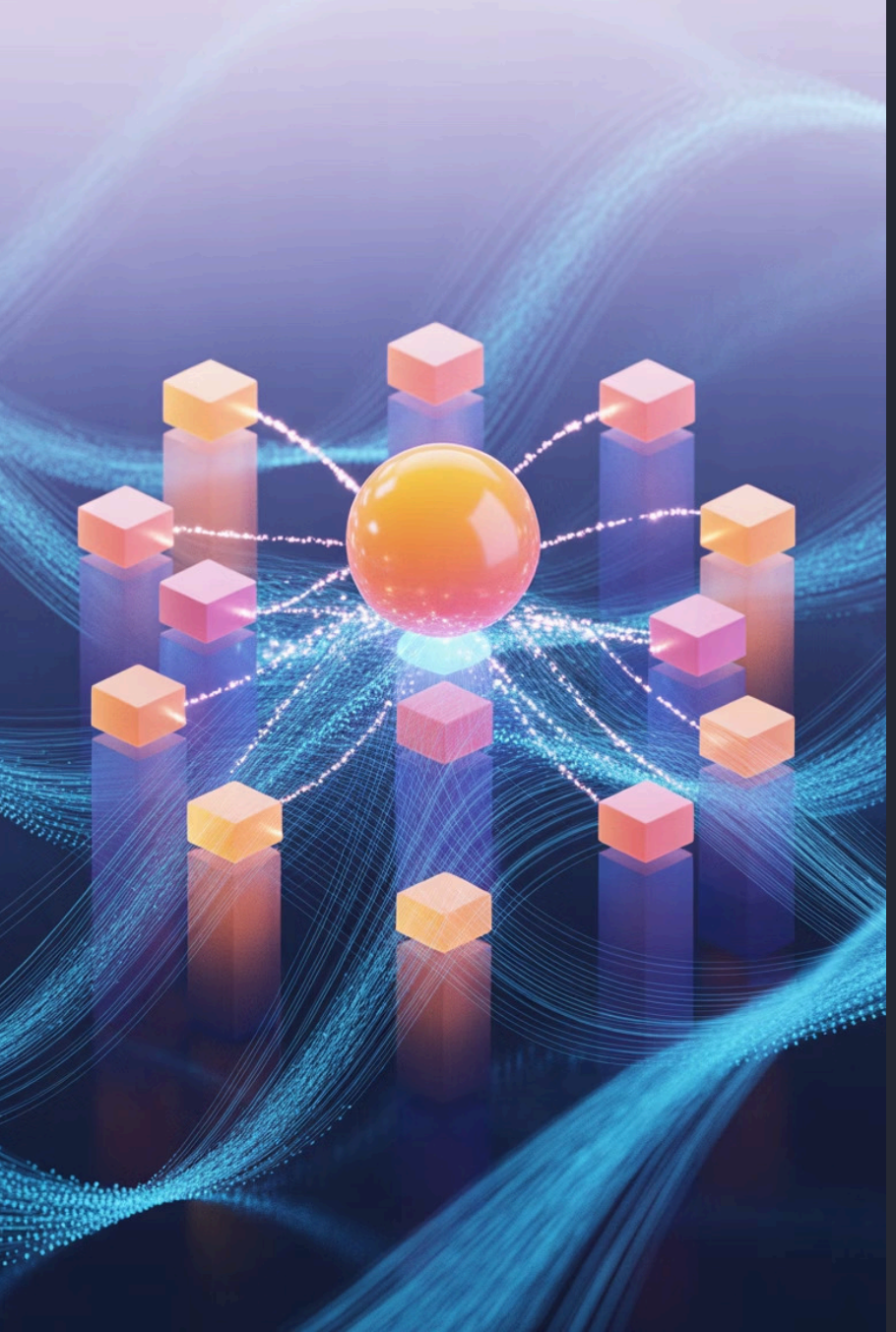
Java Object-Oriented Programming Concepts

Object-oriented programming organizes code around "objects." It's fundamental to modern Java development.

