Java Programming Cheat Sheet

Basic Syntax

Hello World


java

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

Comments


java

```java
// Single-line comment
/* Multi-line comment */
/** Documentation comment */
```

Data Types

Primitive Types


java

```java
byte b = 127;          // 8-bit, -128 to 127
short s = 32767;       // 16-bit, -32,768 to 32,767
int i = 2147483647;    // 32-bit, -2^31 to 2^31-1
long l = 9223372036854775807L; // 64-bit
float f = 3.14f;       // 32-bit floating point
double d = 3.14159;    // 64-bit floating point
boolean bool = true;   // true or false
char c = 'A';          // 16-bit Unicode character
```

Reference Types


java

```
String str = "Hello";
Integer num = 10;       // Wrapper class
int[] arr = {1, 2, 3};  // Array
```

Variables


java

```
int x = 10;                 // Declaration and initialization
final int CONSTANT = 100;     // Constant (cannot be changed)
var name = "Java";            // Type inference (Java 10+)
```

Operators

Arithmetic


java

```
+ - * / %                   // Add, subtract, multiply, divide, modulo
++ --                       // Increment, decrement
```

Comparison


java

```
== != < > <= >=             // Equal, not equal, less, greater
```

Logical


java

```
&& || !                    // AND, OR, NOT
```

Assignment

java

```
= += -= *= /= %=           // Assignment operators
```

Control Flow

If-Else

java

```
if (condition) {
    // code
} else if (anotherCondition) {
    // code
} else {
    // code
}
```

Ternary Operator

java

```
int result = (a > b) ? a : b;
```

Switch

java

```java
switch (variable) {
    case value1:
        // code
        break;
    case value2:
        // code
        break;
    default:
        // code
}
```

Enhanced Switch (Java 12+)


java

```java
String result = switch (day) {
    case "MON", "TUE" -> "Weekday";
    case "SAT", "SUN" -> "Weekend";
    default -> "Invalid";
};
```

Loops

For Loop


java

```java
for (int i = 0; i < 10; i++) {
    // code
}
```

Enhanced For Loop


java

```java
for (Type item : collection) {
    // code
}
```

While Loop


java

```java
while (condition) {
    // code
}
```

Do-While Loop


java

```java
do {
    // code
} while (condition);
```

Arrays


java

```java
int[] arr = new int[5];              // Declaration
int[] arr2 = {1, 2, 3, 4, 5};       // Initialization
arr[0] = 10;                        // Access
int length = arr.length;            // Length

// Multi-dimensional
int[][] matrix = new int[3][3];
int[][] matrix2 = {{1,2}, {3,4}};
```

Strings


java

```java
String str = "Hello";
int len = str.length();              // Length
char ch = str.charAt(0);             // Character at index
String sub = str.substring(1, 4);    // Substring
String upper = str.toUpperCase();    // Convert to uppercase
String lower = str.toLowerCase();    // Convert to lowercase
boolean eq = str.equals("Hello");    // Compare strings
String concat = str + " World";      // Concatenation
String[] parts = str.split(" ");     // Split string
```

Classes and Objects

Class Definition

java

```java
public class Person {
    // Fields
    private String name;
    private int age;

    // Constructor
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    // Getter
    public String getName() {
        return name;
    }

    // Setter
    public void setName(String name) {
        this.name = name;
    }

    // Method
    public void greet() {
        System.out.println("Hello, I'm " + name);
    }
}
```

Creating Objects


java

```java
Person person = new Person("John", 25);
person.greet();
String name = person.getName();
```

Inheritance


java

```java
public class Animal {
    public void eat() {
        System.out.println("Eating...");
    }
}

public class Dog extends Animal {
    @Override
    public void eat() {
        System.out.println("Dog eating...");
    }

    public void bark() {
        System.out.println("Woof!");
    }
}
```

Interfaces

java

```java
public interface Drawable {
    void draw();  // Abstract method

    default void display() {  // Default method
        System.out.println("Displaying...");
    }
}

public class Circle implements Drawable {
    @Override
    public void draw() {
        System.out.println("Drawing circle");
    }
}
```

