

Projet de GL en JAVA

Jhotdraw

Objectifs de ce projet : Étudier la qualité logicielle, d'un logiciel de grande taille qu'on ne connaît pas

Auteur de ce projet :

→ AIT YAHIA Maya

Lien vers mon répertoire git :

https://gitlab-etu.fil.univ-lille.fr/maya.aityahia.etu/projet_gl.git

Enseignant : Mr.Cyril Ferlicot

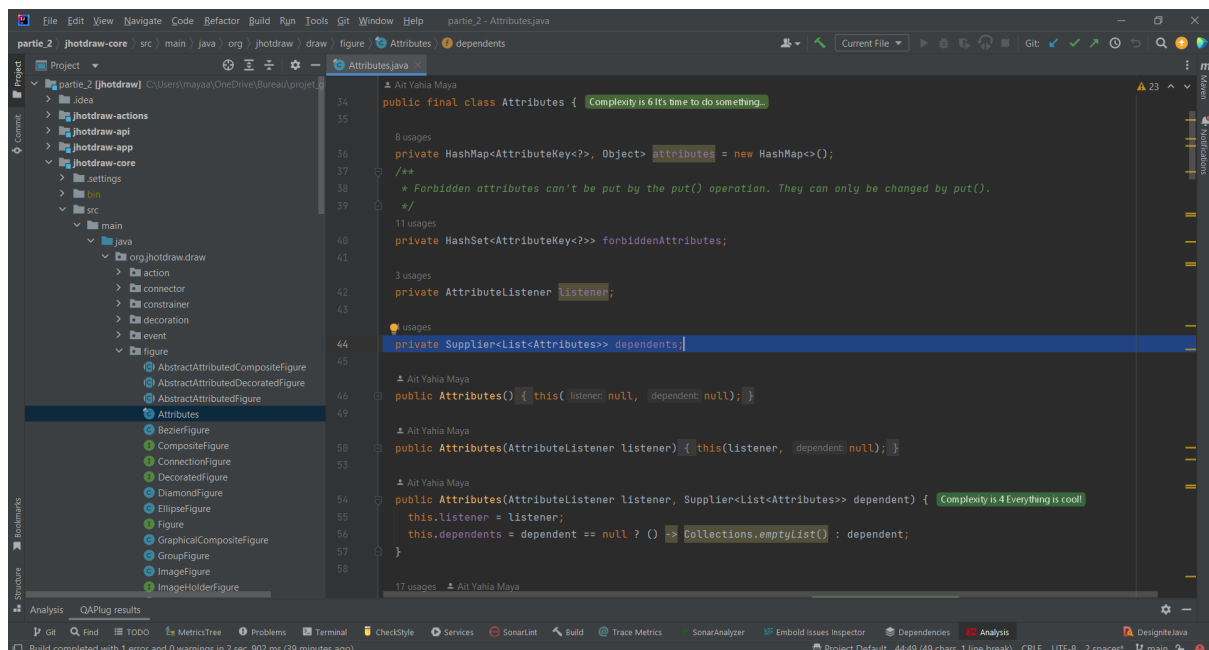
(cyril@ferlicot.me)

6- Petites modifications :

Renommer une classe, une méthode, une variable :

