Java Complete Foundation

This document is designed as a comprehensive clarification knowledge base for AI-driven Java doubt resolution.

1. Introduction to Java

Java is a high-level, object-oriented programming language developed by Sun Microsystems in 1995 and now maintained by Oracle.

Key Characteristics:

• Platform Independent (Write Once, Run Anywhere)

• Object-Oriented

• Secure and Robust

• Multithreaded

• Automatic Memory Management

Java Program Execution Flow:

1. Source Code (.java)
2. Compiled by javac → Bytecode (.class)
3. Executed by JVM

JDK: Development Kit

JRE: Runtime Environment

JVM: Executes bytecode

AI Tutor – 4C Questions (Introduction)

Critical Thinking: Why does Java use a virtual machine instead of compiling directly to machine code?

Creativity: Modify a Hello World program to print three personalized messages.

Communication: Explain JDK, JRE, and JVM using a real-world analogy.

Collaboration: One student draws execution flow, another explains it.

2. Variables and Data Types

Java is strongly typed. Every variable must declare its data type before usage.

Syntax:

datatype variableName = value;

Example:

```
int age = 25;
double salary = 50000.75;
char grade = 'A';
boolean isActive = true;
```

Variable Types:

- Local Variables
- Instance Variables
- Static Variables

Scope determines where a variable can be accessed.

AI Tutor – 4C Questions (Variables)

Critical Thinking: What error occurs if a variable is used without initialization?

Creativity: Create a student profile program using 5 variables.

Communication: Explain variable scope with an example.

Collaboration: One writes variables, another predicts output.

3. Primitive Data Types

Java has 8 primitive types:

byte (1 byte) – small integers
short (2 bytes) – medium integers
int (4 bytes) – most common integer
long (8 bytes) – very large integers (requires L suffix)

float (4 bytes) – decimal (requires f suffix)
double (8 bytes) – high precision decimal

char (2 bytes) – single character (Unicode supported)
boolean – true or false

Example:

```
int number = 10;
long population = 8000000000L;
float price = 19.99f;
double pi = 3.14159;
char grade = 'A';
boolean passed = true;
```

Memory Allocation:
Primitive types store actual values directly in memory.

Type Casting in Detail

Widening (Implicit Casting):

Smaller type automatically converts to larger type.

Example:

```
int x = 10;
double y = x;
```

Narrowing (Explicit Casting):

Larger type converted to smaller type manually.

Example:

```
double value = 9.7;
int result = (int) value;
```

Note: Narrowing may cause data loss.

AI Tutor – 4C Questions (Primitive Types)

Critical Thinking: Why is double preferred over float in financial calculations?

Creativity: Design a temperature converter using type casting.

Communication: Explain narrowing casting with a real-world example.

Collaboration: One writes casting code, another checks edge cases.

4. Non-Primitive Data Types

Non-primitive types store references to memory locations.

Examples:

- String
- Arrays

• Classes

• Interfaces

• Objects

String Example:

String name = "Java";

Arrays Example:

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

Difference:

Primitive → Direct value storage

Non-Primitive → Memory reference storage

AI Tutor – 4C Questions (Non-Primitive Types)

Critical Thinking: Why are Strings immutable in Java?

Creativity: Create an array of marks and calculate average.

Communication: Explain reference vs value using an analogy.

Collaboration: One declares array, another writes loop to print.

5. Operators in Java

Arithmetic Operators: + - * / %

Relational Operators: == != > < >= <=

Logical Operators: && || !

Assignment Operators: = += -= *= /= %=

Unary Operators: ++ --

Ternary Operator: condition ? value1 : value2

Example:

```
int a = 10;
int b = 5;
System.out.println(a + b);
```

Common Mistake:

Using = instead of == inside conditions.

AI Tutor – 4C Questions (Operators)

Critical Thinking: What logical error occurs if operator precedence is misunderstood?

Creativity: Build a calculator supporting all arithmetic operators.

Communication: Explain && vs || using traffic rules example.

Collaboration: One writes condition, another tests multiple inputs.

6. Control Flow Statements

if, if-else, nested if, switch statements control decision making.

Example:

```
if(age >= 18){
   System.out.println("Adult");
}else{
   System.out.println("Minor");
}
```

Switch Example:

```
switch(day){
   case 1: System.out.println("Monday"); break;
   default: System.out.println("Invalid");
```

```
}
```

AI Tutor – 4C Questions (Control Flow)

Critical Thinking: Why is break necessary in switch statements?
Creativity: Create grading system using if-else.
Communication: Compare switch and if-else logically.
Collaboration: Convert switch to if-else.

7. Loops

Loops repeat a block of code.

for Loop:

```
for(int i=0; i<5; i++){
   System.out.println(i);
}
```

while Loop:

```
int i=0;
while(i<5){
   System.out.println(i);
   i++;
}
```

do-while Loop executes at least once.

break stops loop execution.
continue skips current iteration.

AI Tutor – 4C Questions (Loops)

Critical Thinking: What causes infinite loops?

Creativity: Generate multiplication table using loop.

Communication: Explain difference between while and do-while.

Collaboration: One writes loop, another inserts break logic.

8. Object-Oriented Programming Concepts

Class – Blueprint

Object – Instance of class

Constructor – Initializes object

Encapsulation – Data hiding using private variables

Inheritance – One class inherits another

Polymorphism – Same method, different behavior

Abstraction – Hiding internal implementation

Example:

```
class Student {
   String name;
   int age;
}
```

AI Tutor – 4C Questions (OOP)

Critical Thinking: Why is encapsulation important for security?

Creativity: Design a BankAccount class.

Communication: Explain inheritance using family example.

Collaboration: One writes class, another creates object.

9. Exception Handling

Exceptions are runtime errors.

Example:

```
try{
   int a = 10/0;
}catch(Exception e){
   System.out.println("Error occurred");
}finally{
   System.out.println("Program ended");
}
```

Types:

- Checked Exceptions
- Unchecked Exceptions

AI Tutor – 4C Questions (Exceptions)

Critical Thinking: Why is exception handling necessary in production systems?

Creativity: Handle user input errors.

Communication: Explain difference between checked and unchecked exceptions.

Collaboration: One creates exception, another handles it.