DWA_08 Discussion Questions

In this module you will continue with your “Book Connect” codebase, and further iterate on your abstractions. You will be required to create an encapsulated abstraction of the book preview by means of a single factory function. If you are up for it you can also encapsulate other aspects of the app into their own abstractions.

To prepare for your session with your coach, please answer the following questions. Then download this document as a PDF and include it in the repository with your code.

---

1. What parts of encapsulating your logic were easy?

Creating the factory function and returning the object.

1. Modularization: Breaking down complex functionality into smaller, self-contained modules is generally straightforward. It allows for better code reuse and easier maintenance, as each module focuses on a specific task or feature.

2. Function and method definition: Defining functions and methods to encapsulate logic is a fundamental programming concept and typically has a clear syntax and structure. It enables code readability and promotes the principle of "Don't Repeat Yourself" (DRY).

3. Utilizing control flow structures: Employing control flow structures, such as conditional statements (if-else, switch) and loops (for, while), to encapsulate different branches of logic is relatively easy. These structures enable the execution of specific code paths based on conditions or iterations.

4. Organizing data structures: Creating and organizing data structures, like classes, structs, or dictionaries, provides a way to encapsulate related data and operations. Defining classes with properties and methods allows for encapsulation of behavior and data manipulation within a well-defined structure.

Overall, encapsulating logic into smaller, reusable components and leveraging programming constructs makes it easier to manage and understand codebases, leading to improved maintainability and code quality.

---

2. What parts of encapsulating your logic were hard?

1. Identifying appropriate boundaries: Determining the optimal points at which to encapsulate logic can be challenging, especially in complex systems. Deciding how to break down functionality into modules or classes requires careful consideration to strike a balance between granularity and cohesion.

2. Managing dependencies: Encapsulating logic often involves creating dependencies between different components. Ensuring proper dependency management and minimizing tight coupling can be tricky, particularly when dealing with interconnected modules or when making changes to existing code.

3. Balancing abstraction and complexity: Encapsulating logic should aim to simplify code and improve maintainability, but excessive abstraction can introduce its own complexities. Striking the right level of abstraction and designing clear interfaces can be a delicate task, as overly abstracted code can be harder to understand and debug.

By overcoming these challenges requires experience, thoughtful design, and iterative development practices. It is essential to carefully analyze the system's requirements, consider trade-offs, and continuously refine the encapsulation approach as the codebase evolves.

---

3. Is abstracting the book preview a good or bad idea? Why?

Abstracting the book preview can be a good idea as it promotes code reusability and maintainability. By encapsulating the logic related to book previews, it becomes easier to modify or extend the preview functionality without affecting other parts of the codebase.
