📘 Chapter 5: Code Quality

✅ 1. Unit Testing

Feature	Description
Definition	Testing individual units or components (usually functions or methods) in isolation.
Goal	Ensure each function performs as expected.
Tools	JUnit (Java), PyTest/Unittest (Python), NUnit (C#), Mocha (JS), etc.
Best Practices	Small, independent, fast, automated tests with clear assertions.

✅ 2. Code Coverage

Metric	Description
Definition	Measures how much of your source code is tested by automated tests.
Types	Line coverage, branch coverage, path coverage, function coverage.
Goal	Higher coverage = more confidence in the test suite.
Ideal Threshold	80%+ is commonly recommended but depends on project.
Tools	JaCoCo, Cobertura (Java), Coverage.py, Istanbul (JS), etc.

✅ 3. Static Code Analysis

Feature	Description
Definition	Analysis of source code without executing it, to find bugs, code smells, and security issues.
Goal	Improve maintainability, readability, and reliability before runtime.
Checks	Syntax errors, unused variables, complexity, duplication, vulnerabilities.
Timing	Happens during build or commit (before execution).

✅ 4. SonarQube

Feature	Description
What is SonarQube?	A popular open-source platform for continuous inspection of code quality.
Metrics Tracked	Code smells, bugs, vulnerabilities, code coverage, duplication, maintainability.
Supports	Multiple languages like Java, C#, JavaScript, Python, etc.
Integrations	Jenkins, GitHub Actions, Azure DevOps, etc.

✅ 5. Linting Tools

Tool	Language	Purpose
ESLint	JavaScript	Identify problematic patterns, enforce code style.
Pylint / Flake8	Python	Static analysis, code quality checks.
Checkstyle / PMD	Java	Code style & complexity checker.
Rubocop	Ruby	Code style and quality enforcement.

📊 Code Quality Summary Table

Topic	Purpose	Tools
Unit Testing	Validate logic of small code units	JUnit, PyTest, Mocha
Code Coverage	Measure how much code is tested	JaCoCo, Istanbul
Static Code Analysis	Identify code issues without execution	SonarQube, Lint tools

1. What is the primary goal of unit testing?
A. Test the entire system
B. Validate the smallest testable parts of code
C. Test deployment pipeline
D. Validate documentation
✅ **Answer:** B

---

2. What does code coverage measure?
A. Test complexity
B. Percentage of code covered by tests
C. Bugs in code
D. Test execution time
✅ **Answer:** B

---

3. Which tool is used for static code analysis in Java?
A. JUnit
B. JaCoCo
C. Checkstyle
D. Jest
✅ **Answer:** C

---

4. What is SonarQube primarily used for?
A. Compiling code
B. Static code analysis and code quality metrics
C. Unit testing
D. Version control
✅ **Answer:** B

---

5. Which of the following is NOT a type of code coverage?
A. Line coverage
B. Path coverage
C. Deployment coverage
D. Branch coverage
✅ **Answer:** C

---

6. ESLint is a linting tool for which language?
A. Python
B. Java
C. JavaScript
D. Ruby
✅ **Answer:** C

---

7. What does a "code smell" indicate in SonarQube?
A. Compilation error
B. Security vulnerability
C. Maintainability issue or bad practice
D. Passed test
✅ **Answer:** C

---

8. Which of these tools reports code coverage in Python?
A. Istanbul
B. PyTest-Cov / Coverage.py
C. JUnit
D. Checkstyle

✅ **Answer:** B

---

9. Static code analysis is performed at what stage?
A. During runtime
B. During code execution
C. Before execution, during build or commit
D. After deployment
✅ **Answer:** C

---

10. Which metric does SonarQube NOT typically report?
A. Bugs
B. Duplicated code
C. UI responsiveness
D. Code smells
✅ **Answer:** C

---

11. What is the purpose of linting tools?
A. Run automated tests
B. Detect code formatting/style issues
C. Compile code
D. Install dependencies
✅ **Answer:** B

---

12. Which of the following is NOT a unit testing tool?
A. PyTest
B. JUnit
C. SonarQube
D. Mocha
✅ **Answer:** C

---

13. Code with 100% coverage means:
A. No bugs exist
B. Every line of code has been executed by tests
C. Code is deployed
D. Security is guaranteed
✅ **Answer:** B

---

14. Which of these is a benefit of static code analysis?
A. Runtime performance improvement
B. Early bug detection before execution
C. Faster database queries
D. UI testing automation
✅ **Answer:** B

