# API Design

The API design is very important because API is the first impression that the user will have with your code and it will be the decision maker whether the user uses it or just throws it away.

It does not matter if you are writing a public or internal API, it is important to design a good API because it will reduce development and maintenance cost to the end user. Readable code and good API will take longer to write, but more people will use it and also it will get reused over and over.

Followings are some things to consider while designing an API whether it is an open source project or project at your work.

## An API should be:

1. Simple: It should be easy to use and learn without reading a lot of documentation.
2. Self-explanatory: It should be easy to read and understand.
3. Extendable: End user should be able to add functionality if needed.
4. Closed: API should be well written which would allow its behavior to be modified without altering its source code.
5. Testable: There should be test cases which would show how to use the API to the end user.

## While designing your API, consider the followings:

1. Keep it simple: You cannot please everyone. Smaller API is easy to learn.
2. Be consistent
3. Know your requirements
4. Practice TDD in order to design good and well documented API. Everyone makes mistakes. Discover these through unit test.
5. Create code samples through unit test and show how the API should be used through unit tests.
6. Follow on known design principles
   1. **Single Responsibility Principle** - class should always handle single functionality
   2. **Open Close Principle** - Classes and methods should be Open for extension and Closed for modification.
   3. **Dependency Injection principle** - Class should not ask for dependency instead it should be provided by framework.
   4. **Programming for Interface not implementation** - program for interface for flexible code which can work with new implementation of interface.
   5. **Delegation Principle** - Don’t let your class do everything; let it delegate it to respective class.
7. Name packages, classes and methods meaningfully and consistently.
8. Follow OO principles
9. Have strategy for handling exceptions. Convert exceptions at layer boundaries.
10. Add Javadoc comments for all public class and methods.
11. Don’t try to reinvent known concepts and instead use the consistent concepts such as factory pattern, builder pattern, singleton pattern and static methods.

.**Follow the following rules when writing an API**

1. Do not expose implementation details to the user.
2. When you have doubt or something is not tested thoroughly, leave it out. You can always go back and add it in the next revision. However, once it is public you cannot take it out.
3. Apply encapsulation. Public classes should have no public fields besides constants.
4. Code should be easy to read.
5. Make the API easy to remember by choosing consistent names.
6. Add Javadoc comments to every class, interface, methods and expeptions.
7. Keep the checked exceptions to minimal. Do not force the client to add a lot of exception handling.
8. Design your API for worst conditions.

In summary, API design is an important task and it is not easy. Before making the API public make sure to test it for performance and Javadoc comments are up to date. Don’t forget once API is public it can evolve, but it still needs to support the previous releases. When you update from one version to the next, test the previous version and make sure nothing breaks.