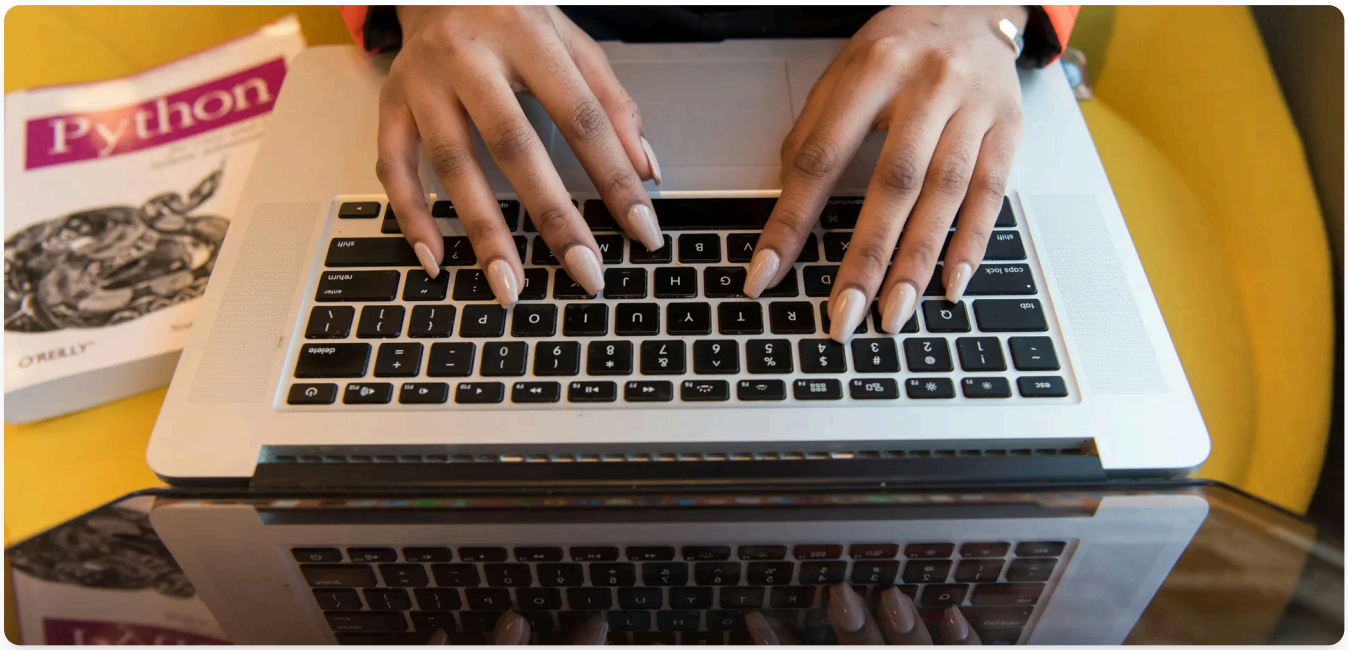

Java Cheatsheet

> *"Java Cheatsheet for java developers"*

By CodeWithHarry Updated: 5 April 2025

Basics

Basic syntax and functions from the Java programming language.

Boilerplate

```
class HelloWorld {
    public static void main(String args[]) {
        System.out.println("Hello World");
    }
}
```

Showing Output

It will print something to the output console.

```
class HelloWorld {
    public static void main(String args[]) {
        System.out.println("Hello World");
    }
}
```

Taking Input

It will take string input from the user.

```
import java.util.Scanner;
class HelloWorld {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        String name = sc.nextLine();
        System.out.println(name);
    }
}
```

It will take integer input from the user.

```
import java.util.Scanner;
class HelloWorld {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int x = sc.nextInt();
        System.out.println(x);
    }
}
```

It will take float input from the user.

```
import java.util.Scanner;
class HelloWorld {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        float x = sc.nextFloat();
```

```
        System.out.println(x);
    }
}
```

It will take double input from the user.

```
import java.util.Scanner;
class HelloWorld {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        double x = sc.nextDouble();
        System.out.println(x);
    }
}
```

Primitive Type Variables

The eight primitives defined in Java are `int`, `byte`, `short`, `long`, `float`, `double`, `boolean`, and `char`. These aren't considered objects and represent raw values.