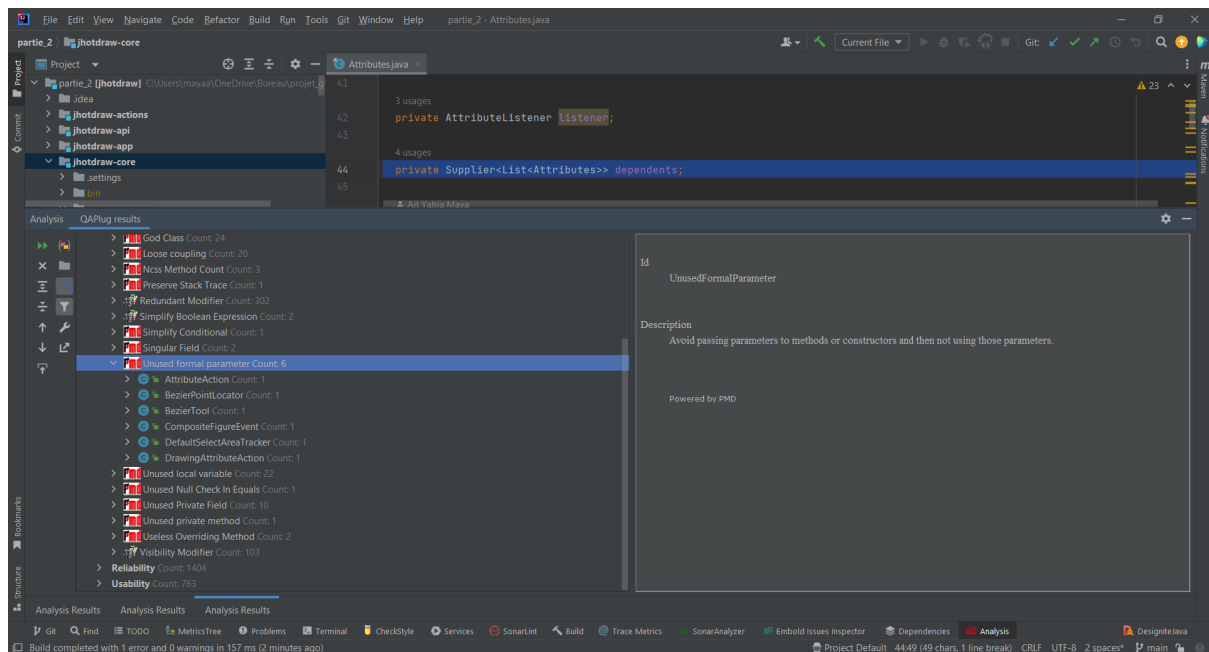
Comme on l'a vu dans la première partie l'utilisation de majuscules pour les noms d'attributs peut être considérée comme une mauvaise pratique car cela peut rendre le code difficile à lire et à comprendre, en particulier si les majuscules sont utilisées de manière inconsistante. De plus, certaines conventions de nommage utilisent des majuscules pour des éléments spécifiques du code, comme les constantes, et l'utilisation de majuscules pour les attributs pourrait causer de la confusion.

Pour cela on a remplacé le nom de l'attribut "DEPENDENT" qui se trouve dans la classe `jhotdraw-core/.../main/java/org/jhotdraw/draw/figure/Attributes.java`, par "dependents"



Changer le type ou le nombre de paramètres d'une méthode :

En analysant le projet (le package `jhotdraw-core`), on a trouvé qu'il y a 6 méthodes qui comportent des paramètres non utilisés. Cependant, quand on les supprime et qu'on essaie de tester ça ne marche pas, on a donc décidé de les laisser afin de préserver la stabilité du code.

