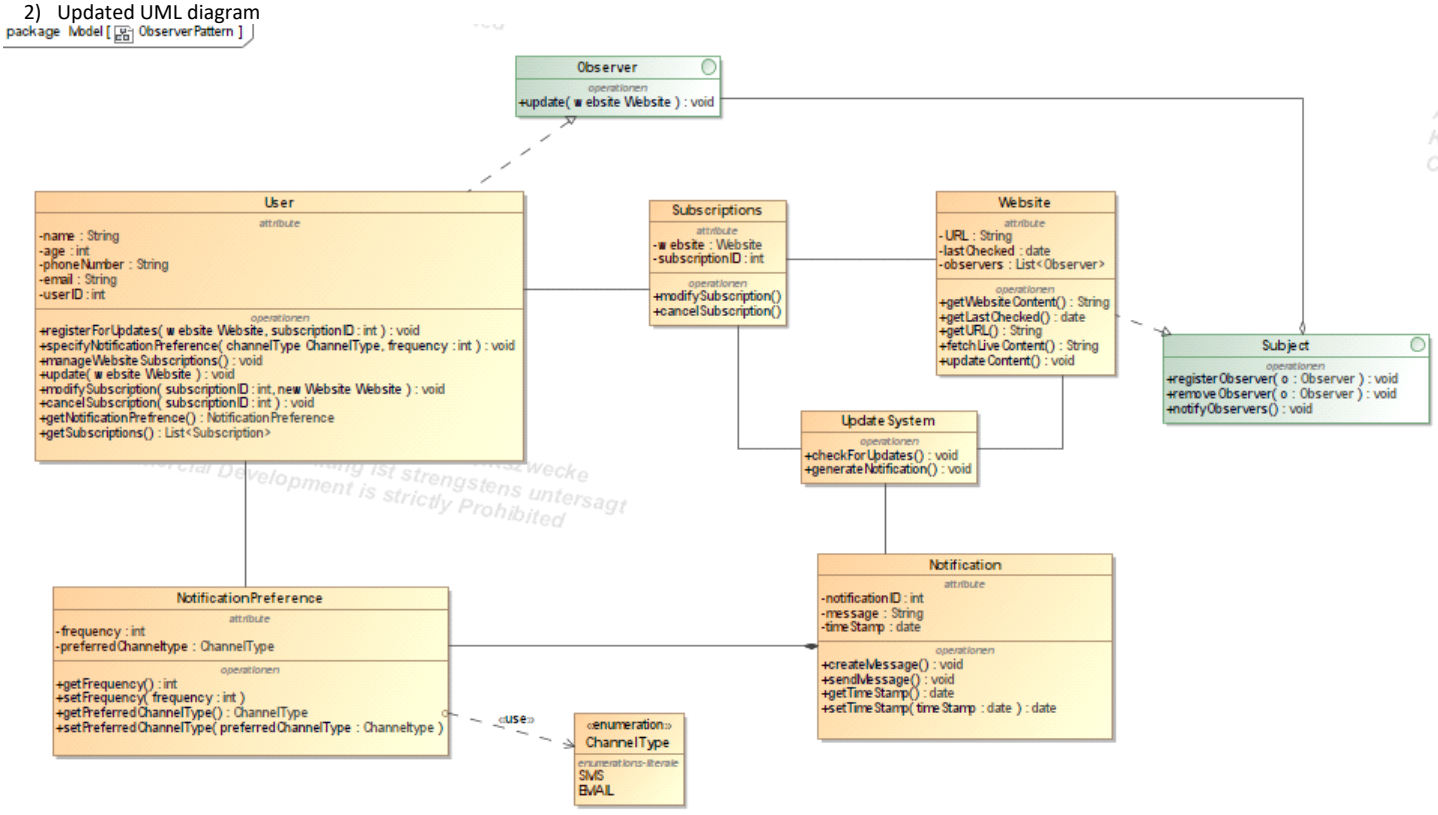
Exercise 5

Montag, 26. Mai 2025 08:32

1) Observer Pattern: Behavioural Design Pattern (ab Folie 8: Pattern Erklärungen)
- Defines a one-to-many dependency between objects so that when one object changes state, all ist dependents are notified and updates automatically
- Notifies multiple objects or subscribers, about any events that happen to the object they are observing
- Subject: Website (beobachtbar)
- Observer: User (registriert sich beim Subjekt und wird benachrichtigt, wenn sich etwas ändert)

2) Updated UML diagram



- The UML diagram represents a WebsiteMonitoring System
- Users subscribe to websites and are notified automatically when website content changes
- The design follows the Observer Pattern for loose coupling and event driven communication:
  - Website: Subject
  - User: Observer

User:
- Implements the Observer interface
- Represents a person using the system

Website:
- Respresents a website being monitored
- Implements the Subject interface
- Maintains its content and a list of all registered users (observers)

Subscription:
- Represents a user's subscription to a website
- Links a user to a website (connects)
- Has a unique subscriptionID

Notification:
- Represents a message send to the user when a website update is detected
- Includes ID, message content and timestamp

NotificationPreference:
- Stores a users's preference for how and how often they want to be notified
- Uses the Channeltype enum for channel selection

UpdateSystem:
- Utility class that checks if the websites were updates and generates notifications
- Uses Website and NotificationPreference

Observer (Interface):
- Declares user(website: Website) method
- Implemented by user

Subject (Interface):
- Declares register, modify and remove Observers

- Implemented by Website

Channeltype (Enum):

- Enum with fixed values (SMS, EMAIL)
- Used in NotificationPreference as preferredChannel

Arrows:

- Realization (dashed line with triangle): User-->Observer, Website-->Subject
- Dependency (dashed line with <<use>>): NotificationPreference-->ChannelType
- Association (line): generic relationship between two classes
- Composition (line with filled diamond): one class own another class exclusively (when the whole is destroyed, the part is too)

Anmerkungen:

- Klassen in Pakete teilen, damit es nicht so unübersichtlich ist
- Kreisläufe vermeiden im UML Diagramm
- Immer Rückgabetyp angeben, auch wenn void