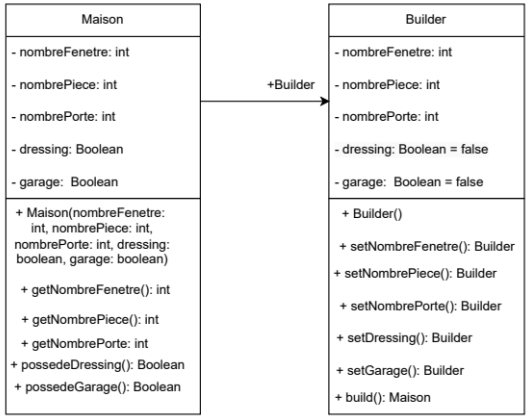
Récapitulatif : Le Pattern Builder

Le pattern **Builder** fut formalisé et standardisé en 1994 par les « GoF » comme beaucoup d'autres patterns du même type, le Builder a pour objectif principal de résoudre les problèmes liés à la création de constructeur, en effet son objectif principal est de faciliter la création d'objet comportant de nombreux attributs.

Le concept principal de ce pattern est simple : séparer la construction d'un objet et sa représentation finale, pour arriver à ce résultat on doit respecter la structure suivante :

On crée une classe produit qui va contenir l'objet qu'on veut créer, suivie d'**une classe Builder** qui contiendra les méthodes pour créer l'objet. Bien sûr l'ajout d'une interface Builder est possible si l'on veut utiliser plusieurs Builders.

Exemple : On a un objet « Maison » qui comporte plusieurs attributs, nombre de fenêtres, nombre de pièces, nombre de portes, la présence ou non d'un dressing, d'un garage, etc.



Pour simplifier tout ça on crée donc une classe Builder suivant ce modèle.

On a donc notre classe Builder qui va nous permettre de créer de nouvelles maisons de manière beaucoup plus simple.

Principe SOLID : Le pattern Builder respecte deux principes SOLID,

- **le principe de responsabilité unique** dû au fait qu'il ne crée qu'un objet précis, ici il ne s'occupe que de créer une maison et rien d'autre.

- **le principe Open/Close** car il n'est pas ouvert aux modifications mais extrêmement facile à étendre, dans l'exemple ci-contre ajoute le nombre de lit ne demande que 3 modifications.

Avantage	Inconvénients
- Lisibilité et maintenance facilitées - Simplifie le constructeur - Rend le constructeur flexible	- Création d'une classe en plus - Intérêt limité si aucun constructeur n'est pas complexe