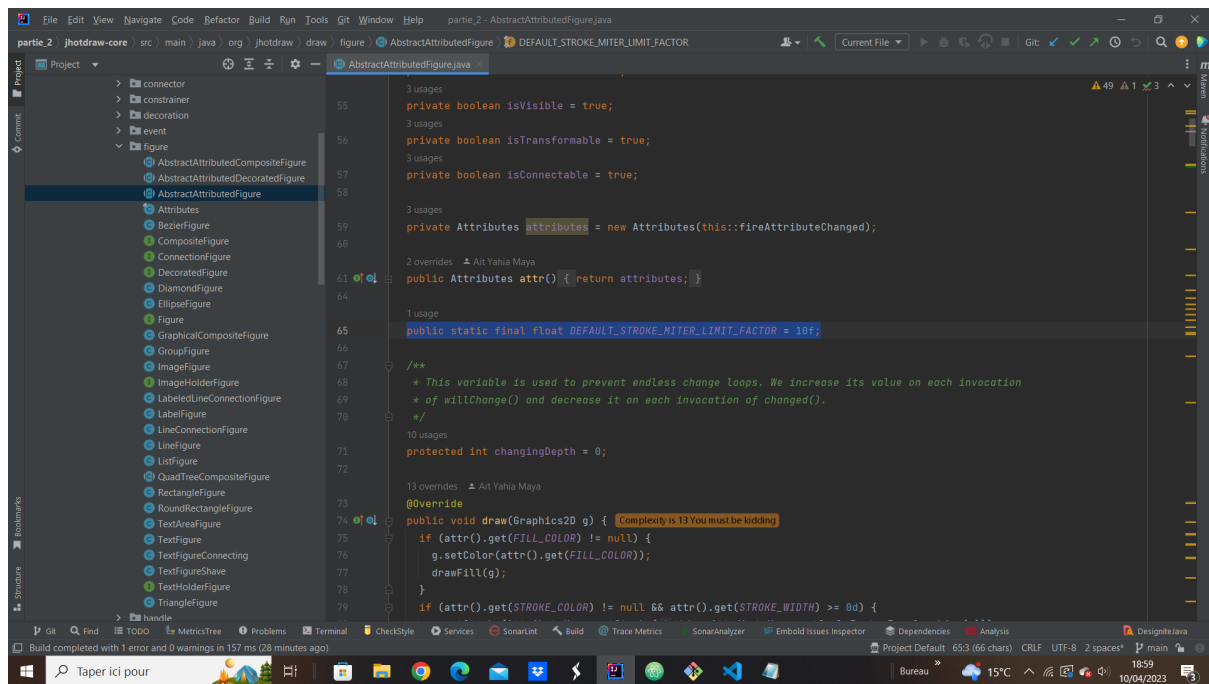


Créer des variables pour supprimer des nombres magiques :

Dans ce projet, il existe plusieurs nombres magiques, notamment 10f, qui se trouve dans la classe
 jhotdraw-core/.../main/java/org/jhotdraw/draw/figure/AbstractAttributedFigure.java.

Ce nombre peut compliquer la compréhension et la maintenance du code. C'est pourquoi on a créé une constante :

`public static final float DEFAULT_STROKE_MITER_LIMIT_FACTOR = 10f;`
 qui peut être utilisée à la place de ce nombre magique 10f.



Supprimer du code mort :

Supprimer le code mort peut aider à améliorer la maintenance du code en réduisant la complexité et en facilitant la compréhension du code restant.

Pour cela on a supprimé ce code mort identifié dans le rapport 1.

→ Les attributs non utilisés : private CompositeFigure prototype qui se trouve dans la classe : jhotdraw-core/.../main/java/org/jhotdraw/draw/action/UngroupAction.java

→ Les méthodes non utilisées : getStrokeWidth(Figure f) qui se trouve dans la classe : jhotdraw-core/.../main/java/org/jhotdraw/draw/connector/ChopEllipseConnector.java

Réorganiser une classe pour le code soit bien structuré :

Dans ce projet on remarque que toutes les classes ne sont pas bien structurées, par exemple dans la classe

jhotdraw-core/.../main/java/org/jhotdraw/draw/DefaultDrawing.java on trouve des classes privées placées avant les classes publiques.

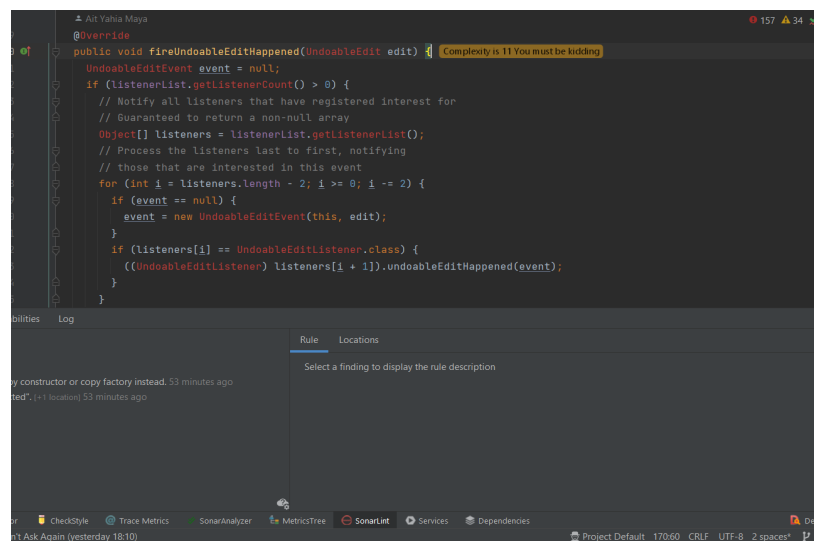
Donc on a réorganisé la classe comme demander les variables d'instance en début de classe, puis méthodes publiques et enfin méthodes privées pour garantir une compréhension claire et précise du code par les développeurs et les utilisateurs, ce

