BluVink

BluVink Verified Notes

UNIT 1: Principles of Object-Oriented Programming (OOP)

AICTE CSE Curriculum Alignment

Prepared by: BluVink Content Team

Syllabus

- Introduction to Object-Oriented Programming and Object-Oriented Thinking
- Need for OOP; comparison with procedural programming
- Basic concepts: Object, Class, Encapsulation, Abstraction, Inheritance, Polymorphism
- Object model and elements of OOP (object, class, inheritance, polymorphism, encapsulation, message passing, dynamic binding, object identity)
- Features, advantages and applications of OOP

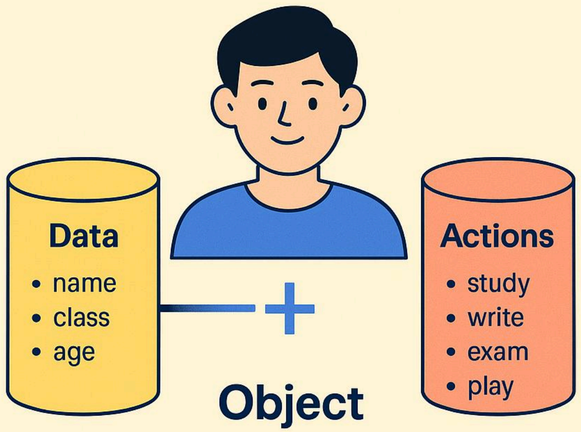
What is OOP?

OOP (Object-Oriented Programming) is a way of building programs using objects.

Object = Data + Actions

Just like real things:

- A Student has data (name, age) and actions (study, write exam).
- A Car has data (color, model) and actions (start, stop).



Why Do We Need OOP?

Before OOP, programmers used **procedural programming**, which had problems:

- Code was long and confusing
- Data could be changed accidentally
- Programs were hard to maintain
- Not suitable for big projects

OOP solves these problems by:

- Breaking code into small, easy parts called objects
- Keeping data safe inside each object
- Allowing reusability of code
- Making programs look like real-world systems

Simple Real-World Example

Think of a classroom:

Thing	What it has	What it does
Student	name, age, roll no	study(), writeExam()
Teacher	name, subject	teach(), evaluate()
Fan	speed	rotate()

All these are objects in real life.

Key OOP Concept: Object

An object is anything that:

- Has **data** (information)
- Can **do actions**

Examples:

- Dog → color, breed + bark(), run()
- Mobile Phone → model, battery + call(), message()

Objects make programs look and feel real.:

Key OOP Concept: Class

A Class is a blueprint or plan used to create objects.

- Has **data** (information)
- Can **do actions**

Real-Life examples:

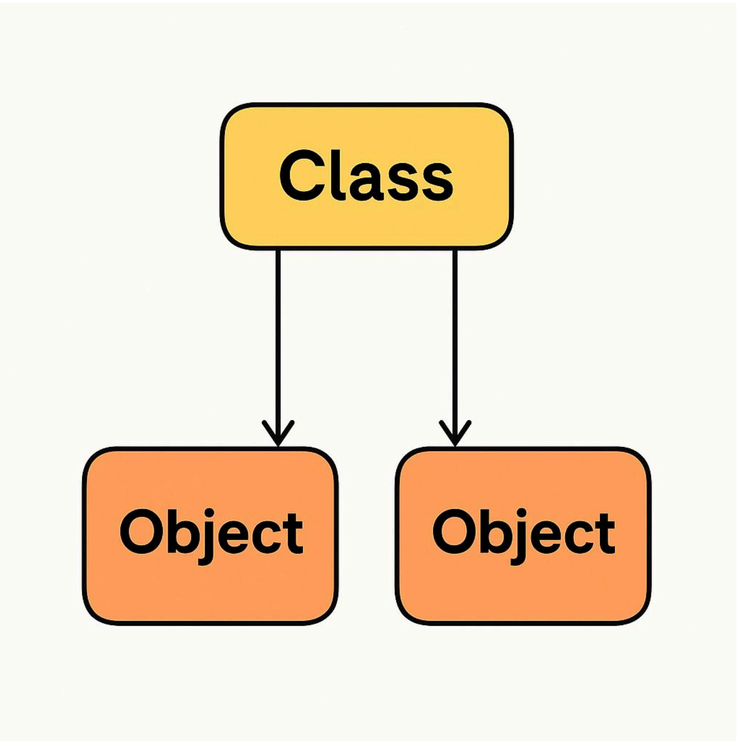
- Recipe → Class
- Cake → Object
- House Plan → Class
- House → Object