byte

`byte` is a primitive data type that only takes up 8 bits of memory.

```
class HelloWorld {
    public static void main(String args[]) {
        byte age = 18;
        System.out.println(age);
    }
}
```

long

`long` is another primitive data type related to integers. `long` takes up 64 bits of memory.

```
class HelloWorld {
```

```
    public static void main(String args[]) {
        long var = 900L;
        System.out.println(var);
    }
}
```

float

We represent basic fractional numbers in Java using the `float` type. This is a single-precision decimal number, which means if we get past six decimal points, this number becomes less precise and more of an estimate.

```
class HelloWorld {
    public static void main(String args[]) {
        float price = 100.05f;
        System.out.println(price);
    }
}
```

char

`char` is a 16-bit integer representing a Unicode-encoded character.

```
class HelloWorld {
    public static void main(String args[]) {
        char letter = 'A';
        System.out.println(letter);
    }
}
```

int

`int` holds a wide range of non-fractional number values.

```
class HelloWorld {
    public static void main(String args[]) {
        int var1 = 256;
        System.out.println(var1);
```

```
    }
}
```

short

If we want to save memory and `byte` is too small, we can use `short` .

```
class HelloWorld {
    public static void main(String args[]) {
        short var2 = 5666;
        System.out.println(var2);
    }
}
```

Comments

A comment is the code that is not executed by the compiler, and the programmer uses it to keep track of the code.

Single line comment

```
// It's a single line comment
```

Multi-line comment

```
/* It's a
multi-line
comment
*/
```

Constants

Constants are like a variable, except that their value never changes during program execution.

```
public class Declaration {
    final double PI = 3.14;

    public static void main(String[] args) {
        System.out.println("Value of PI: " + PI);
    }
}
```

Arithmetic Expressions

These are the collection of literals and arithmetic operators.

Addition

It can be used to add two numbers.

```
public class HelloWorld {
    public static void main(String args[]) {
        int x = 10 + 3;
        System.out.println(x);
    }
}
```

Subtraction

It can be used to subtract two numbers.

```
public class HelloWorld {
    public static void main(String args[]) {
        int x = 10 - 3;
        System.out.println(x);
    }
}
```

Multiplication

It can be used to multiply two numbers.

```
public class HelloWorld {
    public static void main(String args[]) {
        int x = 10 * 3;
        System.out.println(x);
    }
}
```

Division

It can be used to divide two numbers.

```
public class HelloWorld {
    public static void main(String args[]) {
        int x = 10 / 3;
        System.out.println(x);
    }
}
```

Modulo Remainder

It returns the remainder of the two numbers after division.

```
public class HelloWorld {
    public static void main(String args[]) {
        int x = 10 % 3;
        System.out.println(x);
    }
}
```

Augmented Operators

Addition assignment

```
public class HelloWorld {
    public static void main(String args[]) {
        int var = 1;
```

```
        var += 10;
        System.out.println(var);
    }
}
```

Subtraction assignment

```
public class HelloWorld {
    public static void main(String args[]) {
        int var = 1;
        var -= 10;
        System.out.println(var);
    }
}
```

Multiplication assignment

```
public class HelloWorld {
    public static void main(String args[]) {
        int var = 1;
        var *= 10;
        System.out.println(var);
    }
}
```

Division assignment

```
public class HelloWorld {
    public static void main(String args[]) {
        int var = 1;
        var /= 10;
        System.out.println(var);
    }
}
```

Modulus assignment

```
public class HelloWorld {
    public static void main(String args[]) {
        int var = 1;
        var %= 10;
        System.out.println(var);
    }
}
```

Escape Sequences

It is a sequence of characters starting with a backslash, and it doesn't represent itself when used inside a string literal.

Tab

It gives a tab space.

```
public class HelloWorld {
    public static void main(String args[]) {
        System.out.print("\t");
    }
}
```

Backslash

It adds a backslash.

```
public class HelloWorld {
    public static void main(String args[]) {
        System.out.print("\\");
    }
}
```

Single quote

It adds a single quotation mark.

```
public class HelloWorld {
    public static void main(String args[]) {
        System.out.print("\'");
    }
}
```

Question mark

It adds a question mark.

```
public class HelloWorld {
    public static void main(String args[]) {
        System.out.print("?");
    }
}
```

Carriage return

Inserts a carriage return in the text at this point.

```
public class HelloWorld {
    public static void main(String args[]) {
        System.out.print("\r");
    }
}
```

Double quote

It adds a double quotation mark.

```
public class HelloWorld {
    public static void main(String args[]) {
        System.out.print("\"");
    }
}
```

Type Casting

Type Casting is a process of converting one data type into another.