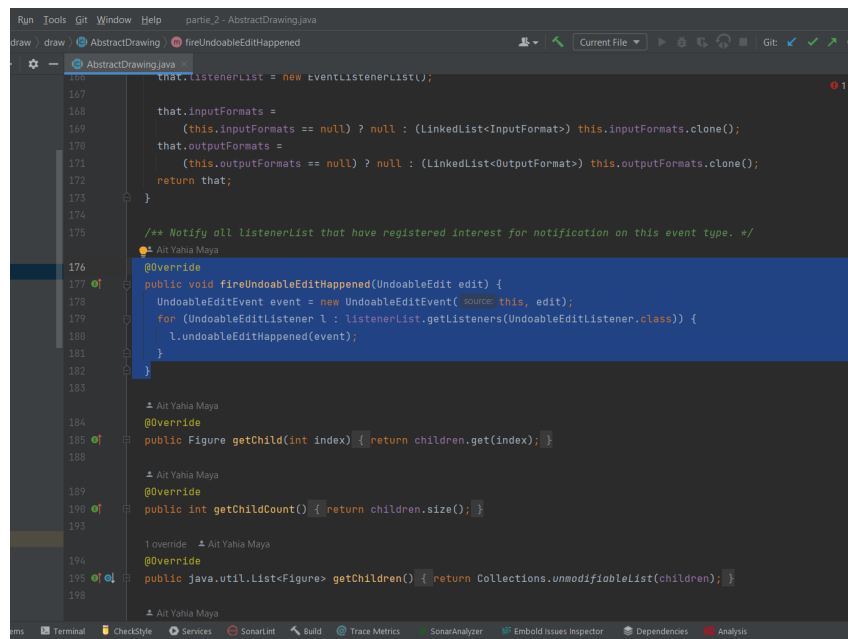
qui facilite la lisibilité et la maintenance du code et contribue à la qualité globale du logiciel

7- Moyennes modifications :

Réduire la complexité cyclomatique ou le nombre de lignes d'une méthode :

La méthode `fireUndoableEditHappend` qui se trouve dans la classe `jhotdraw-core/.../main/java/org/jhotdraw/draw/AbstractDrawing.java` a une complexité de 11, et on sait que plus une méthode a une complexité élevée, plus elle est difficile à maintenir, à tester et à déboguer. Pour cela on va simplifier la méthode en utilisant la méthode `listenerList.getListeners(UndoableEditListener.class)` pour récupérer les listeners. Cela évite la boucle et le test à l'intérieur de la boucle.





