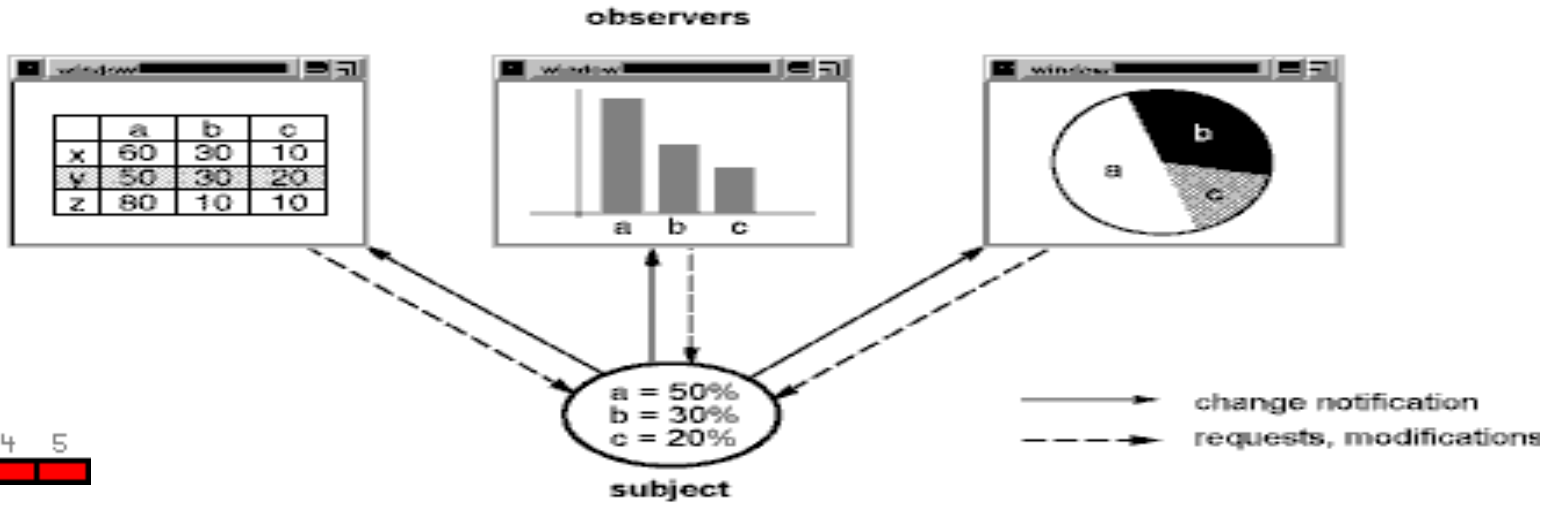
Un exemple :

Observer (comportement)

- **Intention**
  - Définir une dépendance 1-N de telle façon que si l'objet change d'état tous ses dépendants sont prévenus et mis à jour automatiquement
- **Synonymes :** Dépendants, Publier/Abonner, Publish/Subscribe
- **Motivation**
  - Un effet secondaire de découper un logiciel en classes coopératives est la nécessité de maintenir la cohérence des objets associés. Le faire par des classes **fortement couplées** entrave leur réutilisabilité



Fréquence :

