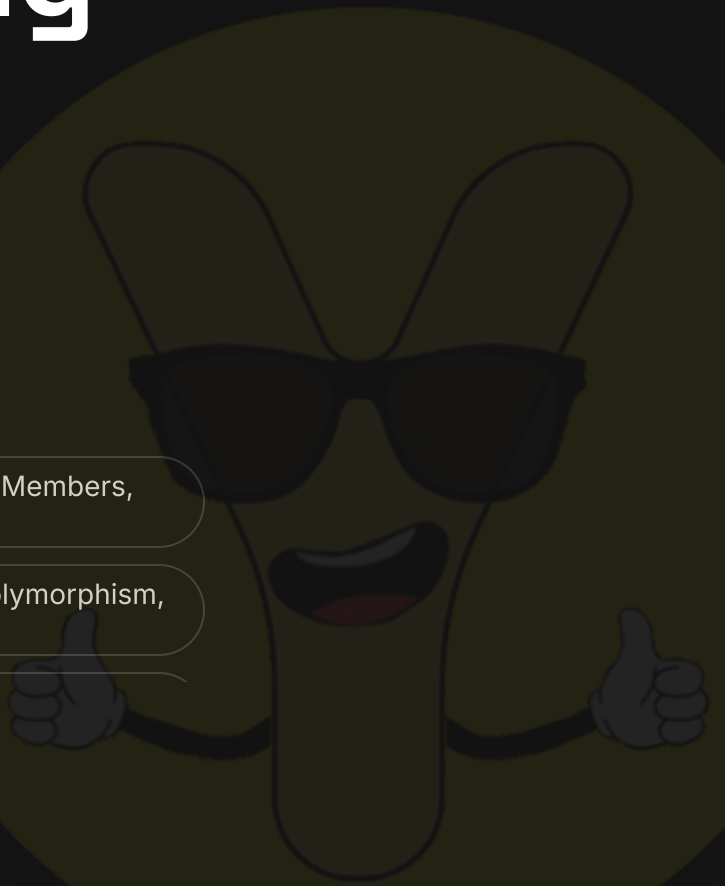
AKTU · 2ND YEAR · SEMESTER 4 · BCS403

Object Oriented Programming with Java

Unit 1 — Introduction, Programming Structures, Object Oriented Programming, and Packages

- Introduction: Why Java, History of Java, JVM, JRE, Java Environment, Java Source File Structure, and Compilation
- Programming Structures in Java: Defining Classes in Java, Constructors, Methods, Access Specifiers, Static Members, Final Members, Comments, Data types, Variables, Operators, Control Flow, Arrays & String
- Object Oriented Programming: Class, Object, Inheritance Super Class, Sub Class, Overriding, Overloading, Encapsulation, Polymorphism, Abstraction, Interfaces, and Abstract Class



DECODED BY

Chirag Jain

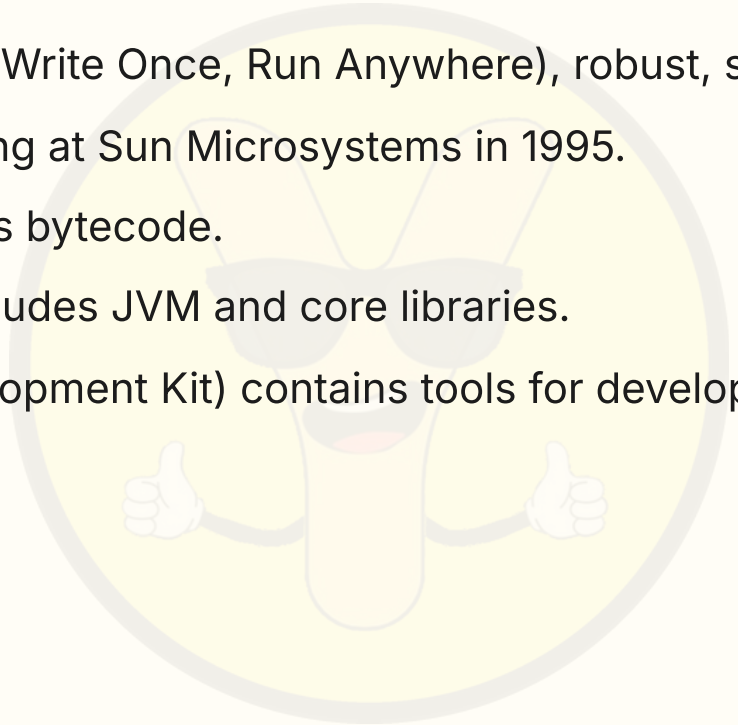
Founder, Ycotes · Gold Medalist in Artificial Intelligence, IIT Madras · taught 10,000+ students

theycotes.com

Introduction to Java

Java is a high-level, class-based, object-oriented programming language designed for portability.