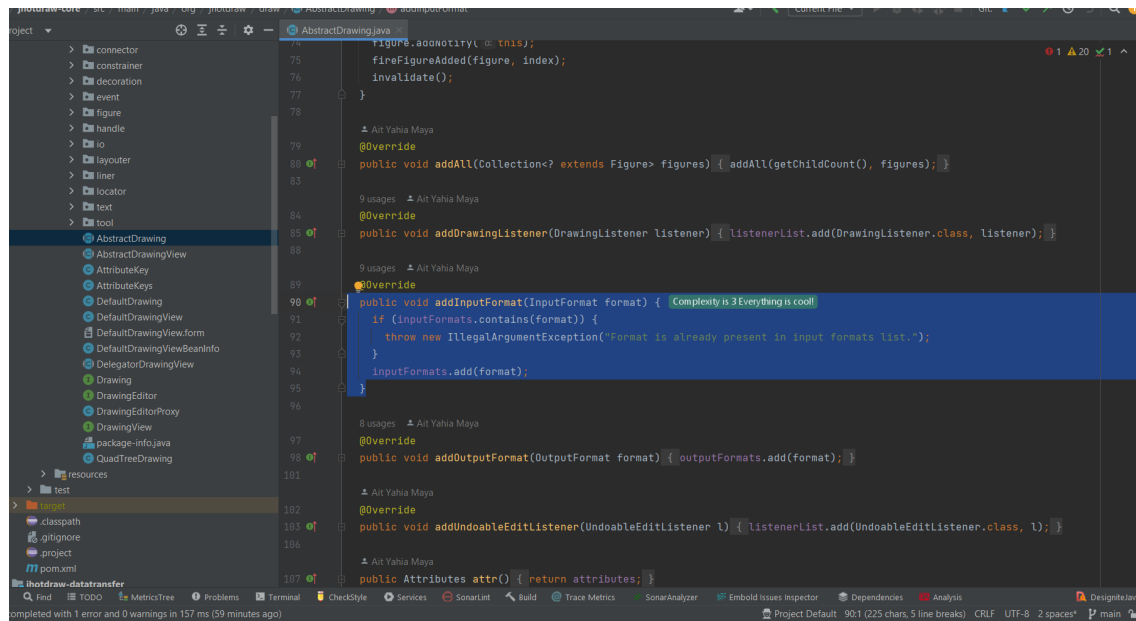
```
166 that.listenerList = new EventListenerList();
167
168 that.inputFormats =
169     (this.inputFormats == null) ? null : (LinkedList<InputFormat>) this.inputFormats.clone();
170 that.outputFormats =
171     (this.outputFormats == null) ? null : (LinkedList<OutputFormat>) this.outputFormats.clone();
172 return that;
173 }
174
175 /** Notify all listenerList that have registered interest for notification on this event type. */
176 @Override
177 public void fireUndoableEditHappened(UndoableEdit edit) {
178     UndoableEditEvent event = new UndoableEditEvent(source, this, edit);
179     for (UndoableEditListener l : listenerList.getListeners(UndoableEditListener.class)) {
180         l.undoableEditHappened(event);
181     }
182 }
183
184 @Override
185 public Figure getChild(int index) { return children.get(index); }
186
187 @Override
188 public int getChildCount() { return children.size(); }
189
190 @Override
191 public java.util.List<Figure> getChildren() { return Collections.unmodifiableList(children); }
192
193
194 @Override
195 public java.util.List<Figure> getChildren() { return Collections.unmodifiableList(children); }
196
197
198
```

Remplacer le fait qu'une méthode retourne un code d'erreur par le fait qu'elle lève une exception :

La méthode `addInputFormat(InputFormat format)` qui se trouve dans la classe `jhotdraw-core/.../main/java/org/jhotdraw/draw/AbstractDrawing.java`, ajoute un format d'entrée à la liste des formats d'entrée pris en charge par le dessin. Actuellement, si un format est déjà présent dans la liste, la méthode renvoie un code d'erreur `false` et ne modifie pas la liste.

Au lieu de retourner un code d'erreur, nous pourrions lever une exception de type `IllegalArgumentException` si le format est déjà présent dans la liste.

Avec ce changement, si un format est déjà présent dans la liste et que la méthode est appelée, elle lèvera une exception avec un message décrivant le problème. L'appelant peut alors gérer cette exception en conséquence, par exemple en affichant un message d'erreur à l'utilisateur.



Supprimer de la duplication de codes entre méthodes :

On a trouvé une duplication de code entre les classes East, West, North et South qui étendent la classe MoveAction

“jhotdraw-core/src/main/java/org/jhotdraw/action/MoveAction.java

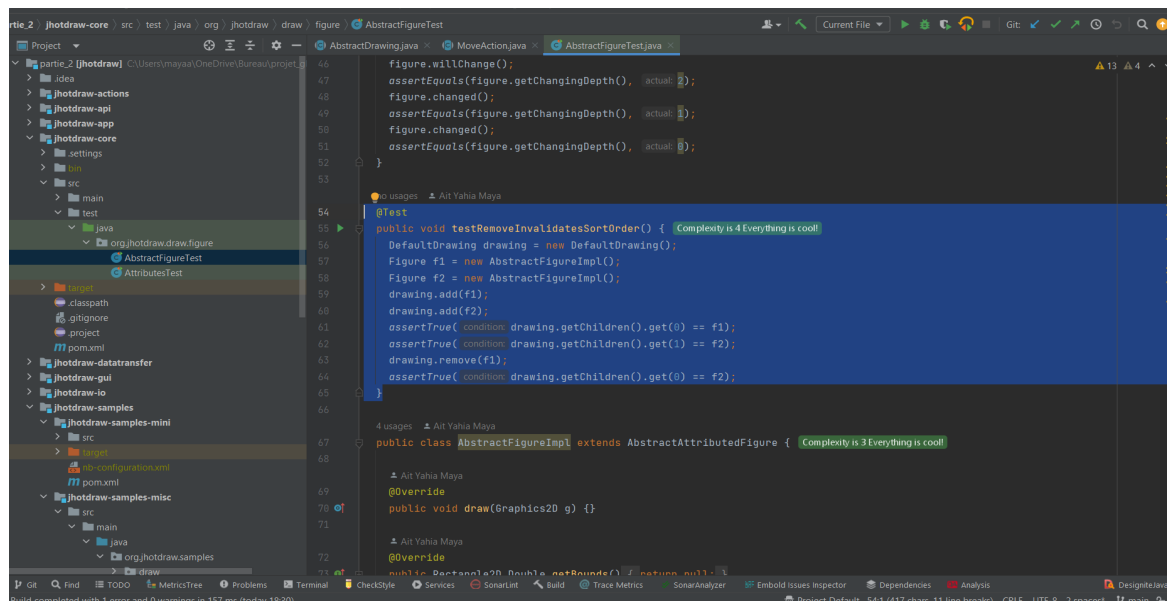
Les seules différences entre ces classes sont les valeurs des champs dx et dy passés au constructeur de la classe parente.

