Why Object-oriented programming (OOP) ?

Object-oriented programming (OOP) and procedural programming each have their strengths, but OOP is often preferred for several key reasons. Here are the main reasons :

1. Modularity and Reusability

- **Encapsulation**: In OOP, encapsulation allows bundling of data and methods within classes. This modular structure helps in organizing code better, making it more manageable and self-contained.
- **Reusability**: Objects and classes can be reused across projects, saving development time and effort. For example, a User class with methods like authenticate() or logout() can be reused across multiple applications with little modification.
- **Inheritance**: OOP supports inheritance, allowing classes to inherit properties and behaviors from other classes. This reuse reduces redundancy and promotes code reuse in a more natural way than copying functions or procedures in procedural programming.

2. Scalability and Maintainability

- **Easier Maintenance**: With a clear structure, OOP code can be more straightforward to update and expand. Encapsulation means that a change to a class's internal implementation won't affect other parts of the codebase as long as the interface remains the same.
- **Handling Complexity**: Large systems are easier to manage in OOP, as it enables a natural breakdown of a system into smaller, manageable objects. Each object can be worked on independently and collaboratively, which makes OOP better suited for complex and scalable applications.

3. Data and Function Organization

- **Association of Data and Behavior**: OOP organizes data (attributes) and functions (methods) together within objects. This leads to a more intuitive mapping between real-world entities and the code structure.
- **Higher Abstraction**: OOP allows for the creation of high-level abstractions (like "Car" or "Employee") that directly relate to the problem domain, making the code easier to understand and maintain over time.

4. Polymorphism and Flexibility

- **Polymorphism**: OOP enables polymorphism, which allows one interface to be used for different types. For example, a draw() function might work for different shapes (like Circle or Square) without requiring modification. This makes the code more flexible and extensible.
- **Dynamic Method Dispatch**: By using polymorphism and inheritance, OOP allows objects to interact without needing to know their specific types in advance. This flexibility enhances the adaptability of the code.

5. Better Representation of Real-World Scenarios

- **Closer to Real-Life Modelling**: OOP's structure and its features like inheritance and polymorphism make it easy to model real-world entities and their relationships. A Vehicle superclass with subclasses like Car or Truck is a natural way to represent the world within code.

Event-Driven Systems: OOP is more compatible with event-driven programming and GUIs, where objects representing UI elements, users, or actions can handle and respond to events effectively.

Akhilesh Kumar Gupta
(Technical Training specialist)
TechCarrel LLP