- Why Java: Platform independence (Write Once, Run Anywhere), robust, secure, and multithreaded.
- History: Developed by James Gosling at Sun Microsystems in 1995.
- JVM: Java Virtual Machine executes bytecode.
- JRE: Java Runtime Environment includes JVM and core libraries.
- Java Environment: JDK (Java Development Kit) contains tools for development and JRE.



Java Execution Architecture



The compilation and execution process of a Java program.

Java Source File Structure

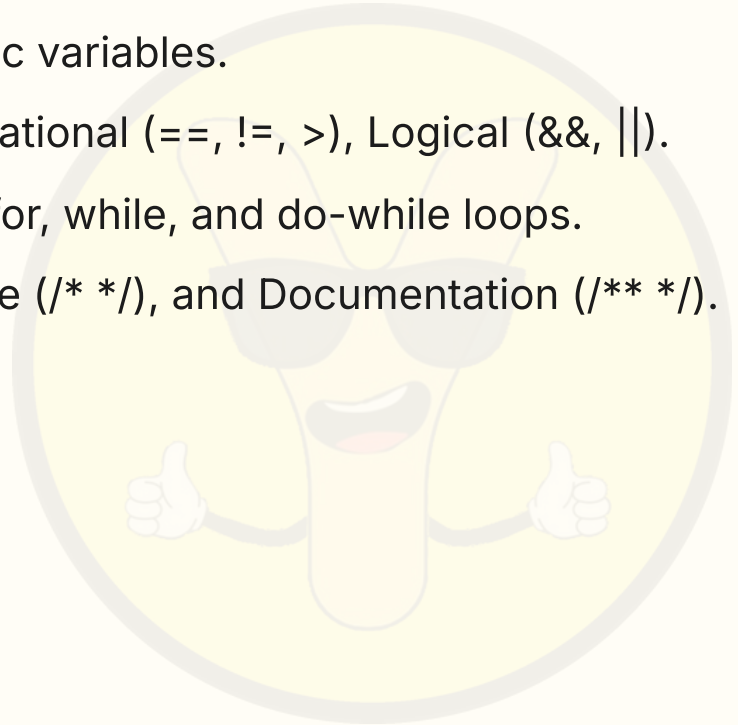
```
package myPackage; // Optional
import java.util.*; // Imports

public class MyClass { // Class definition
    int x; // Variable
    public static void main(String[] args) { // Entry point
        System.out.println("Hello World");
    }
}
```



- Package declaration must be the first line.
- Imports follow package declaration.
- Only one public class per file.
- The file name must match the public class name.