Pour supprimer cette duplication de code, on a utilisé un constructeur de la classe MoveAction qui prend les valeurs dx et dy en paramètres plutôt que de les définir dans chaque classe fille. Ainsi, on a défini une seule classe fille MoveDirection qui appelle le constructeur de MoveAction avec les valeurs dx et dy appropriées.

Ajouter un test pertinent :

Le package jhotdraw-core dispose de deux classes de tests : AbstractFigureTest et AttributesTest qui semblent pertinents

On ajoute un test à la classe AbstractFigureTest qui crée une instance DefaultDrawing et y ajoute deux figures. Ensuite, il supprime la première figure et vérifie si la seconde figure a été déplacée en première position dans la liste des enfants, validant ainsi que l'ordre de tri a été invalidé.



Corriger un test rouge ou orange :

Dans ce projet (la partie qu'on étudie jhotdraw-core) , il y a que deux tests qui passent correctement donc il n'y aucun test rouge ou orange à corriger.

```

[INFO] --- maven-surefire-plugin:3.0.0-M5:test (default-test) @ jhotdraw-samples-mini ---
[INFO] No tests to run.
[INFO] Reactor Summary for jhotdraw 10.0-SNAPSHOT:
[INFO]
[INFO] jhotdraw ..... SUCCESS [ 7.864 s]
[INFO] jhotdraw-utils ..... SUCCESS [ 21.271 s]
[INFO] jhotdraw-datatransfer ..... SUCCESS [ 0.816 s]
[INFO] jhotdraw-api ..... SUCCESS [ 0.788 s]
[INFO] jhotdraw-core ..... SUCCESS [ 31.877 s]
[INFO] jhotdraw-actions ..... SUCCESS [ 2.692 s]
[INFO] jhotdraw-gui ..... SUCCESS [ 11.872 s]
[INFO] jhotdraw-app ..... SUCCESS [ 4.022 s]
[INFO] jhotdraw-vml ..... SUCCESS [ 0.497 s]
[INFO] jhotdraw-io ..... SUCCESS [ 7.323 s]
[INFO] jhotdraw-samples ..... SUCCESS [ 0.085 s]
[INFO] jhotdraw-samples-mini ..... SUCCESS [ 14.290 s]
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 01:48 min
[INFO] Finished at: 2023-04-10T17:48:27+02:00
[INFO]

```


8- Grandes modifications :

