ENSI CAEN
ÉCOLE PUBLIQUE D'INGÉNIEURS
CENTRE DE RECHERCHE

Kata Eratosthène avec Intellij

Refactoring d'un code avec Intellij IDEA

1. Code du crible d'Ératosthène

Soit l'exemple suivant d'un code étudiant sur le crible d'Ératosthène. Nous listons ici les étapes et les raccourcis claviers utilisés pour améliorer la qualité du code.

```
1  package fr.ensicaen.sf.demo_5;
2  import java.util.*;
3  /**
4   * Cette classe génère les nombres premiers jusqu'à un maximum
5   * spécifié par l'utilisateur. L'algorithme utilise le crible d'Ératosthène
6   * <p>
7   * Ératosthène de Cyrène, b. c. 276 BC, Cyrène, Libye --
8   * d. c. 194, Alexandrie. Le premier homme à calculer la
9   * circonférence de la Terre.<p>
10  * L'algorithme est plutôt simple. Étant donné un tableau d'entiers
11  * commençant à 2. Éliminer tous les multiples de 2. Chercher le
12  * prochain entier non éliminé, et éliminer tous ses multiples.
13  * Répéter jusqu'à la racine carré du maximum.
14  *
15  * @author Alphonse
16  * @version 13 Fev 2012
17  */
18 public class Eratosthene
19 {
20     /**
21      * @param valeurMax le nombre limite de la génération.
22      */
23     public static int[] genereNombresPremiers( int valMax )
24     {
25   if (valMax >= 2)  // le seul cas valide
26     {
27         // déclarations
28         int s = valMax + 1; // taille du tableau
29         boolean[] f = new boolean[s];
30         int i;
31         // initialise le tableau à vrai.
32         for (i = 0; i < s; i++)
33           f[i] = true;
34         f[0] = f[1] = false;   // se débarrasser des nombres premiers connus
35        // crible
36         int j;
37         for (i = 2; i < Math.sqrt(s) + 1; i++)
38         { // si i n'est pas éliminé, éliminer ses multiples.
39           if (f[i] != false)
40               for (j = 2 * i; j < s; j += i)
```

```
41                  f[j] = false; // les multiples ne sont pas premiers
42          }
43          // combien de premiers sont faits ?
44          int compteur = 0;
45          for (i = 0; i < s; i++)
46            if (f[i] != false)
47                compteur++; // ajoute au compteur
48          int[] premiers = new int[compteur];
49          // mettre les nombres premiers dans le tableau des résultats
50          for (i = 0, j = 0; i < s; i++)
51        if (f[i] == true) // si premier
52            premiers[j++] = i;
53          return premiers; // retourne les nombres premiers
54      }
55      else // valeurMax < 2
56          / retourne un tableau nul si l'entrée est mauvaise.
57          return new int[0]; /
58       }
59  }
```

1.1 Réécriture de l'entête

1. Ajouter un entête avec le copyright (**Alt + insert**). Cela suppose qu'il y ait des textes de licence définis dans les Settings (**Ctrl + Alt + S** puis « copyright » dans le champ texte).
2. Supprimer la partie HTML du commentaire : /** et les deux <p>. *Ce n'est pas une API donc pas de texte de documentation Javadoc.*
3. Supprimer les commentaires concernant Ératosthène (**CTRL + Y**). *Ce n'est pas un cours d'histoire*.
4. Supprimer @version (**Ctrl + Y**). *Ce commentaire relève d'un gestionnaire de version comme git*).

1.2 Mettre les {} sur les instructions unilignes et reformater

5. Remettre quelques '{' à la main. *Il faut toujours mettre des '{' autour des instructions pour éviter des erreurs de programmation*.
6. Faire quelques reformatages à la main pour les '{'.
7. Mais ensuite faire le reformatage automatiquement (**CTRL + ALT + L**).
8. Finir à la main parce que les commentaires empêchent le reformatage complet.
9. Vérifier que le programme fonctionne toujours avec **Shift + F10**.

1.3 Ajouter le mot clé « final » la classe

Il est toujours bénéfique pour l'efficacité du code de signifier explicitement qu'une classe n'a pas de dérivée (cf. implantation des méthodes virtuelles).

10. Commencer par taper « f » puis final par **CRTL + Space**.

1.4 Renommer la méthode statique

11. Renommer la méthode statique en *GenereNombresPremiersInferieursA()* (**Shift + F6**). *Le nom est ainsi plus explicite*.

1.5 Renommer le paramètre de la méthode statique

12. Renommer le paramètre en *nombreLimite* (**Shift + F6**). *Remarquer que le nom actuel est en désaccord avec la documentation HTML*.
13. Supprimer la documentation Javadoc (**Ctrl + Y l ligne par ligne, ou CTRL + W 4 fois puis Delete**). *Encore une fois, nous ne développons pas une API.*

1.6 Éliminer les commentaires inutiles

14. `// Déclaration` (**Ctrl + Y** sur la ligne pour la supprimer).
15. `// Initialise le tableau à vrai` (**Ctrl + Y**).
16. `// Ajoute au compteur` (**Ctrl + Y**).
17. `// Retourne les nombres entiers`. Ce mettre sur le caractère // et faire **Shift + End puis Delete**).