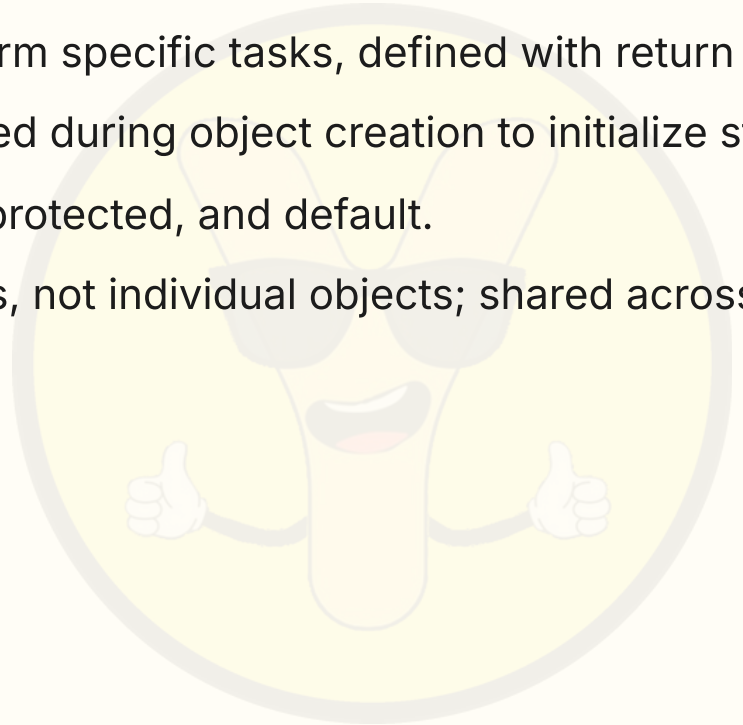
Programming Structures: Basics

- Data Types: Primitive (int, float, char, boolean) and Non-primitive (String, Arrays).
- Variables: Local, Instance, and Static variables.
- Operators: Arithmetic (+, -, *, /), Relational (==, !=, >), Logical (&&, ||).
- Control Flow: if-else, switch-case, for, while, and do-while loops.
- Comments: Single line (//), Multi-line (/* */), and Documentation (/** */).



Classes and Methods

- Class: A blueprint for creating objects, containing data members and methods.
- Methods: Blocks of code that perform specific tasks, defined with return type and parameters.
- Constructors: Special methods called during object creation to initialize state.
- Access Specifiers: public, private, protected, and default.
- Static Members: Belong to the class, not individual objects; shared across all instances.

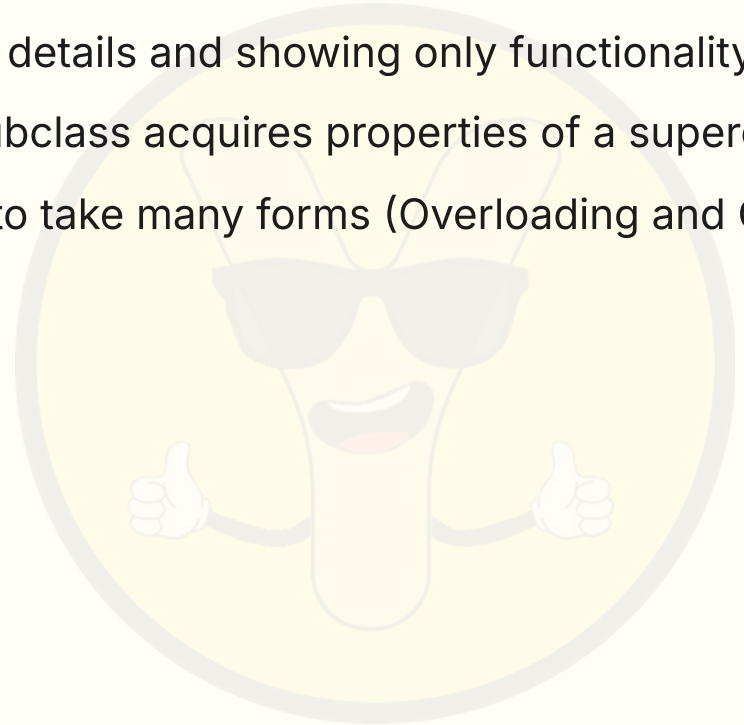


Access Specifiers Comparison

Specifier	Class	Package	Subclass	World
public	Yes	Yes	Yes	Yes
protected	Yes	Yes	Yes	No
default	Yes	Yes	No	No
private	Yes	No	No	No