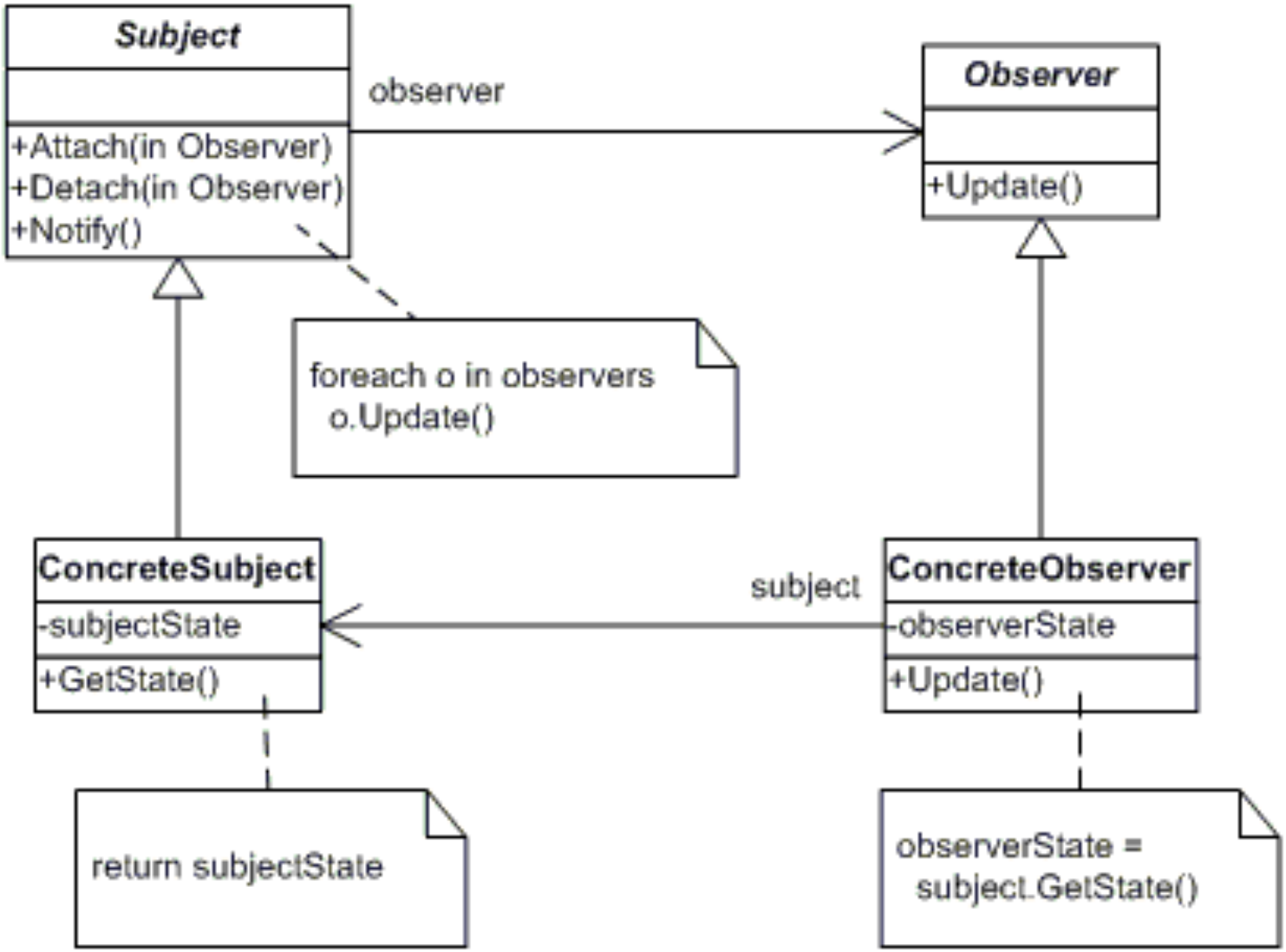
1 2 3 4 5

Observer (2)

- **Champs d'application**
  - Quand une abstraction a deux aspects, un dépendant de l'autre. Encapsuler ces aspects dans des objets séparés permet de les faire varier et de les réutiliser indépendamment
  - Quand un changement sur un objet nécessite de modifier les autres et qu'on ne peut savoir combien d'objets doivent être modifiés
  - Quand un objet doit être capable de notifier d'autres objets sans faire de suppositions sur qui sont ces objets, c'est-à-dire que les objets ne doivent pas être fortement couplés

