**Adapter, Bridge si Proxy**

**Adapter** este un model structural de proiectare care permite obiectelor cu interfețe incompatibile să colaboreze. Modelul adaptor funcționează ca o punte între două interfețe incompatibile.

**Avantajele Adapter pattern:**

* Principiul responsabilității unice (SRP): Interfața sau codul de conversie a datelor poate fi separat de logica principală de afaceri a aplicației.
* Open/Closed Principle (OCP)
* Flexibilitate și mentenabilitate:

**Dezavantajele Adapter pattern:**

* Complexitate
* Utilizare excesivă

**Bridge** este un model de proiectare structurală care vă permite să împărțiți o clasă mare sau un set de clase strâns legate între ele în două ierarhii separate - abstracție și implementare - care pot fi dezvoltate independent una de cealaltă.

**Avantajele Bridge** **pattern:**

* Decuplarea abstractizării de implementare: Această separare facilitează modificarea sau extinderea atât a abstractizării, cât și a implementării, fără a se afecta reciproc.
* Flexibilitate și extensibilitate
* Mentenabilitatea îmbunătățită

**Dezvantajele Bridge** **pattern:**

* Complexitate
* Utilizare excesivă
* Potential Over-Abstraction

**Proxy** este un model de proiectare structurală care vă permite să oferiți un substitut sau un înlocuitor pentru un alt obiect. Un proxy controlează accesul la obiectul original, permițându-vă să efectuați ceva înainte sau după ce cererea ajunge la obiectul original.

**Avantajele Proxy** **pattern:**

* Controlul Accesului: roxy-ul oferă o interfață pentru a controla accesul la obiectul real. De exemplu, se pot impune restricții de securitate sau reguli de validare înainte de a permite accesul la obiectul real.
* Optimizare a Performanței: Acest lucru poate reduce încărcarea inițială a aplicației și economisi resurse.
* Lazy Loading: Proxy-ul poate amâna încărcarea obiectului real până când este necesar.

**Dezvantajele Proxy** **pattern:**

* Complexitate
* Potențiale Probleme de Sincronizare
* Posibile Overhead-uri de Performanță