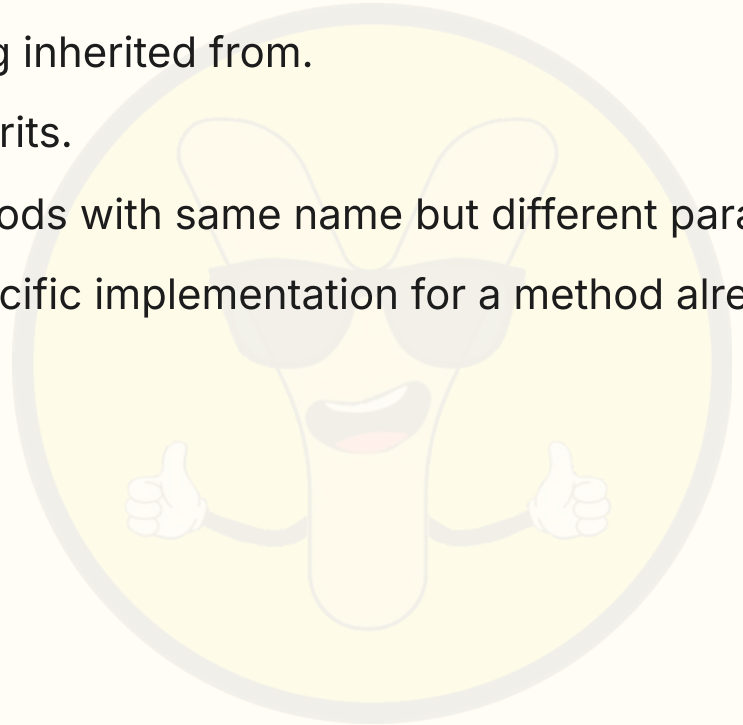
OOP Principles: Core Concepts

- Encapsulation: Wrapping data and methods into a single unit (class) to hide internal state.
- Abstraction: Hiding implementation details and showing only functionality (Abstract classes/Interfaces).
- Inheritance: Mechanism where a subclass acquires properties of a superclass.
- Polymorphism: Ability of an object to take many forms (Overloading and Overriding).



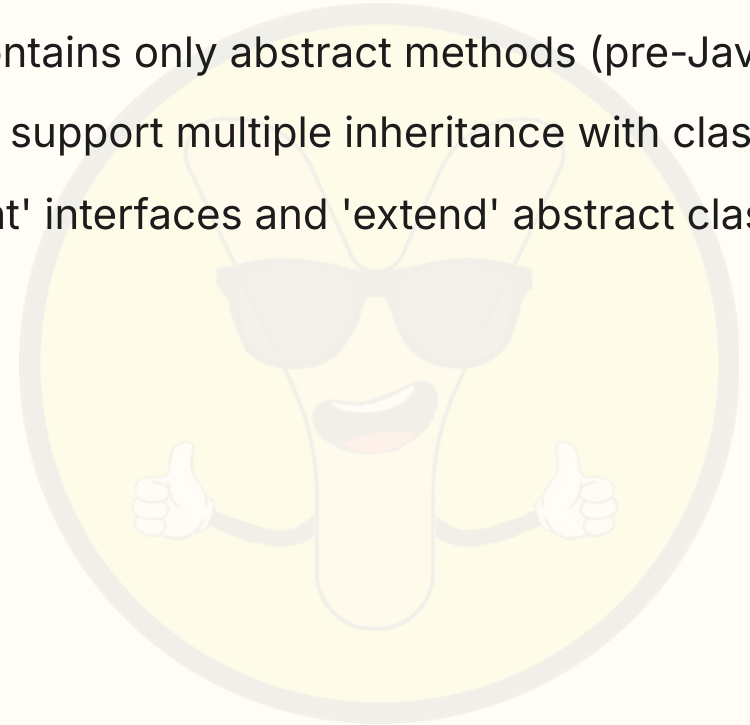
Inheritance and Polymorphism

- Inheritance: Uses 'extends' keyword. Supports single and multilevel inheritance.
- Super Class: The parent class being inherited from.
- Sub Class: The child class that inherits.
- Method Overloading: Multiple methods with same name but different parameters in same class.
- Method Overriding: Providing a specific implementation for a method already defined in superclass.



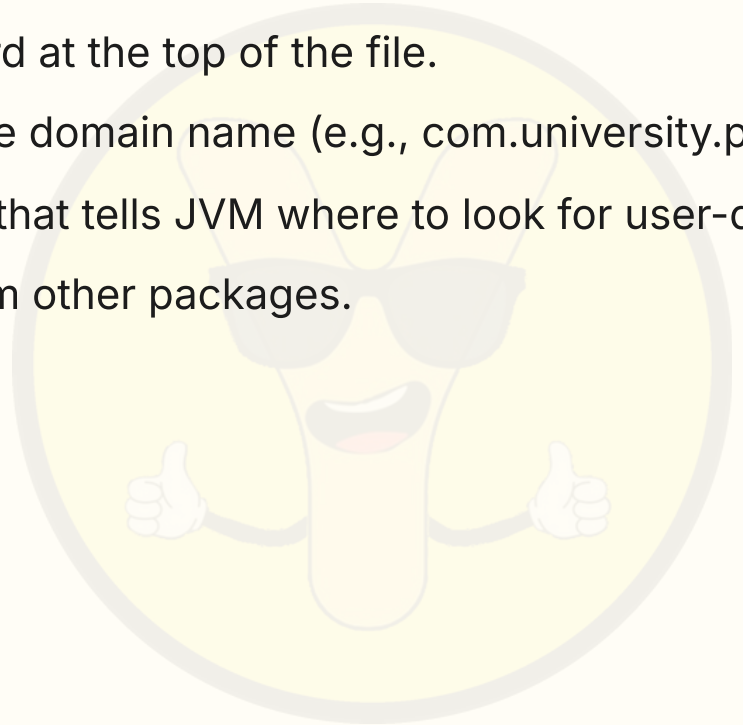
Abstraction: Abstract Class vs Interface