---

15. What does JaCoCo stand for?
A. Java Code Compiler
B. Java Continuous Coding
C. Java Code Coverage
D. Java Copy Constructor
✅ **Answer:** C

---

16. Which tool integrates best with GitHub Actions for linting JavaScript code?
A. ESLint
B. JUnit

C. Flake8
D. Maven
✅ **Answer:** A

---

17. What is a good code coverage benchmark?
A. 20%
B. 40%
C. 80%+
D. 100% only
✅ **Answer:** C

---

18. What kind of tool is Pylint?
A. Testing tool
B. Linting/Static analysis tool for Python
C. Build tool
D. Deployment tool
✅ **Answer:** B

---

19. What kind of issues can SonarQube detect?
A. Build errors
B. Network latency
C. Code duplication, vulnerabilities, bugs
D. API request failures
✅ **Answer:** C

---

20. Which of these improves code maintainability?
A. Complex logic
B. Nested if-else everywhere
C. Modular functions and linted code
D. Skipping tests for faster build
✅ **Answer:** C

1. Which of the following helps ensure code changes don't break existing functionality?
A. Linting
B. Code Coverage
C. Unit Testing
D. Code Formatting
✅ **Answer:** C

---

2. What type of testing focuses on testing one function or method in isolation?
A. Integration Testing
B. System Testing
C. Unit Testing
D. UAT
✅ **Answer:** C

---

3. Which is a metric to measure how well tests cover code paths?
A. Latency
B. Cyclomatic Complexity
C. Code Coverage
D. Code Smells
✅ **Answer:** C

4. What does static analysis NOT require?
A. Compiling the code
B. Access to the source code
C. No program execution
D. Syntax and semantics knowledge
✅ **Answer:** A

---

5. Which statement is TRUE about linting?
A. Linting is a dynamic test
B. Linting only works on compiled code
C. Linting detects formatting and style violations
D. Linting is used for unit testing
✅ **Answer:** C

---

6. In SonarQube, which metric indicates complexity in logic?
A. Duplications
B. Coverage
C. Code Smells
D. Cyclomatic Complexity
✅ **Answer:** D

---

7. What does “dead code” refer to?
A. Code not pushed to Git
B. Code that throws errors
C. Code that’s never executed
D. Code written in inactive branches
✅ **Answer:** C

---

8. What kind of issues can linting tools detect?
A. Runtime errors
B. Logical bugs
C. Indentation, unused imports, naming violations
D. Network issues
✅ **Answer:** C

---

9. Which of the following is considered a "code smell"?
A. Passing all unit tests
B. Using clear naming conventions
C. Deeply nested if-else blocks
D. Using design patterns
✅ **Answer:** C

---

10. What kind of code is said to have high maintainability?
A. Obfuscated code
B. Comment-free code
C. Clean, well-structured, and documented code
D. Hard-coded logic
✅ **Answer:** C

---

11. What does high cyclomatic complexity suggest?
A. Code is faster
B. Code has fewer bugs
C. Code has too many decision paths
D. Code is fully tested

✅ **Answer:** C

---

12. Which static analysis tool is most commonly used for JavaScript?
A. Flake8
B. ESLint
C. Pylint
D. JaCoCo
✅ **Answer:** B

---

13. Which of the following is a dynamic testing method?
A. SonarQube Analysis
B. Unit Testing
C. Linting
D. Style checking
✅ **Answer:** B

---

14. Code coverage tools are best used to:
A. Deploy the code
B. Minimize testing effort
C. Measure the effectiveness of test cases
D. Skip tests
✅ **Answer:** C

---

15. A major goal of static analysis is to:
A. Compile code
B. Run performance benchmarks
C. Catch errors early in development
D. Create test data
✅ **Answer:** C

---

16. What is the output of a linting tool?
A. Test results
B. Syntax tree
C. A list of style and formatting issues
D. Code execution trace
✅ **Answer:** C

---

17. What is the purpose of writing assertions in unit tests?
A. Deploy code
B. Validate expected outcomes
C. Log errors
D. Compile code
✅ **Answer:** B

---

18. What happens if you have 0% code coverage?
A. All code is tested
B. Some functions are covered
C. No automated test cases ran against code
D. 100% tests failed
✅ **Answer:** C

---

19. Which of the following is a reason to keep code complexity low?
A. Increase deployment frequency
B. Improve maintainability and reduce bugs

C. Enable dynamic typing
D. Reduce network load

✅ **Answer:** B

---

20. SonarQube can automatically fail a build if:
A. Comments are missing
B. Coverage is too high
C. Quality gate is not passed
D. CI server is down

✅ **Answer:** C