DWA_07.4 Knowledge Check_DWA7

---

1. Which were the three best abstractions, and why?

Interface Segregation Principle – it makes the web page more user friendly
without confusing the user with unnecessary information and requirements.

Single-Responsibility Principle - it makes the code base more readable to other collaborators and developers looking to understand your code and also helps you modify the code without ruining other parts of the code.

Dependency Inversion Principle - it helps with making sure data is not tampered with by low level functions making security a bit better by not accessing information directly or modifying it.

---

2. Which were the three worst abstractions, and why?

Liskov Substitution Principle (LSP)

Why It Can Be Problematic:

Complexity in Implementation: Ensuring that subclasses can replace their base classes without altering the correct functioning of the program can introduce complexity. This often requires careful design and thorough testing.

Confusion for Developers: As mentioned, developers under tight deadlines might mistakenly call, modify, or remove the wrong class, leading to bugs or unintended behavior.
Rigid Hierarchies: Strict adherence to LSP can lead to rigid class hierarchies that are difficult to refactor or extend.

Open-Closed Principle (OCP)

Why It Can Be Problematic:

Difficulty in Bug Fixes: As you mentioned, OCP encourages extending code rather than modifying it, which can make simple bug fixes more complicated by requiring additional layers of abstraction or new classes.

Increased Complexity: Adhering strictly to OCP can result in more complex codebases with many small classes or modules, making the system harder to understand and maintain.

Principle of Least Astonishment

Why It Can Be Problematic:

Subjectivity: What is least astonishing can vary greatly between developers with different backgrounds and experiences, leading to inconsistent codebases.

Stifling Innovation: Strict adherence to this principle can discourage creative or unconventional solutions that might be more efficient or elegant.

---

3. How can The three worst abstractions be improved via SOLID principles.

1. **Liskov Substitution Principle (LSP)**

Issue: Confusion for developers, complexity in implementation, and rigid hierarchies.

Improvement via SOLID Principles:

Single Responsibility Principle (SRP):

- **Approach**: Ensure each class has one responsibility and thus simplifies the design and reduces the risk of confusion.

2. **Open-Closed Principle (OCP)**

Issue: Difficulty in bug fixes and increased complexity.

Improvement via SOLID Principles:

Single Responsibility Principle (SRP):

- **Approach**: Ensure classes have a single responsibility, making them easier to extend and fix without adding unnecessary complexity.

3. **Principle of Least Astonishment**

Issue: Subjectivity and potential to stifle innovation.

Improvement via SOLID Principles:

Interface Segregation Principle (ISP):

- **Approach**: Create smaller, more focused interfaces that are less likely to surprise developers by being clear and specific about their responsibilities.