A class **defines** what an object will have and do.

Class → Object Flow

Flow:

- Create a **Class** (plan)
- Create **Objects** from it
- Use objects to perform actions



Encapsulation (Data Hiding)

Encapsulation means:

Keeping data safe inside the object and **allowing access only through special methods**.

Real-Life example:

A medicine capsule hides the drug inside.

Same way, a class hides its data.

```
class Student {
private:
  int marks;    // hidden
public:
  void setMarks(int m) { marks = m; }
  int getMarks() { return marks; }
};
```

Abstraction (Show Only What's Needed)

Abstraction means showing only the important details and hiding the rest.

Keeping data safe inside the object and **allowing access only through special methods**.

Real-Life example:

Using a **phone**:

- You see apps
- You see the screen
- You tap and swipe

You don't see circuits, wires, code.

That's abstraction.

Inheritance (Parent → Child)

Inheritance means:

A **child** gets features from a **parent**.

Real-Life Example:

A child **inherits**:

- Eye color
- Hair type
- Height

Programming Meaning:

One class can get features from another.

Polymorphism (One Thing, Many Forms)

Real-Life Example:

The word “**run**” can mean:

- Run fast
- Run the computer
- Run a business

Programming Example (Concept Only):

- draw() for circle
- draw() for square
- draw() for triangle

Message Passing

Objects can talk to each other by sending messages (calling methods).

Real-Life Example:

- You call the teacher → teacher responds
- Teacher calls student → student answers

Programming Example:

- The message “study()” goes to the student object.

Basic Elements of OOP (Object Model)

Objects can talk to each other by sending messages (calling methods).

Major Elements

- Object
- Class
- Inheritance
- Polymorphism
- Abstraction
- Encapsulation

Minor Elements

- Message Passing
- Dynamic Binding (action decided at runtime)
- Object Identity (each object is unique)

OOP vs Procedural Programming

Procedural	What it has
Based on functions	Based on objects
Data is open	Data is protected
Hard to reuse	Easy to reuse
Not real-world friendly	Real-world friendly
Hard to maintain	Easy to maintain

Simple OOP Example

```
class Car {
public:
    string model;
    void start() {
        cout << "Car Started";
    }
};

int main() {
    Car c1;          // object created
    c1.model = "BMW";
    c1.start();      // calling action
}
```

Where OOP Is Used?

Objects can talk to each other by sending messages (calling methods).

- Banking apps
- Games
- Online shopping
- Hospital systems
- School management
- Mobile apps
- Robotics

Everything uses objects like:
User, Product, Account, Doctor, Car.

UNIT 1 Summary (Exam Bullet Points)

Objects can talk to each other by sending messages (calling methods).

- OOP = making programs with objects
- Object = data + actions
- Class = blueprint
- Encapsulation = data hiding
- Abstraction = show only required parts
- Inheritance = parent → child
- Polymorphism = one name, many behaviors
- Message passing = objects communicate
- OOP matches real life
- Used in all modern apps

Expected Exam Questions

1. Define OOP with real-life examples.
2. Explain class and object with a neat diagram.
3. What is encapsulation? Why is it important?
4. Define abstraction with an example.
5. What is inheritance? Give a simple example.
6. Explain polymorphism in simple words.
7. What are the elements of OOP?

All the best