Java Quick Reference Guide

for AP Computer Science Principles Students

Beginner-Friendly Programming Reference

Contents

1 Awesome Video Resources

Tip

Watch these videos BEFORE diving into coding. They'll give you the foundation you need to understand Java concepts!

- **Java in 100 Seconds**: https://www.youtube.com/watch?v=l9AzO1FMgM8
- **Coding with John**: https://www.youtube.com/watch?v=drQK8ciCAjY

2 Java Fundamentals

2.1 Basic Program Structure

Every Java program follows this template:

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

Listing 1: Hello World Program

Important Note

Key Points:

- Every Java program must be inside a `class`
- The `main` method is the entry point
- Class name must match filename (HelloWorld.java)
- Java is case-sensitive

2.2 Data Types & Variables

2.2.1 Primitive Data Types

```java
// Integers
int age = 25;
long population = 7800000000L;

// Floating point
float price = 19.99f;
double salary = 75000.50;

// Other types
boolean isStudent = true;
char grade = 'A';
```

Listing 2: Primitive Data Types

2.2.2 Strings (Non-primitive)

```
String name = "John";
String message = "Hello, " + name + "!";
```

Listing 3: Working with Strings

Common Mistake

Use `.equals()` for string comparison, not `==`

```
// Wrong
if (name == "John") { ... }

// Correct
if (name.equals("John")) { ... }
```

2.3 Control Flow

2.3.1 Conditional Statements

```
int score = 85;

if (score >= 90) {
    System.out.println("Grade: A");
} else if (score >= 80) {
    System.out.println("Grade: B");
} else if (score >= 70) {
    System.out.println("Grade: C");
} else {
    System.out.println("Grade: F");
}
```

Listing 4: If-Else Statements

2.3.2 Loops

```
// For loop
for (int i = 0; i < 10; i++) {
    System.out.println("Count: " + i);
}

// While loop
int count = 0;
while (count < 5) {
    System.out.println("Number: " + count);
    count++;
}

// Do-while loop
int num = 1;
do {
    System.out.println("Number: " + num);
    num++;
} while (num <= 3);
```

Listing 5: Different Types of Loops

2.4 Arrays

2.4.1 Declaration and Initialization

```java
// Method 1: Declare then initialize
int[] numbers = new int[5];
numbers[0] = 10;
numbers[1] = 20;

// Method 2: Initialize with values
int[] scores = {85, 92, 78, 96, 88};
String[] names = {"Alice", "Bob", "Charlie"};
```

Listing 6: Working with Arrays

2.4.2 Working with Arrays

```java
// Get length
int length = scores.length;

// Loop through array
for (int i = 0; i < scores.length; i++) {
    System.out.println("Score " + i + ": " + scores[i]);
}

// Enhanced for loop (for-each)
for (int score : scores) {
    System.out.println("Score: " + score);
}
```

Listing 7: Array Operations

2.5 Methods

2.5.1 Method Declaration

```java
public class Calculator {
    // Method with return value
    public static int add(int a, int b) {
        return a + b;
    }

    // Method without return value (void)
    public static void printMessage(String message) {
        System.out.println(message);
    }

    // Method with multiple parameters
    public static double calculateArea(double length, double width) {
        return length * width;
    }
}
```

Listing 8: Method Examples

Tip

Method Components:

- `public` - Access modifier (can be accessed from anywhere)
- `static` - Belongs to class, not object instance
- `int` - Return type (void if no return value)
- `add` - Method name
- `(int a, int b)` - Parameters

2.6 Classes and Objects

2.6.1 Basic Class Structure

```java
public class Student {
    // Instance variables (fields)
    private String name;
    private int age;
    private double gpa;

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

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

    public int getAge() {
        return age;
    }

    public double getGpa() {
        return gpa;
    }

    // Setter methods
    public void setGpa(double gpa) {
        this.gpa = gpa;
    }

    // Other methods
    public void displayInfo() {
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
        System.out.println("GPA: " + gpa);
    }
}
```

Listing 9: Student Class Example

2.6.2 Creating and Using Objects

```java
public class Main {
    public static void main(String[] args) {
        // Create objects
        Student alice = new Student("Alice", 20, 3.8);
        Student bob = new Student("Bob", 19, 3.6);

        // Use methods
        alice.displayInfo();
        System.out.println("Alice's GPA: " + alice.getGpa());

        // Modify object
        alice.setGpa(3.9);
        System.out.println("Updated GPA: " + alice.getGpa());
    }
}
```

