CSC4102 : Qualité du code JAVA et introduction aux idiomes JAVA — Compléments

Denis Conan

Janvier 2024

Sommaire

1 Exemples SpotBugs

1.1 Exemple SpotBugs a/e

```
1 X x2 = new X2(...);
2 h2.put(x2, "c'est un objet");
```

- *« X2 doesn't define a hashCode() method but is used in a hashed data structure in main(String[]) »*
  - *« A class defines an equals(Object) method but not a hashCode() method, and thus doesn't fulfill the requirement that equal objects have equal hashCodes. An instance of this class is used in a hash data structure, making the need to fix this problem of highest importance. »*

— Le problème vient de ce que la classe possède une méthode `equals` mais pas de méthode `hashCode`

- Rappelons-nous qu'un dictionnaire (ajout avec méthode `put`) utilise `hashCode` puis `equals` de la classe de la clef pour chercher un objet

1.2 Exemple SpotBugs b/e

```
1 pi = null;
2 for(it= vPersonnes.iterator();it.hasNext();pi=it.next()) {
3   if(pi.equals(new Personne("Dupont"+5, "Jules","19102271271"+5))) {
4     it.remove();
5   }
6 }
```

- « Déréférencement d'un pointeur null dans la méthode `main(String[])` »
  - « Un pointeur à `null` est déréférencé ici. Ceci va mener à une `NullPointerException` quand le code sera exécuté. »
    - Le problème est à la ligne 2 : l'affectation « `pi = it.next()` » n'a lieu qu'en fin de boucle ;
      donc, lors de l'entrée dans la boucle `for`, `pi` vaut sa valeur avant la boucle, c'est-à-dire ici `null`

1.3 Exemple SpotBugs c/e

```
1 if (p.getPrenom() == p1.getPrenom()) {//...
```

- « Comparaison d'objets String utilisant == ou !=. »
  - « Ce code compare des objets `String` au moyen de l'égalité par référence des opérateurs == ou !=. À moins que les deux chaînes ne soient des constantes dans le fichier source ou aient été internalisées au moyen de la méthode `String.intern()`, deux chaînes identiques peuvent être représentées par deux objets `String` différents. Envisagez d'utiliser la méthode `equals(Object)` à la place. »
    - SpotBugs signale l'utilisation de l'opérateur « == » à la place de la méthode « equals »
    - C'est sans aucun doute une erreur pour des variables qui sont des références sur des objets

1.4 Exemple SpotBugs d/e

```
1 Calendar o2 = o1;
2 System.out.println("o1 == o2 : " + (o1 == o2) + ",\t o1.equals(o2) : " + (o1.
      equals(o2)));
```

- This method compares a local variable with itself, and may indicate a typo or a logic error. Make sure that you are comparing the right things.
  - L'alerte vient de la combinaison de l'affectation de la ligne 1 avec l'appel de la méthode `equals` à la ligne 2

1.5 Exemple SpotBugs e/e

```
1 public Audio(final String code, final Localisation localisation,
2              final String titre, final String auteur, final String annee,
3              final Genre genre, final String classif) {
4   super(code, localisation, titre, auteur, annee, genre);
5   if (classif == null) {
6     throw new IllegalArgumentException("Ctr␣Audio␣classification␣=␣" +
          classification); }
7   this.classification = classif;
8 }
```

- « Lecture du champ Audio.classification non initialisé dans new Audio(...). »
  - « Ce constructeur lit un champ qui n'a pas encore été initialisé. Une des causes les plus fréquentes est l'utilisation accidentelle par le développeur du champ au lieu d'un des paramètres du constructeur. »
    - Le problème est à la ligne 6 : on voulait sans aucun doute utiliser l'argument classif au lieu de l'attribut classification
      - Mais, dans ce cas, on sait que c'est null qui sera affiché