Widening Type Casting

It means converting a lower data type into a higher.

```
class HelloWorld {
    public static void main(String args[]) {
        int x = 45;
        double var_name = x;
        System.out.println(var_name);
    }
}
```

Narrowing Type Casting

It means converting a higher data type into a lower.

```
class HelloWorld {
    public static void main(String args[]) {
        double x = 40005;
        int var_name = (int) x;
        System.out.println(var_name);
    }
}
```

Decision Control Statements

Conditional statements are used to perform operations based on some condition.

if Statement

```
if (condition) {
    // block of code to be executed if the condition is true
```

```
}
```

if-else Statement

```
if (condition) {
    // If condition is True then this block will get executed
} else {
    // If condition is False then this block will get executed
}
```

if else-if Statement

```
if (condition1) {
    // Codes
} else if(condition2) {
    // Codes
} else if (condition3) {
    // Codes
} else {
    // Codes
}
```

Ternary Operator

It is shorthand for an if-else statement.

Syntax

```
variable = (condition) ? expressionTrue : expressionFalse;
```

Example

```
public class TernaryOperatorExample {
    public static void main(String args[]) {
        int x, y;
        x = 20;
```

```
        y = (x == 1) ? 61 : 90;
        System.out.println("Value of y is: " +  y);
        y = (x == 20) ? 61 : 90;
        System.out.println("Value of y is: " + y);
    }
}
```

Switch Statements

It allows a variable to be tested for equality against a list of values (cases).

```
class SwitchExample {
    public static void main(String args[]) {
        int day = 4;
        switch (day) {
            case 1:
                System.out.println("Monday");
                break;
            case 2:
                System.out.println("Tuesday");
                break;
            case 3:
                System.out.println("Wednesday");
                break;
            case 4:
                System.out.println("Thursday");
                break;
            case 5:
                System.out.println("Friday");
                break;
            case 6:
                System.out.println("Saturday");
                break;
            case 7:
                System.out.println("Sunday");
                break;
        }
    }
}
```

Iterative Statements

Iterative statements facilitate programmers to execute any block of code lines repeatedly and can be controlled as per conditions added by the coder.

while Loop

It iterates the block of code as long as a specified condition is True.

```
public class WhileExample {
    public static void main(String[] args) {
        int i = 1;
        while(i ≤ 10) {
            System.out.println(i);
            i++;
        }
    }
}
```

for Loop

`for` loop is used to run a block of code several times.

```
class HelloWorld {
    public static void main(String args[]) {
        for(int i = 1; i < 100; i++) {
            System.out.println(i);
        }
    }
}
```

for-each Loop

```
public class HelloWorld {
    public static void main(String args[]) {
        int[] arr = {2, 4, 5, 7, 8, 0, 3, 5};
        for (int i : arr) {
            System.out.println(i);
```

```
        }
    }
}
```

do-while Loop

It is an exit-controlled loop. It is very similar to the while loop with one difference, i.e., the body of the do-while loop is executed at least once even if the condition is False.

```
public class HelloWorld {
    public static void main(String args[]) {
        int i = 1;
        do {
            System.out.println(i);
            i++;
        } while(i ≤ 100);
    }
}
```

Break statement

`break` keyword inside the loop is used to terminate the loop.

```
class HelloWorld {
    public static void main(String args[]) {
        for(int i = 1; i < 100; i++) {
            System.out.println(i);
            if(i == 50)
                break;
        }
    }
}
```

Continue statement

`continue` keyword skips the rest of the current iteration of the loop and returns to the starting point of the loop.

```
class HelloWorld {
    public static void main(String args[]) {
        for(int i = 1; i < 100; i++) {
            System.out.println(i);
            if(i == 50)
                continue;
        }
    }
}
```

Arrays

Arrays are used to store multiple values in a single variable.

