1. Method names should have a verb in them because you're telling an object to do something.
2. High cohesion.
3. Loose coupling.
4. High orthogonality: Change in one class won't affect changes in others.
5. Data abstraction and implementation hiding: You can ask objects to do something without having to know how objects do it.
6. Single Responsibility (***S***OLID): An object or a class must be focussed on doing one type of job.
7. Delegation: Instead of pulling information from a class, ask that class do the work.
8. Extends isn't always viable.
9. DPatterns; Template Method: The superclass should try to do as much as possible. If it requires any information from the subclass, create an abstract method in it that's implemented by the subclass and change the superclass to an abstract superclass.
10. Class names should be nouns.
11. Interfaces should be adjectives.
12. Declare everything Final by default and use extends as less as possible; if you think its overriding or extension can cause problems.
13. Interface Segregation (SOL***I***D): A situation where you have to implement something you cannot implement and the only solution is to throw an exception. E.g. Worker and volunteer example (volunteers don't get paid).
14. You can use static classes inside interfaces to implement default behaviour for its methods.
15. Open For Extension / Closed For Modification (S***O***LID): The class can have new methods added to it without having to change existing class definition.

Compiled as a result of taking the course **Java Fundamentals: Object Oriented Design** on Pluralsight.