SOLID

Single Responsibility Principle

Each Java class must perform only one function.

Open-Closed Principle

A class should stay closed to alteration, but it should be possible to extend it.

Liskov's Substitution Principle

Derived classes must be 100% interchangeable with their base classes.

Interface Segregation Principle

A client should never be required to implement an interface that it does not use, or to rely on any method that it does not use.

Dependency Inversion Principle

Rather than real implementations (classes), we should rely on abstractions (interfaces and abstract classes)

DRY

"**D**on't **R**epeat **Y**ourself"
Minimize redundancy in processes and logic.

Git

Branch Naming

<category>_<name>
Category:

- **wip**: Works in progress; things that won't be finished soon
- **feat**: Feature addition or expansion
- **bug**: Bug fix
- **doc:** Documentation
- **junk**: Experimentation

Name:
Descriptive but short name for feature being developed or bug being fixed.

Best Practices

- Minimize pushes directly to main by creating feature branches.
- Avoid pushing user-specific cache files by making use of .gitignores.
- Delete branches after use.

FIRST

Fast

Unit tests should be fast otherwise they will slow down your development/deployment time and will take longer time to pass or fail.

Isolated

Never ever write tests which depend on other test cases. No matter how carefully you design them, there will always be possibilities of false alarms.

Repeatable

A repeatable test is one that produces the same results each time you run it.

Self-validating

Tests must be self-validating means – each test must be able to determine that the output is expected or not. It must determine it is failed or pass.

Timely

Practically, You can write unit tests at any time. You can wait up to code is production-ready or you're better off focusing on writing unit tests in a timely fashion.