Sudoku : Etape 2

Exercice évolutif d’apprentissage du langage Java et de la conception objet.

# Objectifs

1. Mise en place d’une AbstractFactory pour la gestion de plusieurs versions de Sudoku.
   * Le pattern Abstract Factory permet, à partir d'une interface, de créer des objets d’une même famille sans en connaître le type exact. Une fabrique abstraite [encapsule](http://fr.wikipedia.org/wiki/Encapsulation_(programmation)) un ensemble de [fabriques](http://fr.wikipedia.org/wiki/Fabrique_(patron_de_conception)) ayant une thématique commune. On demandera à la fabrique de créer le sudoku correspondant la version demandée.

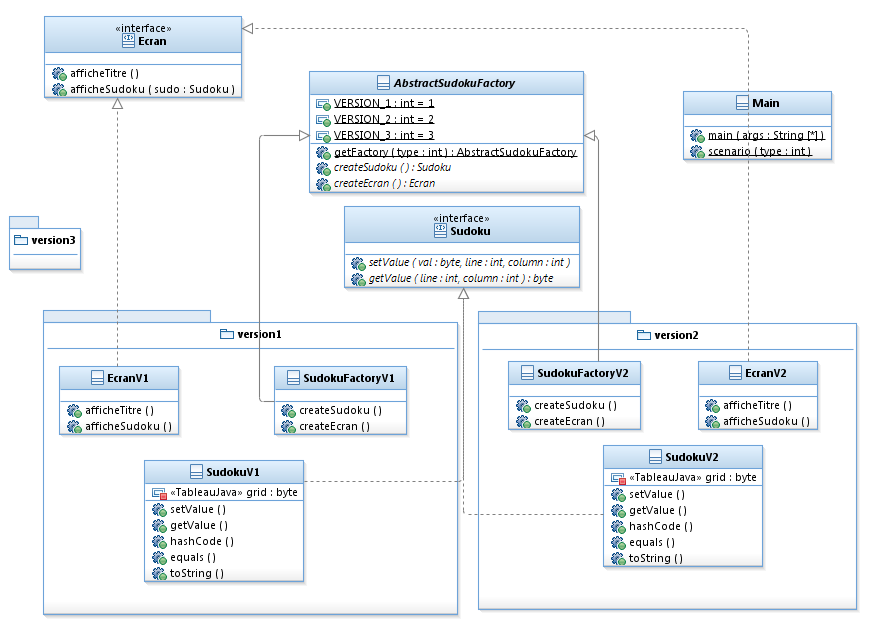
# Apprentissage

1. AbstractFactory
2. Renommer une classe avec Eclipse.
3. Extraction d’interface avec Eclipse.
4. Déplacement de classes de package avec Eclipse.
5. String.format

# Etapes de modification

1. Renommer Ecran en EcranV1 et Sudoku en sudokuV1
2. Extraire les interfaces (refractor : extract interface) :
   * De SudokuV1 vers Sudoku (cocher getValue() et setValue())
   * De EcranV1 vers Ecran(cocher afficheTitre() et afficheSudoku())
3. Création du package «  ste.sudoku.version1 ».
   * Déplacement de EcranV1 et SudokuV1 dans le package (refractor move).
4. Création des classes
5. Implémentation des classes :
   * Les classes du package version1 gère un sudoku de 9 \* 9
   * Les classes du package version2 gère un sudoku de 4 \* 4
   * Les classes du package version3 gère un sudoku de 21 \* 21 avec 369 valeurs
   * NB : La version3 contient des cases vides, la méthode getValue doit renvoyer -1 dans le cas d'une case vide
6. Implémentation des factory
   * AbstractSudokuFactory n’implémente que la méthode getFactory(version :int)
     + En fonction du type elle crée et retourne soit :
       - Une SudokuFactoryV1,
       - Une SudokuFactoryV2.
       - Une SudokuFactoryV3
   * SudokuFactoryV1 implémente :
     + createSudoku() : crée et retourne une nouvelle SudokuV1
     + createEcran() : crée et retourne une nouvelle EcranV1
   * SudokuFactoryV2 implémente :
     + createSudoku() : crée et retourne une nouvelle SudokuV2
     + createEcran() : crée et retourne une nouvelle EcranV2
   * SudokuFactoryV3 implémente :
     + createSudoku() : crée et retourne une nouvelle SudokuV3
     + createEcran() : crée et retourne une nouvelle EcranV3

# Diagramme de classes



# Description des classes

## Main

Classe contenant le méthode main() et différents scénario d’utilisation

### public static void main(String[] args)

Appel de scenario() avec des paramètres différents.

### public static void scenario (int type)

1. Création d’une instance de AbstractFactory via la méthode static et le type en paramètre.
2. Création d’une instance de Sudoku grâce à la factory
3. Création d’une instance de Ecran grâce à la factory.
4. Affectation de valeurs au Sudoku.
5. Affichage du titre grâce à l’écran.
6. Affichage du Sudoku grâce à l’écran.

