```
// ---- Avery's Java Cheatsheet ------------------------------------

// ---- Template code ---------------------------------------------

import java.util.*; // Import a bunch of useful stuff

class Main { // Should match file name (e.g. Main.java)
  public static void main(String[] args) {
    // Your code goes here
  }
}

// ---- Variables -------------------------------------------------

int myVar1; // Declaration
myVar1 = 4; // Assignment

int myVar2 = 5; // Declaration + assignment
var myVar3 = 6; // Type inferrence

final int x = 1; // Final variables and fields cannot be reassigned
x = 2; // Error

// ---- Primitives ------------------------------------------------

int i = 10;      // Whole number
double d = 3.14;  // Decimal number
boolean b = true; // true or false
char c = 'x';     // Single character

// Compare primitives with `==` and `!=`
i == 20 // false
i !=  0 // true

// Use casts to convert between primitive types
(double) 2 // 2.0
(int) 2.8  // 2   (truncates value)
(int) 'a'  // 97  (unicode value)
(char) 98  // 'b'

// ---- Math ------------------------------------------------------

int n = 10;
n + 4        // 14
n - 4        // 6
n * 4        // 40
n / 4        // 2
n / 4.0      // 2.4
n % 7        // 3
Math.pow(n, 4) // 10000.0
n > 4        // true
n <= 4       // false

int x = 0;
x++;    // x = 1
x += 4; // x = 5
x--;    // x = 4
x *= 2; // x = 8

// ---- Strings ---------------------------------------------------

String s = "Hi Java";
s.equals("Hi Java") // true
s == "Hi Java"      // Unreliable, use equals()
s.split(" ")        // String[2] { "Hi", "Java" }
s.length()          // 7
s.charAt(5)         // 'v'
s + " :)"           // "Hi Java :)"

// See also: endsWith, indexOf, repeat, startsWith, strip,
// toCharArray, toLowerCase, toUpperCase

// ---- Defaults --------------------------------------------------

int defaultInt;       // defaultInt = 0
defaultInt + 4        // 4
String defaultString;  // defaultString = null
defaultString.length() // Error

// ---- Output ----------------------------------------------------

String name = "Avery";
System.out.println("Hello " + name); // Prints "Hello Avery"
```

```
// ---- Input -----------------------------------------------------

Scanner scan = new Scanner(System.in); // New Scanner object
System.out.print("Enter age: ");       // (optional prompt)
String name = scan.nextLine();         // Get String input
System.out.print("Enter age: ");       // (optional prompt)
int age = scan.nextInt();              // Get int input

// ---- Random ----------------------------------------------------

Random rand = new Random(); // New Random generator object
rand.nextInt(11)            // Random integer 0 through 10
rand.nextDouble()           // Random double 0.0 through 1.0
rand.nextBoolean()          // Random boolean

// ---- File IO ---------------------------------------------------

import java.nio.file.*; // Add this import to the top of your file

// You'll need to add a `throws Exception` annotation to your
// method or use try/catch to avoid an `unreported exception` error

// Read lines as a List of Strings
List<String> lines = Files.readAllLines(Path.of("something.txt"));

// Read whole file as a String
String text = Files.readString(Path.of("something.txt"));

// ---- If --------------------------------------------------------

if (number > 0) {
  System.out.println("positive");
} else if (number < 0) {
  System.out.println("negative");
} else {
  System.out.println("zero");
}

// ---- Ternary ---------------------------------------------------

int number = 7;
// Choose between two values based on a condition
String parity = number % 2 == 0 ? "even" : "odd"; // parity = "odd"

// ---- While -----------------------------------------------------

while (n > 10) {
  // Loop while condition is true
}

// See also: do-while loops

// ---- For -------------------------------------------------------

for (int i = 1; i <= 10; i++) {
  // Loop over a range of numbers
}

for (String name : names) { // `names` can be a List or Array
  // Loop through a sequence of values
}

for (String name : names) {
  if (name.endsWith("y")) {
    System.out.println(name); // Print the first name ending in 'y'
    break; // Use `break` to end a `for` or `while` loop early
  }
}

// ---- Scope -----------------------------------------------------

int a = 0;
int b = 0;

if (true) {
  a = 1;     // Update a in outer scope
  int b = 1; // New variable b in inner scope
  int c = 1; // New variable c in inner scope
}

System.out.println(a); // Prints 1
System.out.println(b); // Prints 0 (b was only updated inside if)
System.out.println(c); // Error    (c is only defined inside if)
