# MCQ

**Question 1:** Which code smell occurs when a class inherits methods or properties it doesn't need from a superclass?

1. Temporary Field
2. Refused Bequest
3. God Class
4. Alternative Classes with Different Interfaces

**Question 2:** What is the purpose of a "switch statement" in programming?

1. To create instances of objects.
2. To execute multiple cases sequentially.
3. To evaluate a single expression and execute different code blocks based on its value.
4. To define multiple entry points for a program.

**Question 3:** Which anti-pattern involves creating classes with a significant number of methods that are only used occasionally?

1. God Class
2. Temporary Field
3. Refused Bequest
4. Complex Switch Statements

**Question 4:** What is the primary concern with the "Refused Bequest" code smell?

1. Excessive use of temporary fields in a class.
2. Overusing switch statements for complex logic.
3. Unnecessary inheritance that doesn't fully use inherited methods.
4. Using alternative classes with different interfaces.

**Question 5:** Which anti-pattern involves creating different classes that offer similar functionality but present different interfaces to clients?

1. Alternative Classes with Different Interfaces
2. Refused Bequest
3. Complex Switch Statements
4. God Class

**Question 6:** In the context of object-oriented programming, what is the main issue with using switch statements excessively?

1. Switch statements are more efficient than if-else statements.
2. Switch statements can lead to code duplication.
3. Switch statements violate encapsulation principles.
4. Switch statements are only suitable for simple branching logic.

**Question 7:** What is a "Temporary Field" in the context of object-oriented programming?

1. A field used to store temporary values during program execution.
2. A field that is accessed by multiple classes.
3. A field that is marked as private to prevent external access.
4. A field that is used to store global variables.

**Question 8:** Which anti-pattern involves creating multiple classes with overlapping functionality and inconsistent interfaces?

1. Alternative Classes with Different Interfaces
2. God Class
3. Duplicated Code
4. Complex Switch Statements

**Question 9:** What is the potential impact of having alternative classes with different interfaces in a codebase?

1. It simplifies the codebase by reducing the number of classes.
2. It improves encapsulation and data hiding.
3. It increases code reusability and maintainability.
4. It introduces complexity and confusion for developers.

**Question 10:** What is the main problem with using temporary fields in a class?

1. Temporary fields are inaccessible to other classes.
2. Temporary fields increase the memory footprint of the class.
3. Temporary fields can lead to confusion and unintended behavior.
4. Temporary fields violate the Dependency Inversion Principle.

**Question 11:** Which design principle suggests that high-level modules should not depend on low-level modules, but both should depend on abstractions?

1. Dependency Inversion Principle
2. Liskov Substitution Principle
3. Interface Segregation Principle
4. Single Responsibility Principle

**Question 12:** What can be a consequence of using switch statements excessively in code?

1. Improved code readability and maintainability.
2. Enhanced encapsulation and modularity.
3. Increased likelihood of bugs when adding new cases.
4. Reduced cyclomatic complexity.

**Question 13:** Which code smell involves creating multiple classes with the same functionality but different interfaces?

1. Alternative Classes with Different Interfaces
2. God Class
3. Duplicated Code
4. Complex Switch Statements

**Question 14:** What is the primary goal of refactoring when it comes to code smells?

1. To introduce new features to the codebase.
2. To rewrite the entire codebase from scratch.
3. To improve the internal structure of the code while preserving its external behavior.
4. To fix all bugs in the code.

**Question 15:** What is an effective way to address code smells related to switch statements?

1. Introducing more complex if-else chains.
2. Replacing all switch statements with dynamic function calls.
3. Refactoring the code to use polymorphism and abstraction.
4. Removing all branching logic from the code.

**Question 16:** Which code smell involves a class that tries to do too much, handling multiple unrelated responsibilities?

1. Temporary Field
2. God Class
3. Refused Bequest
4. Complex Switch Statements

**Question 17:** What is the main advantage of addressing code smells and anti-patterns in software development?

1. It guarantees bug-free code.
2. It improves the aesthetics of the code.
3. It enhances code readability, maintainability, and quality.
4. It shortens the development time.

**Question 18:** In the context of object-oriented programming, what is the primary issue with classes that have different interfaces but offer similar functionality?

1. They increase the modularity of the codebase.
2. They simplify code documentation.
3. They can lead to confusion and complicate maintenance.
4. They improve code reusability.

**Question 19**: What does the acronym "DRY" stand for in software development?

1. Don't Repeat Yourself
2. Do Refactoring Yearly
3. Design Reusable Yarns
4. Develop Robust Yields

**Question 20:** Which software development principle emphasizes keeping code simple and avoiding unnecessary complexity?

1. Code Optimization Rule
2. KISS Principle
3. YAGNI Principle

4. WET Principle

**Question 21:** What does the principle "YAGNI" stand for in software development?

1. You Are Gaining New Insights
2. You Always Generate New Ideas
3. You Ain't Gonna Need It
4. Your Application Gets New Installs

**Question 22:** According to the "YAGNI" principle, what should you focus on when implementing features?

1. Implement all features requested by stakeholders.
2. Implement only the features that are popular in the industry.
3. Implement features that are predicted to be needed in the future.
4. Implement only the features that you currently need.

**Question 23:** Which principle suggests that developers should avoid duplicating code and strive for reusability?

1. DRY
2. WET
3. KISS
4. YAGNI

**Question 24:** What is the main idea behind the "WET" principle in software development?

1. Write Every Test
2. Write Everything Twice
3. Write Every Time
4. Write Every Term

**Question 25:** What is the main drawback of not following the "DRY" principle?

1. Increased code complexity
2. Faster development process
3. Improved code readability
4. Reduced debugging effort