## EcranV1

### public void afficheSudoku(Sudoku sudo)

Affiche un sudoku de 9 sur 9.

## SudokuV1

Gère un sudoku de 9 \* 9 valeurs

## EcranV2

### public void afficheSudoku(Sudoku sudo)

Affiche un sudoku de 4 sur 4.

## SudokuV2

Gère un sudoku de 4 \* 4 valeurs

## EcranV3

### public void afficheSudoku(Sudoku sudo)

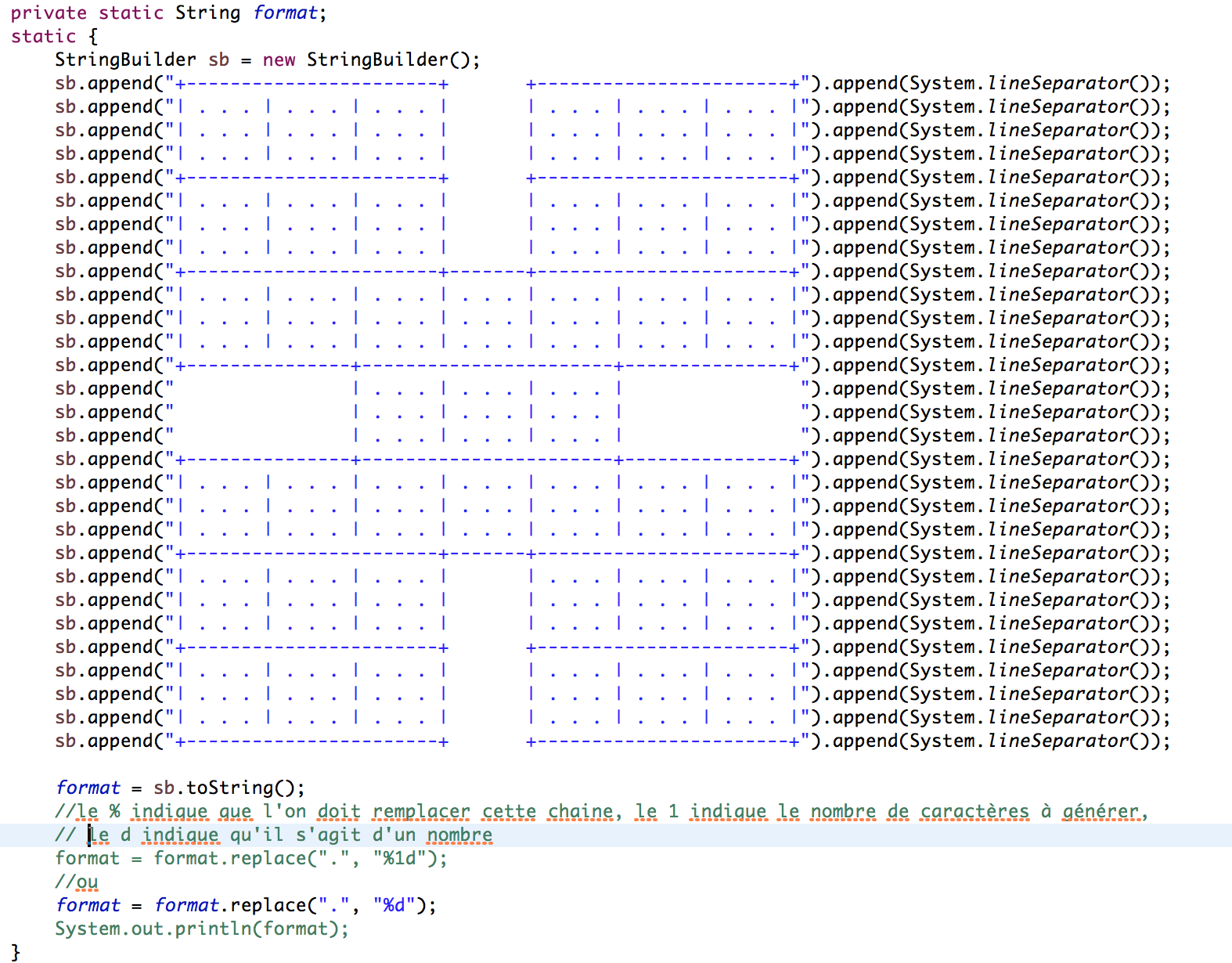
Affiche un sudoku de 21 sur 21, avec des zones « vides »

## SudokuV3

Gère un sudoku de 21 \* 21 avec 369 valeurs

## Classes « Ecran »

Les classes écrans peuvent utiliser une variable d'instance privée statique de type String représentant le format d'affichage. La méthode **String.format(String format, Object... args)** sera utilisée dans la méthode d'affichage du sudoku.



# Copie écran