```

```java
// ---- Arrays ------------------------------------------------

String[] dirs = { "north", "south", "east", "west" };

System.out.println(dirs); // Prints nonsense

// Prints { "north", "south", "east", "west" }
System.out.println(Arrays.toString(dirs));

dirs.length                    // 4
dirs[0]                        // "north"
dirs[dirs.length - 1]          // "west"
dirs[random.nextInt(dirs.length)] // Random direction
String.join("->", dirs)        // "north->south->east->west"

dirs[0] = "up"; // dirs = { "up", "south", "east", "west" }

// Arrays have a fixed length, no equivalent of Python's `append`

// ---- Lists -------------------------------------------------

List<String> dirs = List.of("north", "south", "east", "west");

dirs.size()                            // 4
dirs.get(0)                            // "north" (can't use [0])
dirs.get(dirs.size() - 1)              // "west"
dirs.get(random.nextInt(dirs.size())) // Random direction
String.join("->", dirs)                // "north->south->east->west"

// List.of makes an immutable list that we can't update or add to
dirs.set(0, "up"); // `UnsupportedOperationException`

// To make a mutable List, use `new ArrayList<>(...)`
List<String> mutDirs = new ArrayList<>(dirs);

mutDirs.set(0, "up"); // mutDirs = [up, south, east, west]
mutDirs.add("down");  // mutDirs = [up, south, east, west, down]

// See also: addAll, contains, equals, getLast, indexOf, isEmpty,
// lastIndexOf, removeLast, reversed, sort, subList

// We can't declare a list of primitives, we have to use wrapper
// classes like Integer, Double, Boolean, and Character

List<char> letters = List.of('a', 'b', 'c');      // Doesn't work
List<Character> letters = List.of('a', 'b', 'c'); // Works

// ---- Sets --------------------------------------------------

Set<Character> letters = new HashSet<>();

letters.add('a');    // letters = [a]
letters.add('b');    // letters = [a, b]
letters.size()       // 4
letters.contains('a') // true
letters.remove('b');  // letters = [b]

// See also: containsAll, equals, isEmpty, removeAll, retainAll

// ---- Maps --------------------------------------------------

Map<String, Integer> bank = new HashMap<>();

bank.put("Lamar", 100);    // bank = {Lamar=100}
bank.put("Zay", 50);       // bank = {Zay=50, Lamar=100}
bank.put("Lamar", 200);    // bank = {Zay=50, Lamar=200}
bank.get("Lamar")          // 200
bank.containsKey("Patrick") // false

// See also: compute, containsValue, entrySet, equals,
// getOrDefault, isEmpty, keySet, putAll, putIfAbsent, remove,
// size, values

// ---- Enums -------------------------------------------------

// Use enums to represent sets of options, like compass directions
enum Direction {
  NORTH, SOUTH, EAST, WEST
}

Direction heading = Direction.NORTH;
// Compare Enums with `==` and `!=`
heading != Direction.WEST // true

// ---- Switch ------------------------------------------------

// Use `switch` to check a value against a set of options
switch (fileExtension) {
  case "mp3":
    System.out.println("It's autio");
    break; // Need `break`, `return`, or `yield` at end of case
  case "jpg":
  case "png":
  case "webp": // Multiple cases grouped together
    System.out.println("It's an image");
    break;
  default: // Default runs if no other case matches
    System.out.println("File type unknown");
}

String arrow = switch (heading) {
  case NORTH: yield "↑"; // Get a value from `switch` with `yield`
  case SOUTH: yield "↓";
  case EAST:  yield "→";
  case WEST:  yield "←";
}; // Need semicolon when assigning `switch` result to a variable

// ---- Methods -----------------------------------------------

static void greet(String name) { // No return value
  System.out.println("Hello " + name);
}

static int add(int a, int b) { // Returns an int
  return a + b;
}

static double abs(double n) { // Returns a double
  if (n < 0) {
    return -n; // Return ends the method immediately
  }

  return n;
}

// ---- Overloads ---------------------------------------------

// A method can have multiple variants (overloads) that take
// different arguments

static int maximum(int a, int b) {
  // ...
}

static int maximum(int[] numbers) {
  // ...
}

// ---- Varargs -----------------------------------------------

// Use `...` to allow a flexible number of arguments (varargs)
static void helloAll(String... names) {
  // names is a String[]
  System.out.println("Hello " + String.join(" and ", names));
}

helloAll("Alice");         // Prints "Hello Alice"
helloAll("Alice", "Bob"); // Prints "Hello Alice and Bob"

String[] names = { "Alice", "Bob" };
// You can also pass an Array as varargs
helloAll(people); // Prints "Hello Alice and Bob"
```