Used by 9.4 million developers worldwide, OOP breaks complex problems into smaller, manageable objects.

N **by Naresh Chaurasia**



What is Object-Oriented Programming?

Based on Objects

A programming paradigm that uses objects containing data and methods.

Problem Solving

Breaks complex problems into smaller, more manageable objects.

Data Focus

Concentrates on data and operations that manipulate the data.

Classes vs Objects

Class

Blueprint or template for creating objects.

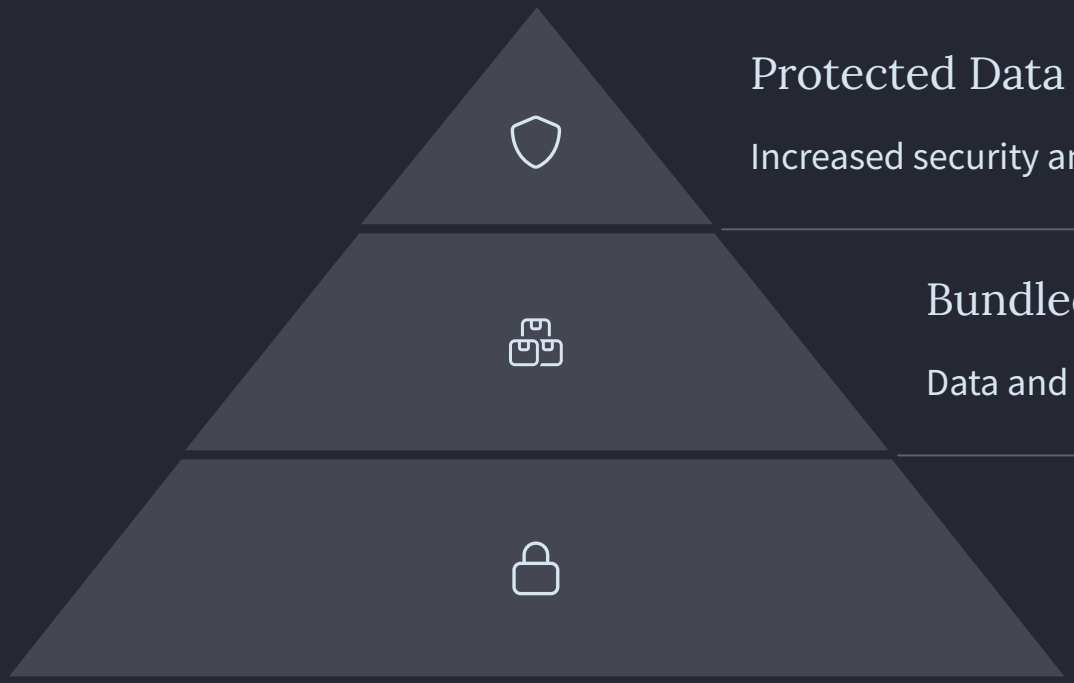
Defines attributes and methods for all objects of that type.

Object

Instance of a class with specific data.

Multiple objects interact through well-defined interfaces.

