Builder Pattern

★ Gamma Categorization

⇒ Design patterns are typically split into three categories:

↳ This is called Gamma categorization after Erich Gamma, one of GoF authors.

① **Creational Pattern**

↳ Deal with the creation (construction) of objects.

② **Structural Pattern**

↳ Concerned with the structure of classes that are involved.

↳ Stress the importance of good API design.

③ **Behavioral Patterns**

↳ They are all different; no central theme.

★ Builder design pattern

"When construction gets a little bit too complicated"

⇒ Some objects are simple and can be created in a single Constructor call.

⇒ Other objects require a lot of ceremony to create.

⇒ Having an object with 10 Constructor arguments is not productive.

⇒ Instead, opt for piecewise construction.

⇒ Builder provides an API for constructing an object step-by-step.

Example

HtmlElement ← Class you want to build

HtmlElementBuilder ← Class used to build HtmlElement

⇒ Fluent interface { return class reference }

⇒ You can have static HtmlElementBuilder ↑ function inside the HtmlElement class that return instance of HtmlElementBuilder.

⇒ You can define operator HtmlElement() const for implisit Conversion HtmlElementBuilder to HtmlElement.

⇒ You can make Constructor of HtmlElement private to force use of HtmlElementBuilder.

↳ Make HtmlElementBuilder friend in HtmlElement.

⇒ Domain specific language ~~[illegible]~~ with c++.

↳ Using initializer list.

★ Builder Facets

"Facets is another design pattern"

⇒ More than one builder to work on an object.