Sequence should be maintained.

1)Abstraction:

* Select key attributes
* Ignore rest
* Reduce complexity.

Using class and interface we implement abstraction

By deciding essential fields and methods.

2)Encapsulation

Wrapping up of data members and member functions together.

Why do we use?

For security: Using access specifier we hide some method and fields

Creating objects we achieve encapsulation

3)Inheritance =is a relationship

* A class can acquire properties of another class
* Reusability
* Java does not allow multiple inheritance not even through interface.

4)Polymorphism

* Different implementation of particular functionality.
* Using Method overriding and method overloading polymorphism is achieved.

Coupling

Coupling refers to the knowledge or information or dependency of another class. It arises when classes are aware of each other. If a class has the details information of another class, there is strong coupling. In Java, we use private, protected, and public modifiers to display the visibility level of a class, method, and field. You can use interfaces for the weaker coupling because there is no concrete implementation.

### Cohesion

Cohesion refers to the level of a component which performs a single well-defined task. A single well-defined task is done by a highly cohesive method. The weakly cohesive method will split the task into separate parts. The java.io package is a highly cohesive package because it has I/O related classes and interface. However, the java.util package is a weakly cohesive package because it has unrelated classes and interfaces.

Association or Containtment types

1. Aggregation=has a relationship

Room has a chair. Room can exist without chair

1. Composition=has a relationship

Room has a wall . Room can not exist without wall