
Q2: What do we mean by coding against interface rather than class?
It means writing code that depends on a contract (interface) instead of a specific implementation (class). This makes the code flexible because you can switch implementations without changing the main logic.

Q2 (continued): What do we mean by code against abstraction not concreteness?
It means the same principle but in a broader sense: depend on abstractions (interfaces or abstract classes) which define *what* should be done, not on concrete classes that define *how* it is done. This keeps systems loosely coupled and extendable.

Q3: What is abstraction as a guideline and how can we implement this through what we have studied?
Abstraction as a guideline means hiding unnecessary details and exposing only the essential features. We can implement it by using interfaces (to define contracts), abstract classes (to enforce structure and share behavior), and encapsulation (to hide fields and expose controlled access through properties and methods).