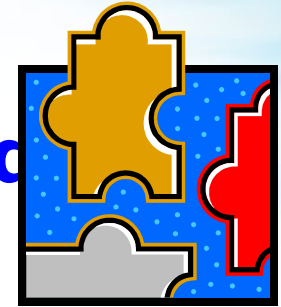


Structural Design Patterns

Structural Patterns

- Structural patterns describe how classes and objects can be **combined** to **form larger structures**.



- You're often required to choose between **Inheritance** and **Composition**.
 - Inheritance is usually simpler and requires less coding & less allocations.
 - Composition may allow much more flexibility.

> Adapter

- > force a class to conform to a new interface.

> Bridge

- > Separates abstraction from implementation so that both may vary (and evolve) separately.

> Composite

- > An object may consist of other objects – recursively.



> Decorator

- > dynamically add functionality to objects.

> Flyweight

- > share objects to save allocations.

This may require to remove state variables, exchanging them with parameters passed to methods.



> Façade

- > Simplify the access to a complex system.

> Proxy

- > The access to an object may be controlled by another object, with a similar interface.

- During the following discussion, you may note that some patterns have **similar outlines**.
- However, those patterns will reflect **different problems**.
 - Different program evolution (maintenance).
 - Different emphasis points.

