History of Programming & OOP - Cheat Sheet

1. Machine Language

- Definition: Lowest-level language written in binary (0s and 1s). Directly executed by the CPU.
- Example: 10110000 01100001 -> MOV AL, 61h
- Usage: Embedded systems, firmware
- Pros: Fastest execution
- Cons: Not portable, hard to debug

2. Assembly Language

- Definition: Low-level language with mnemonics; translated by assembler.
- Example:

```
MOV AX, 5
ADD AX, 3
```

- Usage: Hardware programming, performance-critical code
- Pros: More readable than machine code
- Cons: Hardware-specific

3. Procedural Programming

- Definition: Top-down approach using functions and procedures.
- Example (Java):

```
public static int add(int a, int b) {
  return a + b;
}
```

- Usage: C, Pascal, system-level programming
- Pros: Code reuse via functions
- Cons: Less modular for large projects

4. Object-Oriented Programming (OOP)

- Definition: Paradigm based on objects (data + behavior).
- Key Concepts: Encapsulation, Inheritance, Polymorphism, Abstraction

History of Programming & OOP - Cheat Sheet

- Java Example:

```
class Car {
  String brand;
  Car(String b) { brand = b; }
  void display() {
    System.out.println(brand);
  }
}
```

- Pros: Modular, scalable, reusable
- Used in: Java, Python, C++

Class vs Object

- Class: Blueprint/template for creating objects. Example: Car
- Object: Instance of a class with real values. Example: new Car("Tesla")

Summary Table

Paradigm	Level	Key Feature
Machine Language	Very Low	Direct hardware exec
Assembly Language	Low	Mnemonics, assembler
Procedural Programming	Medium	Functions, top-down
OOP	High	Objects, modular code

Top Interview Questions & Answers

Q1: What is a class in Java?

A: A class is a blueprint for creating objects that contain fields and methods.

Q2: How is OOP different from procedural programming?

A: OOP organizes code into objects; procedural uses functions and global data.

Q3: Why is machine language not widely used?

History of Programming & OOP - Cheat Sheet

A: It's hard to write/debug and hardware-specific.