



**Java Cheatsheet**

Java is a high-level, object-oriented programming language.



**Basics**

- public class ClassName { }: Declares a class.

- public static void main(String[] args) { }: Main method, the entry point of a Java application.

- System.out.println("text");: Prints text to the console.

**Variables**

- int num = -0;: Declares an integer variable.

- double price = 9.99;: Declares a double variable.

- char letter = 'A';: Declares a character variable.

- String text = "Hello";: Declares a string variable.

- boolean flag = true;: Declares a boolean variable.

**Operators**

- +: Addition operator.

- -: Subtraction operator.

- \*: Multiplication operator.

- /: Division operator.

- %: Modulus operator.

- ++: Increment operator.

- --: Decrement operator.

- ==: Equality operator.

- !=: Not equal operator.

- >: Greater than operator.

- <: Less than operator.

- >=: Greater than or equal to operator.

- <=: Less than or equal to operator.

- &&: Logical AND operator.

- ||: Logical OR operator.

- !: Logical NOT operator.

**Control Flow**

- if (condition) { }: If statement.

- else { }: Else statement.

- else if (condition) { }: Else-if statement.

- switch (variable) { case value: break; }: Switch statement.

- for (int i = 0; i < -0; i++) { }: For loop.

- while (condition) { }: While loop.

- do { } while (condition);: Do-while loop.

- break;: Exits a loop or switch statement.

- continue;: Skips the current iteration of a loop.

**Methods**

- returnType methodName(parameters) { }: Declares a method.

- void methodName() { }: Declares a method that does not return a value.

- int methodName() { return value; }: Declares a method that returns an integer.

**Arrays**

- int[] numbers = new int[-0];: Declares an array of integers.

- String[] words = {"Hello", "World"};: Declares and initializes an array of strings.

- arrayName[index]: Accesses an array element.

**Object-Oriented Programming**

- class ClassName { }: Defines a class.

- ClassName obj = new ClassName();: Creates an object of a class.

- public ClassName() { }: Constructor.

- public void methodName() { }: Method.

- public int fieldName;: Field.

- this.fieldName: Refers to the current object's field.

- super.methodName(): Calls the superclass method.

**Inheritance**

- class SubClass extends SuperClass { }: Inheritance.

- @Override: Annotation to override a method.

**Interfaces**

- interface InterfaceName { }: Defines an interface.

- class ClassName implements InterfaceName { }: Implements an interface.

**Exception Handling**

- try { } catch (ExceptionType e) { }: Try-catch block.

- finally { }: Finally block.

- throw new ExceptionType("message");: Throws an exception.

- throws ExceptionType: Declares that a method throws an exception.

**Collections**

- ArrayList<Type> list = new ArrayList<>();: Creates an ArrayList.

- list.add(value);: Adds a value to the list.

- list.get(index);: Retrieves a value from the list.

- HashMap<KeyType, ValueType> map = new HashMap<>();: Creates a HashMap.

- map.put(key, value);: Puts a key-value pair into the map.

- map.get(key);: Retrieves a value from the map.