First Pillar: Encapsulation



Protected Data

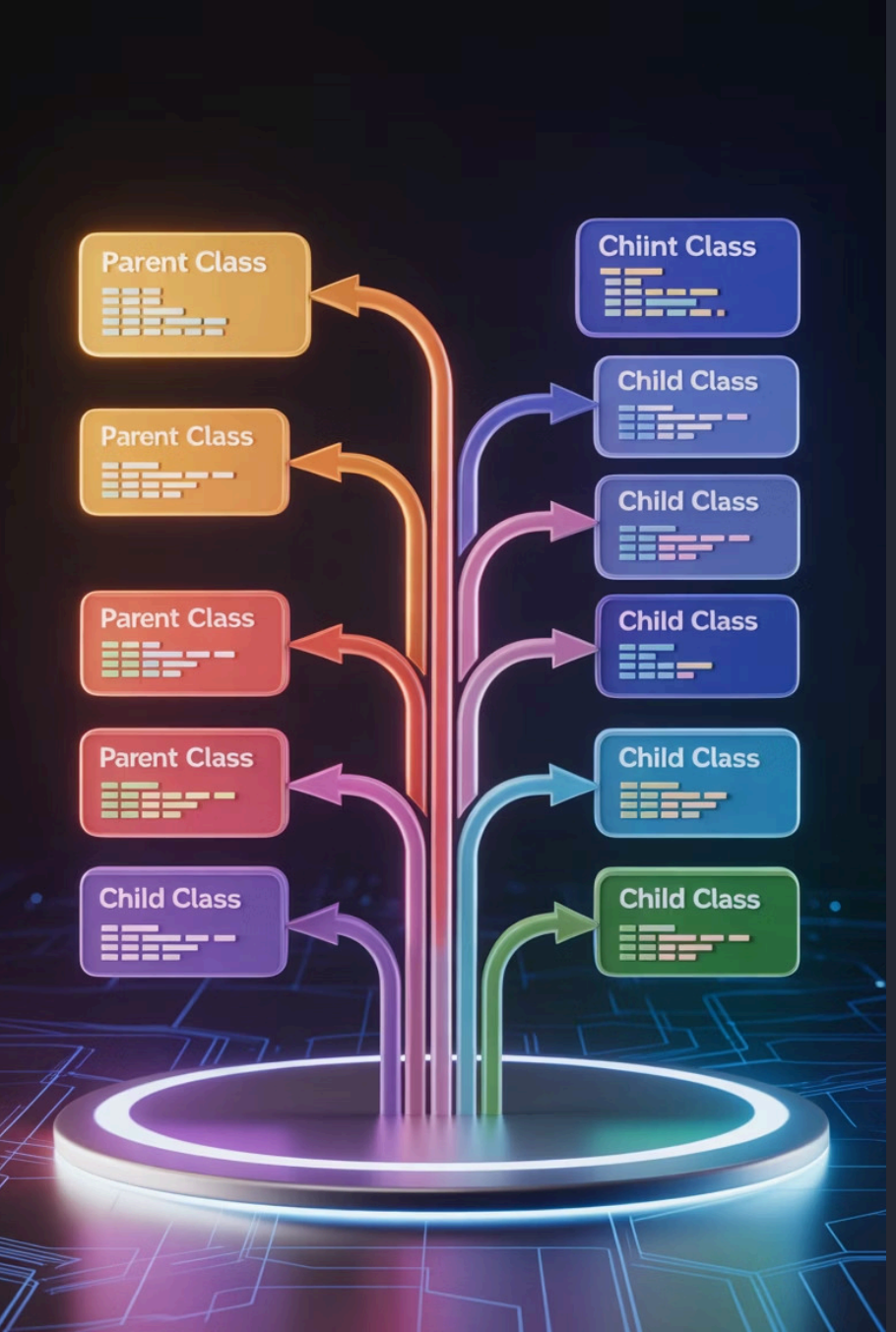
Increased security and controlled access