Abstract Classes

java

```java
public abstract class Shape {
    abstract void draw();  // Abstract method

    public void show() {   // Concrete method
        System.out.println("Showing shape");
    }
}
```

Collections

ArrayList

java

```java
ArrayList<String> list = new ArrayList<>();
list.add("Item");           // Add element
list.get(0);                // Get element
list.remove(0);             // Remove element
list.size();                // Size
list.contains("Item");      // Check if contains
```

HashMap

java

```java
HashMap<String, Integer> map = new HashMap<>();
map.put("key", 10);          // Add entry
int value = map.get("key");  // Get value
map.remove("key");           // Remove entry
map.containsKey("key");      // Check if key exists
```

HashSet

java

```java
HashSet<String> set = new HashSet<>();
set.add("Item");              // Add element
set.remove("Item");           // Remove element
set.contains("Item");         // Check if contains
```

Exception Handling


java

```java
try {
    // Code that may throw exception
    int result = 10 / 0;
} catch (ArithmeticException e) {
    // Handle specific exception
    System.out.println("Error: " + e.getMessage());
} catch (Exception e) {
    // Handle general exception
    e.printStackTrace();
} finally {
    // Always executes
    System.out.println("Cleanup");
}
```

Throwing Exceptions


java

```java
public void checkAge(int age) throws IllegalArgumentException {
    if (age < 0) {
        throw new IllegalArgumentException("Age cannot be negative");
    }
}
```

File I/O

Reading File


java

```java
import java.io.*;
import java.util.Scanner;

// Using Scanner
Scanner scanner = new Scanner(new File("file.txt"));
while (scanner.hasNextLine()) {
    String line = scanner.nextLine();
}
scanner.close();

// Using BufferedReader
BufferedReader reader = new BufferedReader(new FileReader("file.txt"));
String line;
while ((line = reader.readLine()) != null) {
    System.out.println(line);
}
reader.close();
```

Writing File


java

```java
BufferedWriter writer = new BufferedWriter(new FileWriter("file.txt"));
writer.write("Hello World");
writer.newLine();
writer.close();
```

Lambda Expressions (Java 8+)


java

```java
// Syntax: (parameters) -> expression
List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5);

// forEach
numbers.forEach(n -> System.out.println(n));

// Filter and map
numbers.stream()
    .filter(n -> n % 2 == 0)
    .map(n -> n * 2)
    .forEach(System.out::println);
```

Streams (Java 8+)


java

```java
List<String> names = Arrays.asList("Alice", "Bob", "Charlie");

// Filter
names.stream()
    .filter(name -> name.startsWith("A"))
    .collect(Collectors.toList());

// Map
names.stream()
    .map(String::toUpperCase)
    .collect(Collectors.toList());

// Reduce
int sum = numbers.stream()
    .reduce(0, (a, b) -> a + b);
```

Common Methods

Object Methods


java

```java
obj.equals(other)        // Check equality
obj.toString()           // String representation
obj.hashCode()           // Hash code
```

Math


java

```java
Math.abs(-5)             // Absolute value: 5
Math.max(10, 20)         // Maximum: 20
Math.min(10, 20)         // Minimum: 10
Math.pow(2, 3)           // Power: 8.0
Math.sqrt(16)            // Square root: 4.0
Math.random()            // Random [0.0, 1.0)
Math.round(3.7)          // Round: 4
```

Access Modifiers


java

```java
public    // Accessible everywhere
private   // Accessible only within class
protected // Accessible within package and subclasses
default   // Accessible within package (no modifier)
```

Keywords


java

```
static    // Belongs to class, not instance
final     // Cannot be changed/overridden
abstract  // Must be implemented by subclass
this      // Reference to current object
super     // Reference to parent class
new       // Create new object
return    // Return value from method
void      // Method returns nothing
null      // No value/reference
instanceof // Check object type
```

Quick Tips

- Class names: PascalCase (e.g., `MyClass`)
- Methods/variables: camelCase (e.g., `myMethod`)
- Constants: UPPER_SNAKE_CASE (e.g., `MAX_VALUE`)
- One public class per file
- File name must match public class name
- Java is case-sensitive
- Statements end with semicolon (;)
- Blocks use curly braces { }