- Abstract Class: Can have abstract and concrete methods. Uses 'abstract' keyword.
- Interface: A blueprint of a class. Contains only abstract methods (pre-Java 8).
- Multiple Inheritance: Java does not support multiple inheritance with classes, but supports it via Interfaces.
- Implementation: Classes 'implement' interfaces and 'extend' abstract classes.



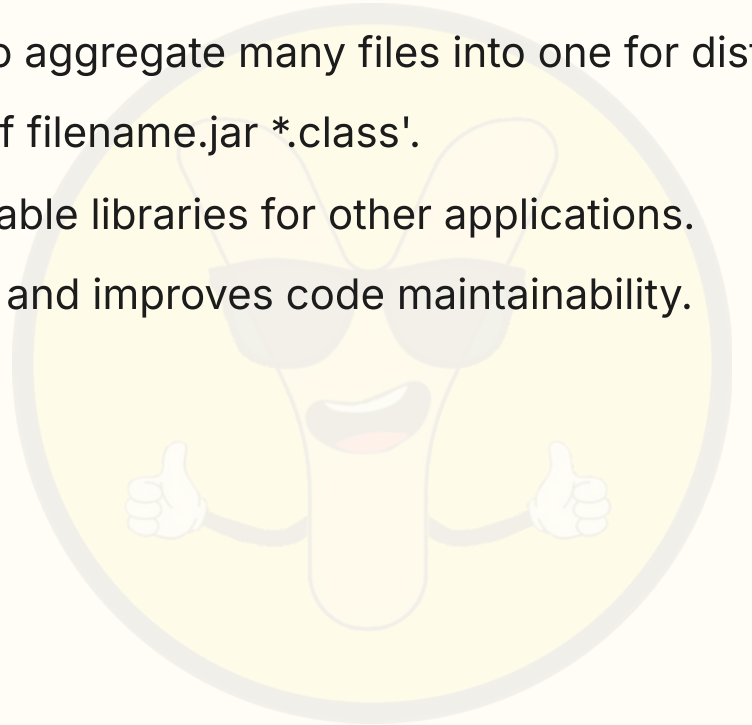
Packages in Java

- Package: A namespace that organizes a set of related classes and interfaces.
- Defining: Use the 'package' keyword at the top of the file.
- Naming Convention: Usually reverse domain name (e.g., com.university.project).
- CLASSPATH: Environment variable that tells JVM where to look for user-defined classes.
- Import: Used to access classes from other packages.



Advanced Package Management

- Static Import: Allows access to static members of a class without class name qualification.
- JAR Files: Java Archive files used to aggregate many files into one for distribution.
- Creating JAR: Use command 'jar cvf filename.jar *.class'.
- Library Packages: JARs act as reusable libraries for other applications.
- Benefits: Prevents naming conflicts and improves code maintainability.



Exam corner

- **Q1** Explain the features of Java that make it 'Platform Independent'.
- **Q2** Differentiate between Method Overloading and Method Overriding with examples.
- **Q3** What is the role of JVM and JRE in Java execution?
- **Q4** Explain the importance of Access Specifiers with a suitable table.
- **Q5** Define Encapsulation and Abstraction. How are they implemented in Java?
- **Q6** What are Packages? Explain the steps to create and use a JAR file.
- **Q7** Differentiate between an Abstract Class and an Interface.

Unit 1 done. You are exam-ready.

Revise the Exam corner once more, then open Unit 2 on theycotes.com. Every AKTU subject, unit by unit, free. Notes by Chirag Jain, Ycotes.