Bundled Logic

Data and methods packaged together

Information Hiding

Access modifiers control visibility



Second Pillar: Inheritance

Vehicle Class

Base class with common attributes and methods

Car Class

Inherits from Vehicle, adds car-specific features

SportsCar Class

Inherits from Car, adds sports-specific features

Third Pillar: Polymorphism

Different Forms

Objects taking multiple forms based on context

Method Overriding

Runtime polymorphism changing behavior

Method Overloading

Compile-time polymorphism for flexibility

Fourth Pillar: Abstraction

Hide Implementation

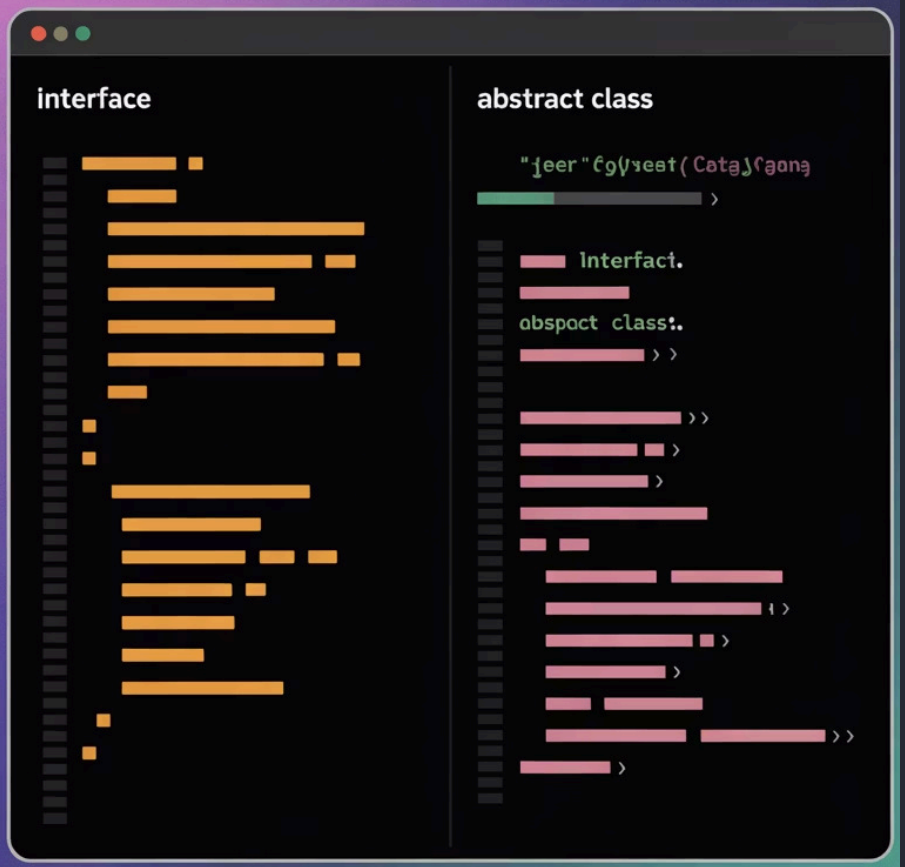
Conceal complex details while exposing only necessary functionality.

Abstract Classes

Provide partial implementation with some abstract methods to be filled in.

Interfaces

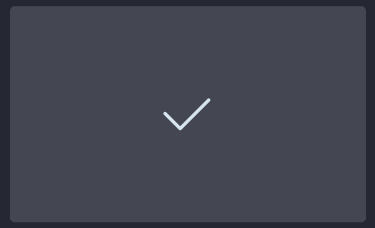
Define contracts that implementing classes must follow.



Interfaces vs Abstract Classes

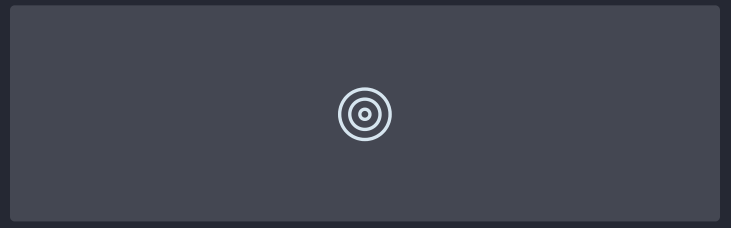
Feature	Interface	Abstract Class
Inheritance	Multiple	Single
Implementation	No implementation	Partial implementation
Fields	Constants only	Any type
Usage	Define behavior contract	Base for related classes

OOP Best Practices



SOLID Principles

Follow these guidelines for robust design



Single Purpose

Keep classes focused on one responsibility

Composition Over Inheritance

Use object composition for flexibility

Benefits of OOP in Java

Modularity

Easier to develop, maintain, and understand complex systems.

Reusability

Inheritance and libraries boost developer productivity.

Security

Access modifiers protect sensitive data from unauthorized access.

Real-world Modeling

Objects mirror real entities for intuitive design.