DWA_07.4 Knowledge Check_DWA7

---

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

- A reusable function that performs a specific action and returns a result, which can be used in different parts of the codebase.
- A clearly defined and organized class hierarchy that makes code easier to understand and modify.
- Interfaces that are flexible and only expose the necessary functionality, allowing for better separation of concerns and easier testing.

---

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

- Over-engineered abstractions that add unnecessary complexity and reduce code readability and maintainability.
- Abstractions that expose too much internal implementation detail to the client code, making it harder to modify the implementation without breaking existing code.
- Interfaces that are too specific to a particular use case or implementation, limiting their usefulness in other contexts.

---

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

Single Responsibility Principle (SRP): This principle states that each class or function should have a single responsibility and should only be modified for one reason. For poorly abstracted code, it may be that a function or class has too many responsibilities, leading to tightly coupled code that is difficult to maintain. To improve this, the class or function can be split into smaller

---