System.out.print(count);
  count++;
} while (count < 5);
// Outputs: 01234
```

Continue Statement

```
for (int i = 0; i < 5; i++) {
  if (i == 3) {
    continue;
  }
  System.out.print(i);
```

```
}
// Outputs: 01245
```

Break Statement

```
for (int i = 0; i < 5; i++) {
  System.out.print(i);
  if (i == 3) {
    break;
  }
}
// Outputs: 0123
```

Java Collections Framework

Java Collections

Collection	Interface	Ordered	Sorted	Thread safe	Duplicate	Nullable
ArrayList	List	Y	N	N	Y	Y
Vector	List	Y	N	Y	Y	Y
LinkedList	List, Deque	Y	N	N	Y	Y
CopyOnWriteArrayList	List	Y	N	Y	Y	Y
HashSet	Set	N	N	N	N	One `null`
LinkedHashSet	Set	Y	N	N	N	One `null`
TreeSet	Set	Y	Y	N	N	N
CopyOnWriteArraySet	Set	Y	N	Y	N	One `null`
ConcurrentSkipListSet	Set	Y	Y	Y	N	N

Collection	Interface	Ordered	Sorted	Thread safe	Duplicate	Nullable
HashMap	Map	N	N	N	N (key)	One `null` (key)
HashTable	Map	N	N	Y	N (key)	N (key)
LinkedHashMap	Map	Y	N	N	N (key)	One `null` (key)
TreeMap	Map	Y	Y	N	N (key)	N (key)
ConcurrentHashMap	Map	N	N	Y	N (key)	N
ConcurrentSkipListMap	Map	Y	Y	Y	N (key)	N
ArrayDeque	Deque	Y	N	N	Y	N
PriorityQueue	Queue	Y	N	N	Y	N
ConcurrentLinkedQueue	Queue	Y	N	Y	Y	N
ConcurrentLinkedDeque	Deque	Y	N	Y	Y	N

ArrayList

```
List<Integer> nums = new ArrayList<>();

// Adding
nums.add(2);
nums.add(5);
nums.add(8);

// Retrieving
System.out.println(nums.get(0));

// Indexed for loop iteration
for (int i = 0; i < nums.size(); i++) {
    System.out.println(nums.get(i));
}

nums.remove(nums.size() - 1);
nums.remove(0); // VERY slow
```

```
for (Integer value : nums) {
    System.out.println(value);
}
```

HashMap

```
Map<Integer, String> m = new HashMap<>();
m.put(5, "Five");
m.put(8, "Eight");
m.put(6, "Six");
m.put(4, "Four");
m.put(2, "Two");

// Retrieving
System.out.println(m.get(6));

// Lambda forEach
m.forEach((key, value) -> {
    String msg = key + ": " + value;
    System.out.println(msg);
});
```

HashSet

```
Set<String> set = new HashSet<>();
if (set.isEmpty()) {
    System.out.println("Empty!");
}

set.add("dog");
set.add("cat");
set.add("mouse");
set.add("snake");
set.add("bear");

if (set.contains("cat")) {
    System.out.println("Contains cat");
}

set.remove("cat");
for (String element : set) {
    System.out.println(element);
}
```

ArrayDeque

```
Deque<String> a = new ArrayDeque<>();

// Using add()
a.add("Dog");

// Using addFirst()
a.addFirst("Cat");

// Using addLast()
a.addLast("Horse");

// [Cat, Dog, Horse]
System.out.println(a);

// Access element
System.out.println(a.peek());

// Remove element
System.out.println(a.pop());
```

Misc

Access Modifiers

Modifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
no modifier	Y	Y	N	N
private	Y	N	N	N

Regular expressions

```
String text = "I am learning Java";
// Removing All Whitespace
text.replaceAll("\\s+", "");
```

```
// Splitting a String
text.split("\\|");
text.split(Pattern.quote("|"));
```

See: Regex in java

Comment

```
// I am a single line comment!

/*
And I am a
multi-line comment!
*/

/**
 * This
 * is
 * documentation
 * comment
 */
```

Keywords

abstract	continue	for	new	switch	assert
default	goto	package	synchronized	boolean	do
if	private	this	break	double	implements
protected	throw	byte	else	import	public
throws	case	enum	instanceof	return	transient
catch	extends	int	short	try	char
final	interface	static	void	class	finally
long	strictfp	volatile	const	float	native
super	while				

Math methods

`Math.max(a,b)`	Maximum of a and b

`Math.min(a,b)`	Minimum of a and b
`Math.abs(a)`	Absolute value a
`Math.sqrt(a)`	Square-root of a
`Math.pow(a,b)`	Power of b
`Math.round(a)`	Closest integer
`Math.sin(ang)`	Sine of ang
`Math.cos(ang)`	Cosine of ang
`Math.tan(ang)`	Tangent of ang
`Math.asin(ang)`	Inverse sine of ang
`Math.log(a)`	Natural logarithm of a
`Math.toDegrees(rad)`	Angle rad in degrees
`Math.toRadians(deg)`	Angle deg in radians

Try/Catch/Finally

```
try {
  // something
} catch (Exception e) {
  e.printStackTrace();
} finally {
  System.out.println("always printed");
}
```