Sudoku : Etape 1

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

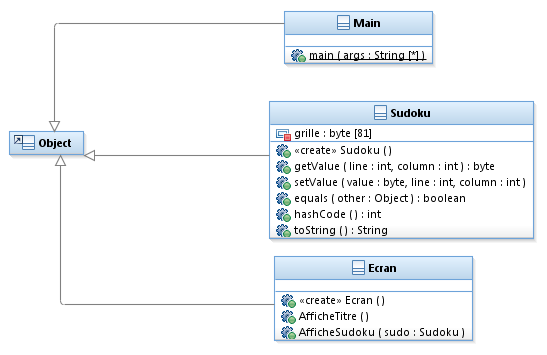
# Objectifs

1. Affichage d’une grille de Sudoku

# Apprentissage

1. Création du projet et organisation package
2. Gestion de la documentation :
   1. /\*\*
   2. @params
   3. @return
3. Création d’une classe Sudoku
   1. setValue(byte,int,int)
   2. getValue(int,int)
   3. tableau de byte 9x9
4. Création d’une classe d’affichage
5. Méthode main manipule une instance de Sudoku et de Ecran.
6. Surcharge des méthodes :
   1. equals()
   2. hashCode()
   3. toString()
7. type primitif : int, byte, boolean
   1. casting en byte.
8. tableau à 2 dimensions de byte.
   1. Arrays.deepHashCode()
   2. Arrays.deepEquals()
   3. Arrays.deepToString()
9. StringBuilder dans Ecran

# Diagramme de classes



# Description des classes

## Ecran

### public void afficheTitre()

Affiche le titre Sudoku

### public void afficheSudoku(Sudoku sudo)

Utilisation d’un StringBuilder pour construire la grille à afficher.

## Sudoku

### public byte getValue(int line, int column)

Retourne la valeur à la ligne *line* et colonne *column*

### public voit setValue(byte value, int line, int column)

Insère la valeur à la ligne *line* et colonne *column*

Surcharge des méthodes equals(), hashCode(), toString()

## Main

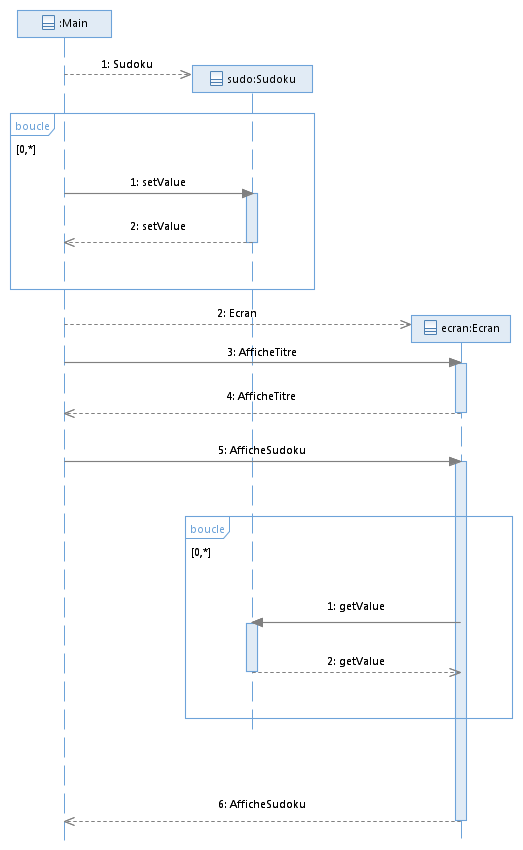
Classe contenant le méthode main() et différents scénario d’utilisation

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

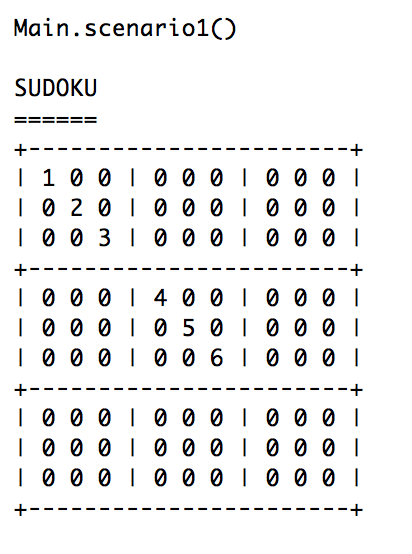
Appel des différents scenarios d’utilisation.

### public static void scenario1()

#### Diagramme de séquence scénario 1



1. Création d’une instance de Sudoku.
2. Création d’une instance de Ecran.
3. Affectation de valeurs au Sudoku.
4. Affichage du titre grâce à l’écran.
5. Affichage du Sudoku grâce à l’écran.



### public static void scenario2()

1. Vérification de la méthode equals

* Symétrie : pour deux références a et b, si a.equals(b) alors il faut obligatoirement que b.equals(a)
* Réflexivité : pour toute référence non null, a.equals(a) doit toujours renvoyer true
* Transitivité : si a.equals(b) et b.equals(c) alors a.equals(c)
* Consistance avec la méthode hashCode() : si deux objets sont égaux en invoquant la méthode equals() alors leur méthode hashCode() doit renvoyer la même valeur pour les deux objets (L'inverse n'est pas vrai, deux objets dont la méthode hashCode() renvoie la même valeur, n'implique pas obligatoirement que l'invocation de la méthode equals() sur les deux objets renvoie true.)
* Pour toute référence non null, a.equals(null) doit toujours renvoyer false

1. Vérification de la method hashCode()

La redéfinition de la méthode hashCode() doit explicitement respecter plusieurs règles :

* La valeur renvoyée doit être constante lors de plusieurs invocations sur un même objet durant la durée de vie de l'application. Cette valeur n'a cependant pas d'obligation à être constante entre plusieurs exécutions de l'application
* Deux objets égaux (l'invocation de la méthode equals() sur une instance avec l'autre en paramètre renvoie true) doivent obligatoirement avoir le même hash code.
* Si deux objets ne sont pas égaux en invoquant la méthode equals(), alors l'invocation de la méthode hashCode() de chacun des objets n'a pas l'obligation de renvoyer des valeurs entières différentes