# Bridge Patterns

Bridges gaps between representation and implementation.

How? Using intermediate interface or parent class

What problems solved: Tight coupling

Advantage: More scalable

Sample code:

# Builder Patterns

Separate large object creation from its representation so that same construction process creates different representations.

Client Code -> Maker -> Uses Builder type (Which is a type of builder)

In other sense

**Client Code call 🡪 Director 🡪 Builder 🡪 Concrete Builder 🡪 Actual Product**

# Adapter Patterns

# Façade Patterns

# Factory Patterns

## A Very Simple Factory

Pass an argument and factory will return instance based on param.

Factory basically maintains dictionary of all types in an assembly and when we pass an object type as param then it returns from that dictionary.

## Factory Method

## Abstract Factory