📘Java Programming Notes – AetherCode (Notion Template)

🖥️Cover Page

Course Title: Java Programming – Object-Oriented and Platform-Independent Development\ **Prepared By:** AetherCode Team\ **Website:** www.aethercode.com

> *"Code. Notes. Clarity."*

🕡Table of Contents

Introduction

Java is a high-level, class-based, object-oriented programming language. Created by James Gosling at Sun Microsystems in 1995, it's designed to have as few implementation dependencies as possible. Java applications are typically compiled to bytecode that runs on the Java Virtual Machine (JVM).

💡Features of Java

- Platform Independent (WORA: Write Once, Run Anywhere)
- Object-Oriented
- Robust and Secure
- Automatic Garbage Collection
- Multi-threaded
- Rich API and large standard library

🧰Java Program Structure

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

- `class`: Blueprint
- `main()`: Entry point
- `System.out.println()`: Output

📊Data Types and Variables

Type	Description	Example
int	Integer (4 bytes)	int age = 21;
float	Decimal (4 bytes)	float pi = 3.14f;
double	Large decimal	double e = 2.71828;
char	Single character	char ch = 'A';
boolean	true/false	boolean flag = true;

Operators

- Arithmetic: `+ - * / %`
- Relational: `== != > < >= <=`
- Logical: `&& || !`
- Bitwise: `& | ^ ~ << >>`
- Assignment: `= += -= *= /=`

🕐Control Statements

- if, if-else, switch-case
- Loops: for, while, do-while
- Break, continue, return

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

🧱Classes and Objects

```
class Car {
    String model;
    void start() {
        System.out.println("Car started");
    }
}
```

- Create object: `Car c = new Car();`

🔧Constructors

```
class Student {
    String name;
    Student(String n) {
        name = n;
    }
}
```

- Default, Parameterized, Copy constructors

🧬Inheritance

```
class Animal {
    void sound() { System.out.println("Animal sound"); }
}
class Dog extends Animal {
    void bark() { System.out.println("Bark"); }
}
```

🕖Polymorphism

- Compile-time: Method Overloading
- Run-time: Method Overriding

```
class A {
    void show() { System.out.println("A"); }
}
class B extends A {
```

```java
    void show() { System.out.println("B"); }
}
```

🧰Abstraction & Interfaces

```java
abstract class Shape {
    abstract void draw();
}
interface Printable {
    void print();
}
```

- Abstract class can have concrete methods
- Interfaces support multiple inheritance

🔐Encapsulation

```java
class Account {
    private int balance;
    public void setBalance(int b) { balance = b; }
    public int getBalance() { return balance; }
}
```

🔍Exception Handling

```java
try {
    int x = 10 / 0;
} catch (ArithmeticException e) {
    System.out.println("Error: " + e);
} finally {
    System.out.println("End of try-catch");
}
```

📁File I/O

```java
import java.io.*;
FileWriter fw = new FileWriter("output.txt");
fw.write("Hello File");
fw.close();
```

📦Packages and Access Modifiers

- Access modifiers: `public`, `private`, `protected`, `default`
- Custom packages:

```
package mypackage;
public class MyClass { ... }
```

🕐Multithreading

```
class MyThread extends Thread {
    public void run() {
        System.out.println("Thread running");
    }
}
```

- Runnable interface and Thread class
- Methods: `start()`, `sleep()`, `join()`

🧰Collection Framework

Interface	Implementation
List	ArrayList, LinkedList
Set	HashSet, TreeSet
Map	HashMap, TreeMap

```
ArrayList<String> list = new ArrayList<>();
list.add("Java");
```

🔗Summary

Java enables platform-independent programming with strong OOP support. It is widely used in web, enterprise, mobile, and backend systems.

Next: Explore Python – simple syntax, powerful libraries.