# Builder

*Créez un projet Java simple pour les exercices. Les exercices sont dépendants et doivent être réalisés dans l'ordre.*

## Exercice 1 - La souris

*Créer une énumération "Color" qui contient les valeurs "Yellow", "Green", "Red", "Black", "White"*

*Créer une classe Mouse qui contient les champs :*

- *rgb (boolean)*
- *wired (boolean)*
- *wireLength (float)*
- *color (Color)*

*Ajoutez des getters et un constructeur (pas de setters !).*

## Exercice 2 - le CPU

*Créer une énumération "Chipset" qui contient les valeurs "Intel" et "AMD".*

*Créer une classe CentralProcessingUnit qui contient les champs :*

- *frequency (float)*
- *chipset (Chipset)*
- *hyperthreading (boolean)*
- *coreCount (integer)*

## Exercice 3 - Les barrettes mémoire

*Créer une énumération "DDRType" qui contient les valeurs "DDR2", "DDR3", "DDR4"*

*Créer une classe "MemoryModule" qui contient les champs :*

- *frequency (float)*
- *memory (integer)*
- *ddrType (DDRType)*

# *Exercice 4 - L'ordinateur*

*Créer une classe Computer qui contient les champs :*

- *mouse (Mouse)*
- *cpu (CentralProcessingUnit )*
- *memoryModules (List<MemoryModule>)*

# *Exercice 5 - Instancier l'ordinateur*

*Dans une méthode main, instancier un Computer qui a :*

- *Une souris de couleur Rouge, qui n'a pas le RGB, qui est wired, avec un cable de longueur 2.5.*
- *Un cpu Intel 3.2Ghz sur 8 coeurs avec hyperthreading*
- *2 barrettes mémoires, l'une ayant :*
  - *4 Go*
  - *DDR4*
  - *3200 Mhz*
- *et l'autre :*
  - *8 Go*
  - *DDR4*
  - *3600 Mhz*

# *Exercice 6 - Les builders vides*

*Créer une classe ComputerBuilder qui a une méthode :*

*public Computer Build()*

*qui renvoie une nouvelle instance d'un Computer (mettez à null les champs du constructeur pour l'instant).*

*Faites de même pour un MouseBuilder, un ProcessingUnitBuilder et un MemoryModuleBuilder.*

# *Exercice 7 - Utilisez les sous-builders*

*La classe ComputerBuilder va maintenant se reposer sur les autres builders pour assembler l'ordinateur.*

*Ajoutez à ComputerBuilder trois attributs :*

- *mouseBuilder (MouseBuilder)*
- *processingUnitBuilder (ProcessingUnitBuilder)*
- *memoryModuleBuilders (List<MemoryModuleBuilder>)*

*Instanciez ces attributs dans un constructeur vide.*

*Au sein de la méthode "build" de ComputerBuilder, au lieu de donner "null" comme valeur pour les attributs de Computer, nous allons maintenant utiliser les objets renvoyés par les méthodes "build" respectives des autres builders.*

*Dans le cas de la liste, il faudra transformer la liste de MemoryModuleBuilder en une liste de MemoryModule.*

*Modifiez le code de "ComputerBuilder.build" en conséquence.*

*A ce stade, notre code ne fait rien d'intelligent, mais nous avons "câblé" nos builders. Il ne nous reste plus qu'à rendre nos builders utiles.*

# *Exercice 8 - MouseBuilder*

*Ajoutez les champs suivants à MouseBuilder :*

- *rgb (boolean)*
- *wired (boolean)*
- *wireLength (float)*
- *color (Color)*

*Initialiser ces champs avec des valeurs par défaut de votre choix (pour faire une souris "classique").*

*Dans la méthode "build" de MouseBuilder, utilisez ces champs pour construire une instance de Mouse.*

*Pour chacun de ces champs, ajoutez un setter. Le setter peut renvoyer le builder actuelle pour permettre une syntaxe fonctionnelle :*

```
public Builder setValue(int value)
{
    this.value = value;
    return this;
}
```