Listing 10: Using Objects

2.7 Input/Output

2.7.1 Reading User Input

```java
import java.util.Scanner;

public class InputExample {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter your name: ");
        String name = scanner.nextLine();

        System.out.print("Enter your age: ");
        int age = scanner.nextInt();

        System.out.print("Enter your GPA: ");
        double gpa = scanner.nextDouble();

        System.out.println("Hello, " + name + "!");
        System.out.println("You are " + age + " years old.");
        System.out.println("Your GPA is " + gpa);

        scanner.close(); // Good practice
    }
}
```

Listing 11: Scanner Input Example

2.8 Common String Methods

```java
String text = "Hello World";

// Length
int length = text.length(); // 11

// Character at index
char firstChar = text.charAt(0); // 'H'

// Substring
String sub = text.substring(0, 5); // "Hello"

// Case conversion
String upper = text.toUpperCase(); // "HELLO WORLD"
String lower = text.toLowerCase(); // "hello world"

// Check if contains
boolean contains = text.contains("World"); // true

// Replace
String replaced = text.replace("World", "Java"); // "Hello Java"

// Split
String[] words = text.split(" "); // ["Hello", "World"]
```

Listing 12: String Method Examples

3 Common Mistakes to Avoid

Common Mistake

Watch out for these common beginner mistakes:

1. **String Comparison**: Use `.equals()`, not `==`
2. **Array Length**: Use `.length` (not `.length()`)
3. **Case Sensitivity**: Java is case-sensitive
4. **Semicolons**: Don't forget semicolons after statements
5. **Curly Braces**: Use {} for code blocks, not indentation
6. **Variable Declaration**: Always declare variable type
7. **Array Indexing**: Arrays start at index 0

Tip

Download the next page if you want a mini reference to the basic structures in java

4 Quick Reference Cheat Sheet

4.1 Variable Declaration

```java
int x = 10;
String name = "John";
boolean flag = true;
int[] numbers = {1, 2, 3, 4, 5};
```

4.2 Control Structures

```java
// If-else
if (condition) {
    // code
} else {
    // code
}

// For loop
for (int i = 0; i < 10; i++) {
    // code
}

// While loop
while (condition) {
    // code
}
```

4.3 Method Declaration

```java
public static returnType methodName(parameters) {
    // method body
    return value; // if not void
}
```

4.4 Class Template

```java
public class ClassName {
    // Fields
    private dataType fieldName;

    // Constructor
    public ClassName(parameters) {
        // initialization
    }

    // Methods
    public returnType methodName(parameters) {
        // method body
    }
}
```

4.5 Why Java for AP Computer Science Principles?

Important Note

Java directly supports AP CSP's five Big Ideas:

- **Creative Development**: Build interactive programs and solve real-world problems
- **Data**: Process, analyze, and visualize data using arrays, collections, and file I/O
- **Algorithms & Programming**: Implement computational solutions with proper syntax and logic
- **Computer Systems & Networks**: Understand how programs execute and interact with systems
- **Impact of Computing**: Create meaningful applications that demonstrate computing's societal impact

Learning Java helps you develop **computational thinking skills** - the core of AP CSP. You'll learn to break down complex problems, recognize patterns, abstract solutions, and design algorithms. These skills are essential for your AP CSP Performance Tasks and the multiple-choice exam.

4.6 Advanced Topics for Continued Learning

After mastering these fundamentals, explore these topics:

1. **Object-Oriented Programming**: Inheritance, Polymorphism, Encapsulation
2. **Exception Handling**: try-catch blocks for robust error handling
3. **Collections Framework**: ArrayList, HashMap, LinkedList for advanced data structures
4. **File I/O**: Reading and writing files for data persistence
5. **Packages and Imports**: Organizing code into reusable modules
6. **GUI Development**: JavaFX or Swing for interactive applications
7. **Web Development**: Spring Boot framework for web applications

4.7 Inspiration for Your End-of-Year Project (30% of Grade)

If you're passionate about Java and want to create an impressive EOY project, consider these library categories and project ideas:

Tip

Data Visualization & Analysis Projects

- **Libraries**: JFreeChart, Processing, JavaFX Charts
- **Project Ideas**: COVID-19 trend analyzer, climate data visualizer, sports statistics dashboard
- **Impact Focus**: Address real-world problems with data-driven insights

Tip

Game Development

- **Libraries**: LibGDX, Java2D, JavaFX
- **Project Ideas**: Educational math game, environmental awareness game, historical simulation
- **Impact Focus**: Use gaming to educate or raise awareness about important issues

Tip

Web Applications

- **Libraries**: Spring Boot, Spark Java, Vaadin
- **Project Ideas**: Community service platform, local business directory, study group organizer
- **Impact Focus**: Connect communities and solve local problems

Common Mistake

EOY Project Success Tips:

- Choose a project that addresses a real problem you care about
- Start simple and add features incrementally
- Focus on demonstrating computational thinking and problem-solving
- Document your development process and reflect on societal impact
- Consider accessibility and user experience in your design