Declaring an array

Declaration of an array.

```
public class HelloWorld {
    public static void main(String args[]) {
        String[] var_name;
    }
}
```

Defining an array

Defining an array.

```
public class HelloWorld {
    public static void main(String args[]) {
        String[] var_name = {"harry", "rohan", "aakash"};
    }
}
```

Accessing an array

Accessing the elements of an array.

```
public class HelloWorld {
    public static void main(String args[]) {
        String[] var_name = {"Harry", "Rohan", "Aakash"};
        System.out.println(var_name[0]);
    }
}
```

Changing an element

Changing any element in an array.

```
public class HelloWorld {
    public static void main(String args[]) {
        String[] var_name = {"Harry", "Rohan", "Aakash"};
        var_name[2] = "Shubham";
    }
}
```

Array length

It gives the length of the array.

```
public class HelloWorld {
    public static void main(String args[]) {
        String[] var_name = {"Harry", "Rohan", "Aakash"};
        System.out.println(var_name.length);
    }
}
```

Loop through an array

It allows us to iterate through each array element.

```
public class HelloWorld {
    public static void main(String args[]) {
        String[] var_name = {"Harry", "Rohan", "Aakash"};
        for (int i = 0; i < var_name.length; i++) {
            System.out.println(var_name[i]);
```

```
            }
        }
    }
```

Multi-dimensional Arrays

Arrays can be 1-D, 2-D, or multi-dimensional.

```
// Creating a 2x3 array (two rows, three columns)
int[][] matrix = new int[2][3];
matrix[0][0] = 10;
// Shortcut
int[][] matrix = {
    { 1, 2, 3 },
    { 4, 5, 6 }
};
```

Methods

Methods are used to divide an extensive program into smaller pieces. It can be called multiple times to provide reusability to the program.

Declaration

Declaration of a method.

```
returnType methodName(parameters) {
    // statements
}
```

Calling a method

Calling a method.

```
methodName(arguments);
```

Example

```
public static void findEvenOdd(int num) {
    // method body
    if(num % 2 == 0)
        System.out.println(num + " is even");
    else
        System.out.println(num + " is odd");
}

import java.util.Scanner;
public class EvenOdd {
    public static void main (String args[]) {
        // creating Scanner class object
        Scanner scan = new Scanner(System.in);
        System.out.print("Enter the number: ");
        // reading value from the user
        int num = scan.nextInt();
        // method calling
        findEvenOdd(num);
    }
}
```

Method Overloading

Method overloading means having multiple methods with the same name, but different parameters.

```
class Calculate {
    void sum(int x, int y) {
        System.out.println("Sum is: " + (x + y));
    }
    void sum(float x, float y) {
        System.out.println("Sum is: " + (x + y));
    }
    public static void main(String[] args) {
        Calculate calc = new Calculate();
        calc.sum(5, 4); // sum(int x, int y) is called.
        calc.sum(1.2f, 5.6f); // sum(float x, float y) is called.
    }
```

```
}
```

Recursion

Recursion is when a function calls a copy of itself to work on a minor problem. The function that calls itself is known as the Recursive function.

```
void recurse() {
    recurse();
}
```

Strings

It is a collection of characters surrounded by double quotes.

Creating String Variable

```
String var_name = "Hello World";
```

String Length

Returns the length of the string.

```
public class str {
    public static void main(String args[]) {
        String var_name = "Harry";
        System.out.println("The length of the string is: " +
var_name.length());
    }
}
```

String Methods toUpperCase()

Convert the string into uppercase.

```
public class str {
    public static void main(String args[]) {
        String var_name = "Harry";
        System.out.println(var_name.toUpperCase());
    }
}
```

toLowerCase()

Convert the string into lowercase.

```
public class str {
    public static void main(String args[]) {
        String var_name = "Harry";
        System.out.println(var_name.toLowerCase());
    }
}
```

indexOf()

Returns the index of a specified character from the string.

```
public class str {
    public static void main(String args[]) {
        String var_name = "Harry";
        System.out.println(var_name.indexOf("a"));
    }
}
```

concat()

Used to concatenate two strings.

```
public class str {
    public static void main(String args[]) {
        String var1 = "Harry";
        String var2 = "Bhai";
        System.out.println(var1.concat(var2));
```

```
    }
}
```

Math Class

Math class allows you to perform mathematical operations.

Methods max() method

It is used to find the greater number among the two.

```
public class Demo {
    public static void main(String[] args) {
        // using the max() method of Math class
        System.out.print("The maximum number is
```

Tags

java cheatsheet

Share

Main

Home
Contact
Work With Us
My Gear

Learn

Courses
Tutorials
Notes

Legal

Terms
Privacy
Refund

Social

GitHub
Twitter (X)
YouTube
Facebook

Made with ❤️ and ☕ in India