# Exercice 9 - CentralProcessingUnitBuilder

*Faites de même pour CentralProcessingUnitBuilder.*

# Exercice 10 - MemoryModuleBuilder

*Faites de même pour MemoryModuleBuilder.*

# Exercice 11 - Récupérer les sous-builders

*Nous allons maintenant permettre à l'utilisateur d'accéder aux sous-builders de ComputerBuilder.*

*Créez un getter dans ComputerBuilder nommé "setMouse", qui renvoie l'attribut mouseBuilder.*

*Créez un getter dans ComputerBuilder nommé "setCentralProcessingUnit", qui renvoie l'attribut centralProcessingUnit.*

*(notez l'inversion où notre getter s'appelle "set", car il a pour objectif de modifier l'instance).*

# Exercice 12 - Ajouter de la mémoire

*Les modules mémoires sont un peu différents, car il s'agit d'une liste. Il est possible d'ajouter plusieurs barrettes de mémoire, nous allons donc à la place créer une méthode "addMemoryModule".*

*La méthode addMemoryModule ajoute un MemoryModuleBuilder à la liste et renvoie cette nouvelle instance à notre utilisateur.*

*Plusieurs appels à addMemoryModule permettrons donc d'ajouter plusieurs barettes mémoire.*

# Exercice 13 - Utilisez ComputerBuilder

*Utilisez ComputerBuilder pour instancier un Computer qui a :*

- *Une souris de couleur Rouge, qui n'a pas le RGB, qui est wired, avec un cable de longueur 2.5.*
- *Un chipset Intel 3.2Ghz sur 8 coeurs*
- *2 barrettes mémoires, l'une ayant :*
    - *4 Go*
    - *DDR4*
    - *3200 Mhz*
- *et l'autre :*
    - *8 Go*
    - *DDR4*
    - *3600 Mhz*

# *Exercice 14 - Navigation arrière*

*Actuellement, lorsque nous accédons au MouseBuilder, nous sommes coincés avec celui-ci et devons arrêter notre chaîne d'instruction pour retourner sur le ComputerBuilder.*

*Nous allons modifier MouseBuilder pour que cela ne soit plus le cas.*

*Pour cela, nous allons ajouter un attribut ComputerBuilder à MouseBuilder.*

*Le constructeur de MouseBuilder va prendre en argument un ComputerBuilder (et n'aura donc plus de constructeur par défaut).*

*Lorsque ComputerBuilder va instancier un MouseBuilder, il va se passer lui-même au MouseBuilder.*

*Nous allons ensuite ajouter une méthode "and" à MouseBuilder qui renvoie le ComputerBuilder.*

*L'objectif est de pouvoir écrire ceci :*

```
Computer computer = new ComputerBuilder()
    .setMouse()
          .setRGB(true)
          .setColor(Color.White)
          .and() // retour au ComputerBuilder
    .build();
```

*Appliquez la même logique à MemoryModuleBuilder et à CentralProcessingUnitBuilder.*

# *Exercice 15 - Utilisez ComputerBuilder (encore)*

*Tout est dans le titre.*

*Reprenez l'exercice 13, mais instancier un Computer en une seule grande séquence d'appels, en utilisant les retours en arrière via "and()".*

# *Exercice 16 - Interfaces comme guides*

*Il nous reste un problème à résoudre. Actuellement, l'utilisateur peut écrire :*

```
new ComputerBuilder()
    .setMouse()
          .setRGB(true)
          .setColor(Color.White)
          .build();
```

*Cette séquence va instancier une souris et interrompre la séquence, mais ce n'est pas ce que nous souhaitons et cela peut induire notre utilisateur en erreur.*

*Un bon pattern "builder" ne permet pas à l'utilisateur d'utiliser des fonctions dans le mauvais ordre ou des fonctions qui ne lui sont pas destinées.*

*Pour cela, nous introduire des interfaces. L'interface IMouseBuilder est implémentée par MouseBuilder et contient toutes les méthodes de MouseBuilder sans la méthode "build".*

*La méthode setMouse de ComputerBuilder ne renverra pas un MouseBuilder mais l'interface MouseBuilder. Cela nous servira à masquer la méthode "build" de MouseBuilder.*

*Faites de même avec MemoryModuleBuilder et CentralProcessingUnitBuilder.*