2 Exemples CheckStyle

2.1 Exemple CheckStyle a/d

Classe `eu.telecomsudparis.csc4102.util.Datutil`

```
1           /**
2            * compare la date avec aujourd'hui : vrai si aujourd'hui.
3            *
4            * @param date la date a comparer avec le jour courant.
5            * @return vrai si aujourd'hui
6            * @throws IllegalArgumentException si {@code date} est {@code null}
7            */
8           public static boolean dateEstAujourdhui(final LocalDate date) {
9                   if (date == null) {
10                          throw new IllegalArgumentException("date == null");
11                  }
12                  return date.equals(aujourdhui());
13          }
```

http://www.oracle.com/technetwork/java/javase/documentation/index-jsp-135444.html

[Javadoc, JSE8]

Notez que les commentaires Javadoc commencent par la chaîne de caractères « /** »

2.2 Exemples CheckStyle b/d

- Vérification de la présence et correction « du Javadoc » avec CheckStyle

```
1 /**
2  * Bla bla.
3  * @param e explication sur e.
4  */
5 public void ajouter(final int a, final int b) { };
```

- « Balise javadoc @param inutilisé pour ‘e’. »
- « Balise javadoc @param manquante pour ‘a’. »
- « Balise javadoc @param manquante pour ‘b’. »

2.3 Exemples CheckStyle c/d

- Aussi présenté dans [Bloch, 2008], *Item 8*

```
1 private int Valeur;
2 public void MaMethode() { //...
```

  - « Le nom ‘Valeur’ n’est pas conforme à l’expression ‘^[a-z][a-zA-Z0-9]*$’. »
  - « Le nom ‘MaMethode’ n’est pas conforme à l’expression ‘^[a-z][a-zA-Z0-9]*$’. »

- Standard d’écriture lexicale du code JAVA

Paquetage	`seance5.mediathequesimplifie`, `eu.telecomsudparis.csc4102`
Classe, interface	`Timer`, `HashMap`, `HttpServlet`
Méthode, attribut	`remove`, `ensureCapacity`, `getVote`
Constantes	`MIN_VALUE`, `NEGATIVE_INFINITY`
Variable locale	`i`, `somme`, `sommeVote`
Paramètre de type	`T`, `V`, `K`, `T1`

2.4 Exemple CheckStyle d/d

```
1 public int b;
```

- « La variable ‘b’ devrait être privée et avoir des accesseurs. »

```
1 public String toString() { return "A␣[Valeur=" + valeur + "]"; }
```

- « overrides java.lang.Object.toString »
  - Ajouter `@Override`, sinon on risque une erreur difficile à trouver
    - P.ex. si on écrit
      « `public String toString(boolean versionCourte) {...}` »
- Autre exemple plus difficile à trouver pour le corriger :
  « `public boolean equals(Document obj) {...}` »

```
1 public int ajouter(int a, int b) {
```

3 Exemples sur les pipelines des *Streams*

3.1 Exemple sur les pipelines des *Streams* a/i

- Début : génération d'une séquence infinie avec `IntStream.iterate`
  Intermédiaire : troncature avec `limit` + insertion déverminage avec `peek`
  Terminaison : test de présence avec `anyMatch`
- Démonstration de l'évaluation tardive

Classe `seance7/lambdaexpressions/ExampleNaftalin2015`

```
1 boolean trouve = IntStream.iterate(1, i -> i * 2)
2                           .limit(10)
3                           .peek(System.out::println)
4                           .anyMatch(v -> v == 16);
```

Affichage :

```
1
2
4
8
16
```

3.2 Exemple sur les pipelines des *Streams* b/i

- Début : `stream de Collection`
  Intermédiaire : filtrage avec `filter`
  Terminaison : recherche d'un élément avec `findFirst`

- Recherche dans une collection avec filtrage
  + obtention de la première valeur trouvée

Classe `etudesdecas/mediathequeaveclambdasoptionaletstreams/Mediatheque`

```
1 /**
2  * chercherGenre cherche un Genre dans la liste des genres.
3  *
4  * @param nomGenre du Genre a chercher
5  * @return le genre correspondant au nom dans la collection
6  */
7 private Optional<Genre> chercherGenre(final String nomGenre) {
8   return lesGenres.stream().filter(g -> g.getNom().equals(nomGenre)).
        findFirst();
9 }
```

La classe `Optional` est étudiée dans la section qui suit celle-ci.

