# Builder Design Pattern

The Builder Pattern is a creational pattern whose intent is to separate the construction of a complex object from its representation so that you can use the same construction process to create different representations.

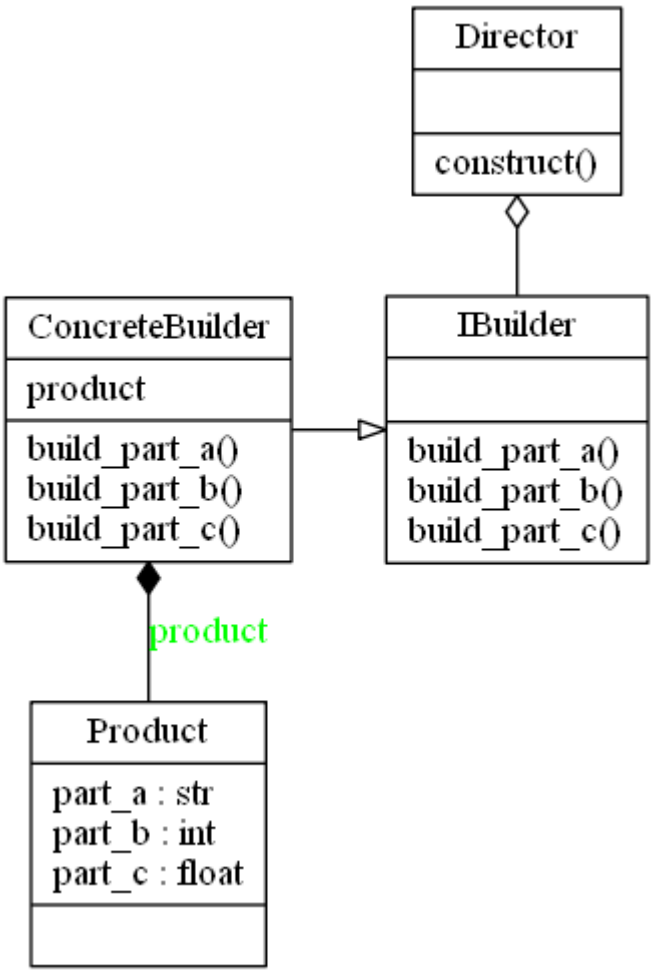
The Builder Pattern tries to solve,

- How can a class create different representations of a complex object?
- How can a class that includes creating a complex object be simplified?

The Builder and Factory patterns are very simiar in the fact they both instantiate new objects at run time. The difference is when the process of creating the object is more complex, so rather than the Factory returning a new instance of `ObjectA`, it could call the builders director construct method `ObjectA.construct()`. Both return an Object.

Parts of the Builder Pattern

1. **Product** - The Product being built
2. **Concrete Builder** - Build the concrete product. Implements the IBuilder interface
3. **Builder Interface** - The Interface which the Concrete builder should implement
4. **Director** - Has a construct method which when called creates a customised product




The Builder Pattern in the context of a House Builder. There are multiple directors creating there own complex objects