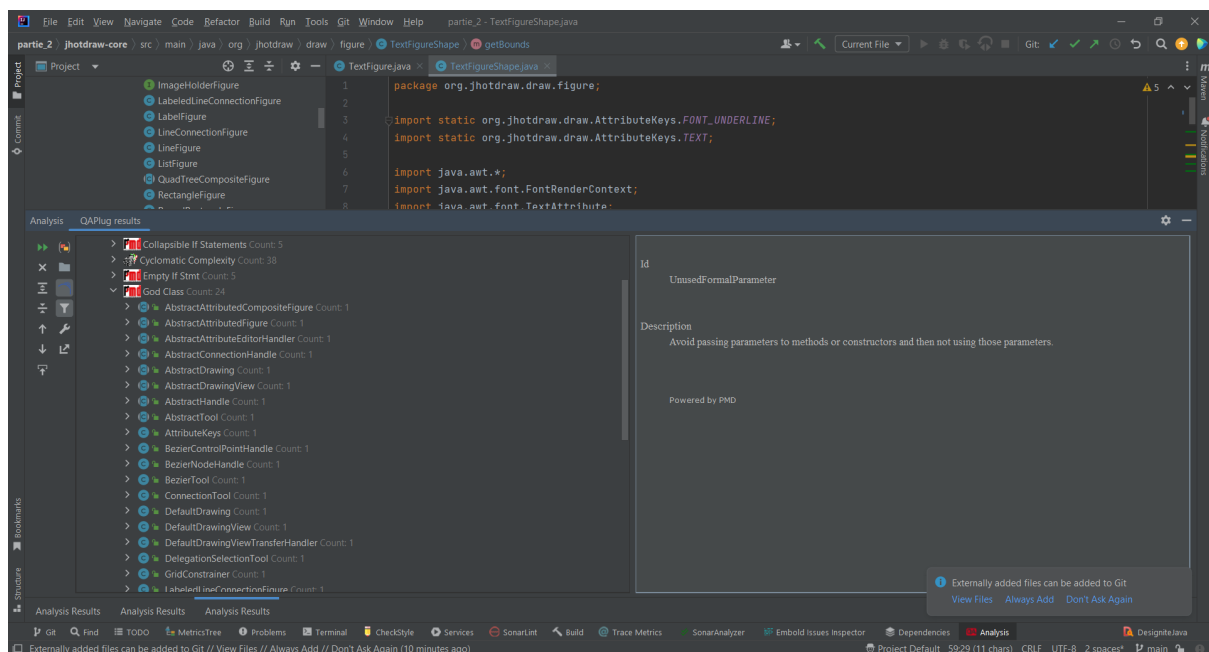
Décomposer une god classe :

Une "god class" est une classe qui a trop de responsabilités et de fonctionnalités, ce qui la rend difficile à comprendre, à maintenir et à modifier. Pour cela qu'il faut la décomposer.

Dans la partie jhotdraw-core il y a 25 god class, parmi eux on trouve la classe textFigure , on remarque qu'on peut la décomposer à partir des fonctionnalités des différentes méthodes :

- La partie "Drawing" (méthodes : drawStroke(), drawFill(), drawText()) a été déplacé dans les classes nodeFigure et labelFigure qui héritent de textFigure
- La partie "Shape and Bounds" (méthodes : transform(), setBounds, figureContains(), getTextLayout(), getBounds(), getBaseline(), getFigureDrawinfArea(), restoreTransformTo(), getTransformRestoreData(), getText(), getFont()) a été déplacé dans la classe testFigureShape
- La partie "Connecting" (méthodes : invalidate() , validate(), clone(), isTextOverFlow()) a été déplacé dans la classe testFigureConnecting

Ainsi après une nouvelle analyse des God class, on constate qu'on a maintenant 24 god class au lieu de 25 trouvé avant de décomposer la classe textFigure

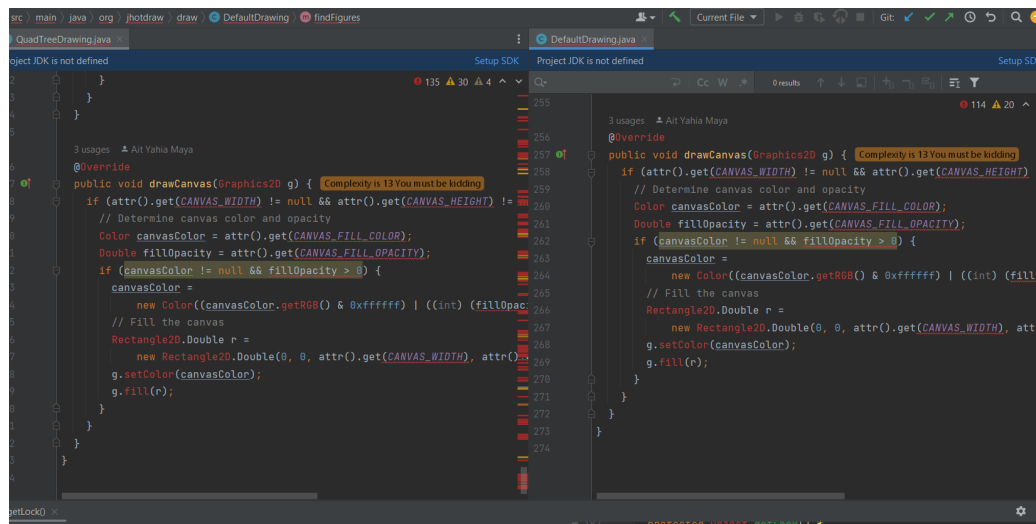


Ajouter une super classe pour supprimer des méthodes dupliquées :