3.3 Exemple sur les pipelines des *Streams* c/i

- Début : `stream de Collection`
  Intermédiaire : filtrage avec `filter`
  Terminaison : test de présence avec `anyMatch`
- Existe-t-il un document avec le genre donné ?

Classe `etudesdecas/mediathequeaveclambdasoptionaletstreams/Mediatheque`

```
1 /**
2  * cherche un document dont le genre est indique en parametre.
3  *
4  * @param g Genre du document a chercher
5  * @return true s'il en existe un false sinon
6  */
7 private boolean existeDocument(final Genre g) {
8  return lesDocuments.values().stream().anyMatch(d -> d.getGenre().equals(g));
9 }
```

3.4 Exemple sur les pipelines des *Streams* d/i

- Début : génération d'une séquence infinie avec `IntStream.iterate`
  Intermédiaire : troncature avec `limit` + transformation avec `mapToObj`
  Terminaison : réduction avec `collect` en une liste avec `toList de Collectors`
- Collecte dans une liste

Classe `seance7/lambdaexpressions/ExampleNaftalin2015`

```
1 List<String> l = IntStream.iterate(1, i -> i * 2)
2                           .limit(10)
3                           .mapToObj(String::valueOf)
4                           .toList(); // idem .collect(Collectors.toList())
5 System.out.println(l);
```

Affichage :

```
[1, 2, 4, 8, 16, 32, 64, 128, 256, 512]
```

- Cet exemple démontre implicitement une construction d'objet
  - On pourrait écrire `mapToObj(Integer::new)` pour récupérer une `List<Integer>`

Depuis JAVA 16, `.collect(Collectors.toList())` est écrit plus simplement `.toList()`

3.5 Exemple sur les pipelines des *Streams* e/i

- Début : génération d'une séquence infinie avec `IntStream.iterate`
  Intermédiaire : troncature avec `limit` + transformation avec `mapToObj`
  Terminaison : réduction avec `collect` en une chaîne de caractères avec `joining de Collectors`
- Collecte dans une chaîne de caractères

Classe `seance7/lambdaexpressions/ExampleNaftalin2015`

```
1 String s = IntStream.iterate(1, i -> i * 2)
2                     .limit(10)
3                     .mapToObj(String::valueOf)
4                     .collect(Collectors.joining("␣+␣"));
5 System.out.println(s);
```

Affichage :

```
1 + 2 + 4 + 8 + 16 + 32 + 64 + 128 + 256 + 512
```

3.6 Exemple sur les pipelines des *Streams* f/i

- Début : génération d'une séquence infinie avec `IntStream.iterate`
  Intermédiaire : troncature avec `limit`
  Terminaison : réduction avec `reduce`
  à partir d'une valeur de départ dite « identité » et avec une fonction d'accumulation telle que accumulateur.apply(identité, t).equals(t)

Classe `seance7/streams/ExemplesStreamsReduce`

```
1 IntStream stream = IntStream.iterate(1, i -> i * 2).limit(10);
2 int result = stream.reduce(0, Integer::sum); // identite, accumulateur
3 //equivalent a :
4 stream = IntStream.iterate(1, i -> i * 2).limit(10);
5 int resultBis = 0; // 0 : valeur dite identite
6 for (Integer element : stream.toArray()) {
7     resultBis = Integer.sum(resultBis, element);
8 }
9 System.out.println(result + "␣=␣" + resultBis);
```

Affichage :

```
1023 = 1023
```

3.7 Exemple sur les pipelines des *Streams* g/i

- Début : génération d'un *stream*
  Terminaison : application d'un effet de bord avec `forEach`
- Exemple d'application d'un effet de bord

