1. **Command Design Pattern**:
   * Explanation: The Command pattern encapsulates a request as an object, allowing for parameterization of clients with queues, requests, and operations.
   * Added Value: Utilizing the Command pattern in Jabberpoint allows for the decoupling of the sender of a request from the object that executes the command. This promotes extensibility and flexibility in handling user interactions with the presentation.
   * Role: Commands can be used to implement actions such as navigating slides, changing slide layout, or modifying text. Each command represents a specific action that can be executed on the presentation.
   * Comparison: When comparing with other creational patterns like Factory, Command focuses on encapsulating a request as an object, while Factory focuses on creating objects without exposing the instantiation logic to the client. Command is more suitable for handling user actions and interactions.
2. **Factory Design Pattern**:
   * Explanation: The Factory pattern provides an interface for creating objects in a superclass but allows subclasses to alter the type of objects that will be created.
   * Added Value: In Jabberpoint, using the Factory pattern for creating different types of slides (e.g., title slide, content slide, image slide) allows for flexibility in adding new slide types without modifying existing client code. It promotes encapsulation and separation of concerns.
   * Role: Factories can be employed to create instances of various slide types based on user input or configuration. This enables the application to support different slide formats and content structures seamlessly.
   * Comparison: Compared to the Command pattern, which focuses on encapsulating requests, Factory concentrates on creating objects. Factory is more suitable for managing the creation of complex objects or families of related objects.
3. **Decorator Design Pattern**:
   * Explanation: The Decorator pattern attaches additional responsibilities to an object dynamically. It provides a flexible alternative to subclassing for extending functionality.
   * Added Value: Applying the Decorator pattern in Jabberpoint allows for dynamically adding features to slides or slide elements (e.g., adding borders, shadows, animations) without modifying their structure. This enhances modularity and maintainability.
   * Role: Decorators can be used to augment the behavior or appearance of slides or slide elements. For example, decorators can add animations to slides or apply formatting to text elements.
   * Comparison: Unlike the Command and Factory patterns, which focus on encapsulating requests and creating objects, respectively, Decorator concentrates on dynamically adding functionality to objects. It's more suitable for enhancing or extending the behavior of existing objects at runtime.

In summary, each chosen design pattern (Command, Factory, and Decorator) serves a distinct purpose in enhancing the design and functionality of Jabberpoint. The Command pattern facilitates the encapsulation of user actions, the Factory pattern supports flexible object creation, and the Decorator pattern enables dynamic augmentation of object functionality. Depending on the specific requirements and design goals of Jabberpoint, these patterns can be effectively employed to improve modularity, extensibility, and maintainability of the application.