

Java Quick Reference

Frequently Used Static Constants/Methods

```
Math.PI
Math.random()
Math.pow(a, b)
System.currentTimeMillis()
System.out.println(anyValue)
JOptionPane.showMessageDialog(null,
    message)
JOptionPane.showInputDialog(
    prompt-message)
Integer.parseInt(string)
Double.parseDouble(string)
Arrays.sort(type[] list)
Arrays.binarySearch(type[] list, type key)
```

Array/Length/Initializer

```
int[] list = new int[10];
list.length;
int[] list = {1, 2, 3, 4};
```

Multidimensional Array/Length/Initializer

```
int[][] list = new int[10][10];
list.length;
list[0].length;
int[][] list = {{1, 2}, {3, 4}};
```

Ragged Array

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

Text File Output

```
PrintWriter output =
    new PrintWriter(filename);
output.print(...);
output.println(...);
output.printf(...);
```

Text File Input

```
Scanner input = new Scanner(
    new File(filename));
```

File Class

```
File file =
    new File(filename);
file.exists()
file.renameTo(File)
file.delete()
```

Object Class

```
Object o = new Object();
o.toString();
o.equals(o1);
```

Comparable Interface

```
c.compareTo(Comparable)
c is a Comparable object
```

String Class

```
String s = "Welcome";
String s = new String(char[]);
int length = s.length();
char ch = s.charAt(index);
int d = s.compareTo(s1);
boolean b = s.equals(s1);
boolean b = s.startsWith(s1);
boolean b = s.endsWith(s1);
String s1 = s.trim();
String s1 = s.toUpperCase();
String s1 = s.toLowerCase();
int index = s.indexOf(ch);
int index = s.lastIndexOf(ch);
String s1 = s.substring(ch);
String s1 = s.substring(i,j);
char[] chs = s.toCharArray();
String s1 = s.replaceAll(regex, repl);
String[] tokens = s.split(regex);
```

ArrayList Class

```
ArrayList<E> list = new ArrayList<E>();
list.add(object);
list.add(index, object);
list.clear();
Object o = list.get(index);
boolean b = list.isEmpty();
boolean b = list.contains(object);
int i = list.size();
list.remove(index);
list.set(index, object);
int i = list.indexOf(object);
int i = list.lastIndexOf(object);
```

printf Method

```
System.out.printf("%b %c %d %f %e %s",
    true, 'A', 45, 45.5, 45.5, "Welcome");
System.out.printf("%-5d %10.2f %10.2e %8s",
    45, 45.5, 45.5, "Welcome");
```

Java Quick Reference

Console Input

```
Scanner input = new Scanner(System.in);
int intValue = input.nextInt();
long longValue = input.nextLong();
double doubleValue = input.nextDouble();
float floatValue = input.nextFloat();
String string = input.next();
```

Console Output

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

GUI Input Dialog

```
String string = JOptionPane.showInputDialog(
    "Enter input");
int intValue = Integer.parseInt(string);
double doubleValue =
    Double.parseDouble(string);
```

Message Dialog

```
JOptionPane.showMessageDialog(null,
    "Enter input");
```

Primitive Data Types

byte	8 bits
short	16 bits
int	32 bits
long	64 bits
float	32 bits
double	64 bits
char	16 bits
boolean	true/false

Arithmetic Operators

+	addition
-	subtraction
*	multiplication
/	division
%	remainder
++var	preincrement
--var	predecrement
var++	postincrement
var--	postdecrement

Assignment Operators

=	assignment
+=	addition assignment
-=	subtraction assignment
*=	multiplication assignment
/=	division assignment
%=	remainder assignment

Relational Operators

<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	equal to
!=	not equal

Logical Operators

&&	short circuit AND
 	short circuit OR
!	NOT
^	exclusive OR

if Statements

```
if (condition) {
    statements;
}

if (condition) {
    statements;
}
else {
    statements;
}

if (condition1) {
    statements;
}
else if (condition2) {
    statements;
}
else {
    statements;
}
```

switch Statements

```
switch (intExpression) {
    case value1:
        statements;
        break;
    ...
    case valueN:
        statements;
        break;
    default:
        statements;
}
```

Loop Statements

```
while (condition) {
    statements;
}

do {
    statements;
} while (condition);

for (init; condition;
    adjustment) {
    statements;
}
```