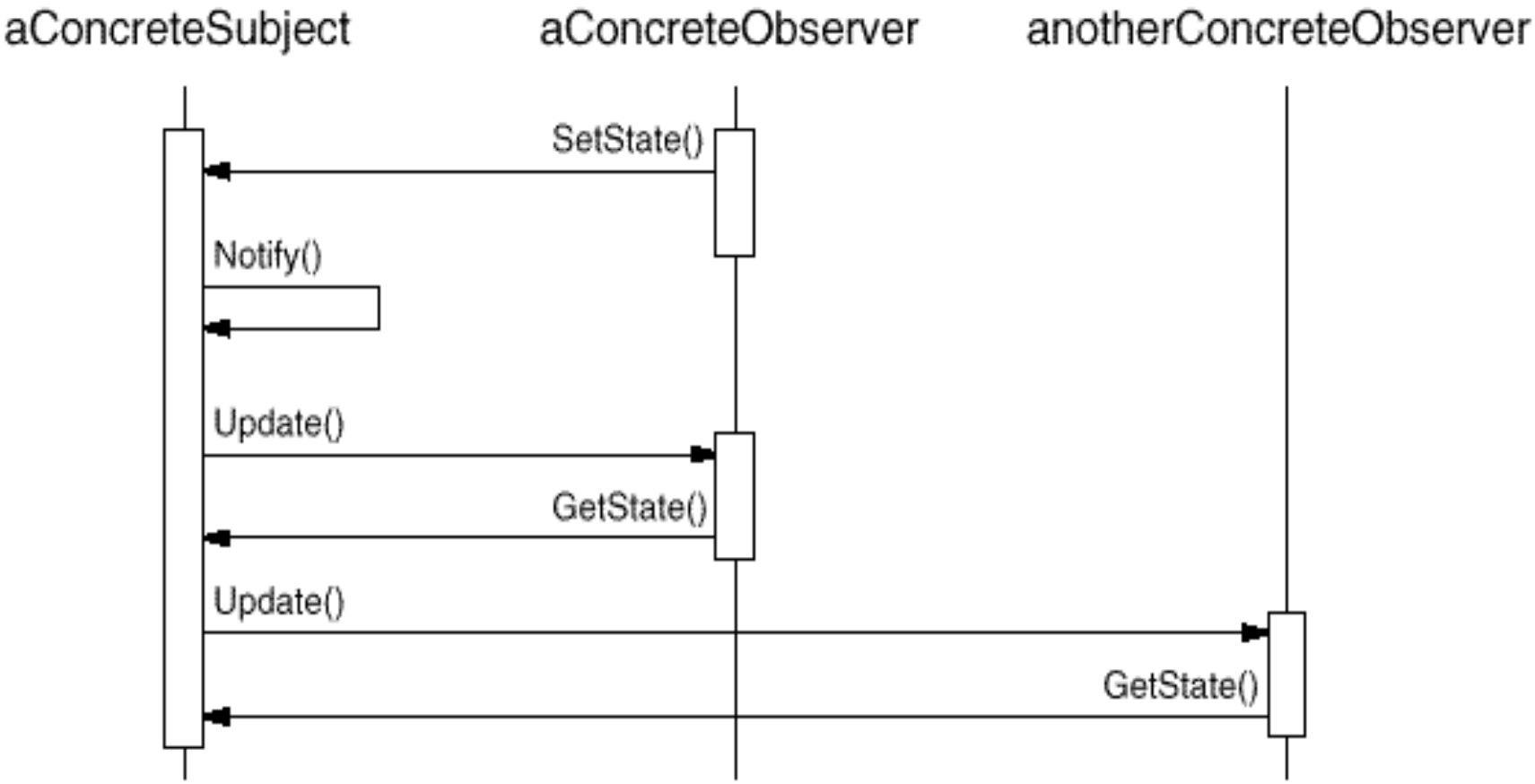
Observer (3)

- **Structure**



Observer (4)

- **Collaborations**



Observer (5)

- **Conséquences**
  - Variations abstraites des sujets et observateurs
  - Couplage abstrait entre le sujet et l'observateur
  - Prise en charge de communication broadcast
  - Mises à jour non prévues
    - Protocole additionnel pour savoir ce qui a précisément changé

Observer (6)

- **Implémentation**
  - Référence sur les observateurs, *hash-table*
  - Un observateur peut observer plusieurs sujets
    - Étendre le *update* pour savoir qui notifie
  - Attention lors de la suppression des observés
  - Attention à la consistance de l'observé avant notification
  - Attention aux informations de notification
- **Patterns associés**
  - Mediator, Singleton