M3: Design Patterns

One of the significant primary functionalities of EduPool, an e-learning platform, is that instructors, as content creators, can post various types of content on a course page. Students can view the content, which can take many forms, such as an assignment or an announcement.

This process can be broken down into two distinct pieces:

- Instructors creating varied content for the EduPool platform.
- Observers (students) being notified of and viewing the content.

This process can, hence, be created using two distinct design patterns integrated with each other in their implementation: the Factory model (creational), which is responsible for creating different types of content, and the Observer model (behavioural), which notifies observers (students) of the new content.

This integration will bring several known advantages from both models. On the one hand, the Factory Model will bring flexibility to the system, making adding new types of content objects easier going forward. It also allows for decoupling the content creation from the controller code and hides this aspect from the calling controller code, too. On the other hand, the Observer model will enable us to dynamically add and subtract groups of observers, providing a consistent way to notify them about uploaded or edited content (object state change). When integrated together, these models complement each other's functionality to make EduPool more robust and significantly more flexible, scalable, and maintainable since the functionalities have been decoupled into various classes.

The implementation would briefly be as follows:

- Factory Model
    - contentFactory [Interface]
        - createContent() [method]: Concrete methods for creating different content will be based on this.
    - Concrete Factories:
        - assignmentFactory [class]: will override the createContent() method to make assignments.
        - announcementFactory [class]: will override the createContent() method to make announcements.
    - Content [Interface]: Blueprint for concrete content classes such as assignments, mentioning relevant and specific information.
    - Concrete Content Classes:
        - assignment [class]
        - announcement [class]

- Observer Model
    - subject [class]
        - observers [list<Observer>]: list of all observers
        - register(observer) [method]: adds observers from subject
        - deregister(observer) [method]: removes observers from subject
        - notify() [method]: lets all registered observers know of the change
    - observer [interface]

        - update() [method]: updates self about the information and reacts to it
    - Concrete observer classes
        - student [class]

- Calling Method
    - The instructor creates content using a concrete factory's createContent() method (described above in the Factory model). This returns an instance of the content added to the subject.
    - When the content is added to the subject, it notifies all registered observers by calling the update() method for all of them.
    - Observers receive notifications and access the updated course content.