DWA_07.4 Knowledge Check_DWA7

---

1. Which were the three best abstractions, and why?
Object-Oriented Programming (OOP):
OOP is a powerful abstraction that organizes software design around objects, which are instances of classes. It provides a way to model real-world entities, their properties, and their interactions. By encapsulating data and behavior within objects, OOP promotes modularity, code reuse, and maintainability. OOP concepts such as inheritance, polymorphism, and encapsulation have been instrumental in building large-scale, complex software systems.

Virtualization:
Virtualization is an abstraction that enables the creation of virtual resources, such as virtual machines (VMs) or virtual networks, which mimic the behavior of physical resources. It allows multiple operating systems or applications to run simultaneously on a single physical machine. Virtualization has revolutionized the IT industry by improving resource utilization, enabling efficient deployment and management of applications, and providing isolation and security between different computing environments.

Internet Protocol (IP):
IP is a fundamental abstraction that underlies the functioning of the Internet. It provides an addressing scheme and a set of rules for routing data packets across interconnected networks. IP allows devices to communicate with each other over the internet, regardless of their underlying physical infrastructure or the specific applications they run. This abstraction has facilitated global connectivity, enabling seamless communication, information sharing, and the development of countless internet-based services and applications.

---

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

God Objects:

God Objects are classes or modules that try to handle all functionalities and responsibilities within a system. They violate the Single Responsibility Principle (SRP) by encompassing too many responsibilities, making them difficult to understand, maintain, and test.

Tight Coupling:
Tight coupling occurs when modules or components depend heavily on each other, making it challenging to modify or replace one without affecting others. This violates the Dependency Inversion Principle (DIP) and makes the system less flexible and adaptable.

Spaghetti Code:
Spaghetti Code refers to poorly structured code with tangled and unorganized control flow. It often lacks clear abstractions and violates the Open/Closed Principle (OCP), making it difficult to extend or modify the codebase without introducing unintended side effects.

---

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

God Objects:

To improve a God Object, you can apply the SOLID principle of Single Responsibility. Split the responsibilities into separate classes, each with a single responsibility. This promotes better encapsulation, improves code organization, and makes the system more maintainable.

Tight Coupling:
.
To address tight coupling, you can apply the SOLID principle of Dependency Inversion. Introduce abstractions and dependency injection to decouple modules. By depending on interfaces or abstract classes instead of concrete implementations, you can swap implementations easily, improve testability, and reduce dependencies.

Spaghetti Code:

To improve Spaghetti Code, you can apply the SOLID principle of Open/Closed. Encapsulate varying behavior behind abstractions (e.g., interfaces or abstract classes)

and use polymorphism to provide different implementations. This enables you to add new functionality by extending existing abstractions without modifying existing code, making it more maintainable and extensible.