1.7 Inverser le test de validité

Il est toujours préférable d'évacuer les tests non intéressants en premiers.

18. Inverser le test en se plaçant sur le « if » puis **Alt + Enter**.

En fait, on va en faire trois :

1. `if (borneMax ≤ 0)` throw new IllegalArgumentException. Ur cela, on s'aide de la complétion. Puis, **Alt + enter** pour ajouter le throw dans la signature.

2. if (`borneMax == 1`) : `return new int[0];`
3. else le crible.

1.8 Mettre la valeur de s dans la taille du tableau f.

19. `boolean[] f = new boolean[s];` : Se positionner sur s et remplacer la variable s dans sa définition (**Ctrl + Alt +N**).

1.9 Remplacer s par f.length

20. Le tableau « f » porte sa longueur, il ne faut donc pas utiliser de variable non liée.
21. Supprimer la ligne de la déclaration de « s » (**Ctrl + Y**).
22. Remplacer chaque occurrence de « s » par « f.length »en utilisant le multicurseur :
    1. Se positionner sur le premier s.
    2. **Alt + J** pour sélectionner chaque occurrence de s (ici 4 fois)
    3. Remplacer s par f.length
    4. **Esc** pour finir.

1.10 Renommer « f » en « candidats »

23. Changer le nom en « candidats » (**Shit + F6**)

1.11 Méthodes

24. *InitializeTableauCandidatsMoinsNombresPremiersConnus*().
    Ctrl + Alt + M avec le code sélectionné pour faire la méthode.
    Remarque commencer à 2.
25. *crible*()
    1. Changer *`candidats[i] != false`* par *estPremier(candidats[i])*.
26. Faire de « *candidats* » un attribut « *_candidats* » (**Ctrl + Alt + F**).
27. Changer le prototype des deux premières fonctions en supprimant le

paramètre « candidats » (**Ctrl + F6**)

28. Avec le reste : *construitNombresPremiersAvecCandidatsRestants*. (**Ctrl + Alt + M**)

1.12 Les imports

29. Optimise les imports (**CTRL + Alt + O**)

1.13 Exécution

30. Exécution **Shit +F10**

31. puis **Shit + Esc** pour fermer la console.

1.14 Code final


```java
package fr.ensicaen.sf.demo_5;
/*
 * Copyright (c) 2018 ENSICAEN
 *
 *  ENSICAEN, École publique d'ingénieurs et centres de recherche, Caen, FRANCE.
 *  https://www.ensicaen.fr
 *
 * This file is owned by ENSICAEN students.
 * No portion of this document may be reproduced, copied
 * or revised without written permission of the authors.
 */
package fr.ensicaen.sf.demo_5;
/*
 * Cette classe génère les nombres premiers jusqu'à un maximum
 * spécifié par l'utilisateur. L'algorithme utilise le crible d'Ératosthène
 *
 * L'algorithme est plutôt simple.
 * Étant donné un tableau d'entiers commençant à 2.
 * Éliminer tous les multiples de 2.
 * Chercher le prochain entier non éliminé, et éliminer tous ses multiples.
 * Répéter jusqu'à la racine carré du maximum.
 *
 * @author Alphonse
 */
public final class Eratosthene {
    private static boolean[] _candidats;
    public static int[] genereNombresPremiersInferieursA( int nombreLimite ) {
        if (nombreLimite < 1) {
            throw new IllegalArgumentException(Integer.toString(nombreLimite));
        } else if (nombreLimite < 2) {
            return new int[0];
        } else {
            _candidats = new boolean[nombreLimite + 1];
            initialiseTableauCandidatsSansNombresPremiersConnus();
            crible();
            return construitNombresPremiersAvecCandidatsRestants();
        }
    }
    private static int[] construitNombresPremiersAvecCandidatsRestants() {
```

```
40          int nombreNombresPremiers = 0;
41          for (boolean _candidat : _candidats) {
42              if (isPremier(_candidat)) {
43                  nombreNombresPremiers++;
44              }
45          }
46          int[] premiers = new int[nombreNombresPremiers];
47          for (int i = 0, j = 0; i < _candidats.length; i++) {
48              if (isPremier(_candidats[i])) {
49                  premiers[j++] = i;
50              }
51          }
52          return premiers;
53      }
54      private static void crible() {
55          for (int i = 2; i < Math.sqrt(_candidats.length) + 1; i++) {
56              if (isPremier(_candidats[i])) {
57                  for (int j = 2 * i; j < _candidats.length; j += i) {
58                      _candidats[j] = false;
59                  }
60              }
61          }
62      }
63      private static boolean isPremier( boolean candidat ) {
64          return candidat;
65      }
66      private static void initialiseTableauCandidatsSansNombresPremiersConnus() {
67          for (int i = 2; i < _candidats.length; i++) {
68              _candidats[i] = true;
69          }
70      }
71      public static void main( String[] args ) {
72          int[] nombres = Eratosthene.genereNombresPremiersInferieursA(100);
73          for (int nombre : nombres) {
74              System.out.print(nombre + ", ");
75          }
76          System.out.println("");
77      }
78  }
```