Classe `seance7/lambdaexpressions/ExampleNaftalin2015`

```
1 List<Point> points = Arrays.asList(new Point(1, 2), new Point(2, 4));
2 System.out.println(points);
3 points.stream().forEach(p -> p.translate(1, 2));
4 System.out.println(points);
```

Affichage :

```
[java.awt.Point[x=1,y=2], java.awt.Point[x=2,y=4]]
[java.awt.Point[x=2,y=4], java.awt.Point[x=3,y=6]]
```

3.8 Exemple sur les pipelines des *Streams* h/i

- Début : génération de deux *streams*
  Intermédiaire : concaténation des deux *streams* avec `concat de Stream`
  + retrait des doublons avec `distinct`
  Terminaison : collecte dans un ensemble avec `toSet`

- Démonstration de la concaténation de *streams*

Classe `seance7/lambdaexpressions/ExampleNaftalin2015`

```
1 List<List<String>> listeDeListes = new ArrayList<>();
2 listeDeListes.add(Arrays.asList("a", "b", "c"));
3 listeDeListes.add(Arrays.asList("c", "d", "a"));
4 Set<String> concat = Stream.concat(listeDeListes.get(0).stream(),
5                                    listeDeListes.get(1).stream())
6                           .distinct()
7                           .collect(Collectors.toSet());
8 System.out.println(concat);
```

Affichage :

```
[a, b, c, d]
```

Le prototype de la méthode `asList` de `Arrays` est « `List<String> asList(String... a)` ». Pour rappel, l'écriture « `String...` » est une commodité d'écriture appellée varargs.

3.9 Exemple sur les pipelines des *Streams* i/i

- Même exemple avec généralisation à un nombre quelconque de listes
  - Début : obtention de la séquence de listes de la liste englobante
    Intermédiaire : transformation de chaque liste de `String` en un *stream* avec `stream de List` et insertion dans une séquence de `String`
    - `flatMap` = *flattened map*
      - Transforme un `Stream<Stream<String>>` en un `Stream<String>`

Classe `seance7/lambdaexpressions/ExampleNaftalin2015`

```
1 List<List<String>> listeDeListes = new ArrayList<>();
2 listeDeListes.add(Arrays.asList("a", "b", "c"));
3 listeDeListes.add(Arrays.asList("c", "d", "a"));
4 Set<String> flatMap = listeDeListes.stream()
5                                    .flatMap(List::stream)
6                                    .distinct()
7                                    .collect(Collectors.toSet());
8 System.out.println(flatMap);
```

4 *Streams* + Optional

- Règle 45 de « *Effective JAVA* » [Bloch, 2018] : « *Use streams judiciously* »
  - « *Overusing streams makes programs hard to read and maintain* »

Classe etudesdecas/mediathequeaveclambdasoptionaletstreams/Mediatheque

```
1  private Optional<FicheEmprunt> chercherEmprunt(final String nom, final String
       prenom, final String code) {
2    return lesEmprunts
3            .stream()
4            .filter(f -> f.correspond(
5                            lesClients.values().stream()
6                              .filter(c -> c.getNom().equals(nom) && c.
                                   getPrenom().equals(prenom))
7                              .findFirst()
8                              .orElseThrow(() -> new IllegalArgumentException(
9                                    "pas␣de␣client␣'"+nom+"␣"+prenom+"'")),
10                           lesDocuments.values().stream()
11                             .filter(d -> d.getCode().equals(code))
12                             .findFirst()
13                             .orElseThrow(() -> new IllegalArgumentException(
14                                   "pas␣de␣document␣'" + code + "'"))))
15           .findFirst();
16 }
```

Références I

Bloch, J. (2008).
Effective Java, 2nd Edition.
Addison-Wesley.

Bloch, J. (2018).
Effective Java, 3rd Edition.
Addison-Wesley.

Javadoc (JSE8).
Javadoc Tool.
http://www.oracle.com/technetwork/java/javase/documentation/index-jsp-135444.html.