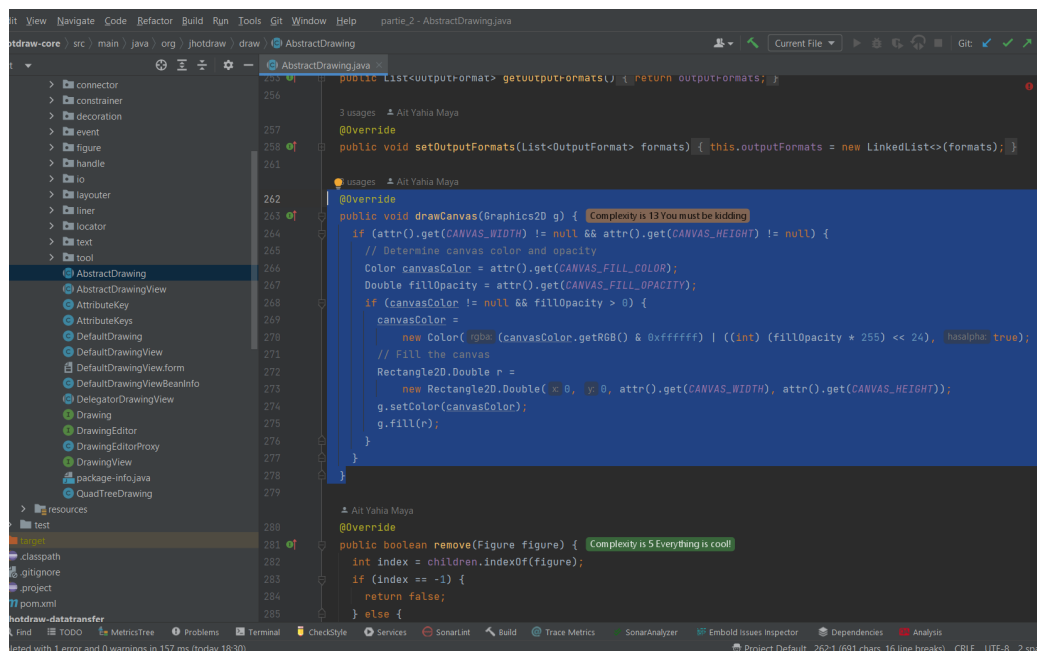
Comme l'a identifié dans le premier rapport, la méthode drawCanvas() est présente dans ces deux classes :

jhotdraw-core/src/main/java/org/jhotdraw/draw/QuadTreeDrawing.java et
jhotdraw-core/src/main/java/org/jhotdraw/draw/DefaultDrawing.java

Pour résoudre ce problème, on a supprimé cette méthode dupliquée en la mettant dans la super-classe AbstractDrawing qu'elles étendent toutes les deux.



```
public void drawCanvas(Graphics2D g) {  
    if (attr().get(CANVAS_WIDTH) != null && attr().get(CANVAS_HEIGHT) != null) {  
        // Determine canvas color and opacity  
        Color canvasColor = attr().get(CANVAS_FILL_COLOR);  
        Double fillOpacity = attr().get(CANVAS_FILL_OPACITY);  
        if (canvasColor != null && fillOpacity > 0) {  
            canvasColor =  
                new Color((canvasColor.getRGB() & 0xffffffff) | ((int) (fillOpacity * 255) << 24), hasAlpha: true);  
            // Fill the canvas  
            Rectangle2D.Double r =  
                new Rectangle2D.Double(0, 0, attr().get(CANVAS_WIDTH), attr().get(CANVAS_HEIGHT));  
            g.setColor(canvasColor);  
            g.fillRect(r);  
        }  
    }  
}
```



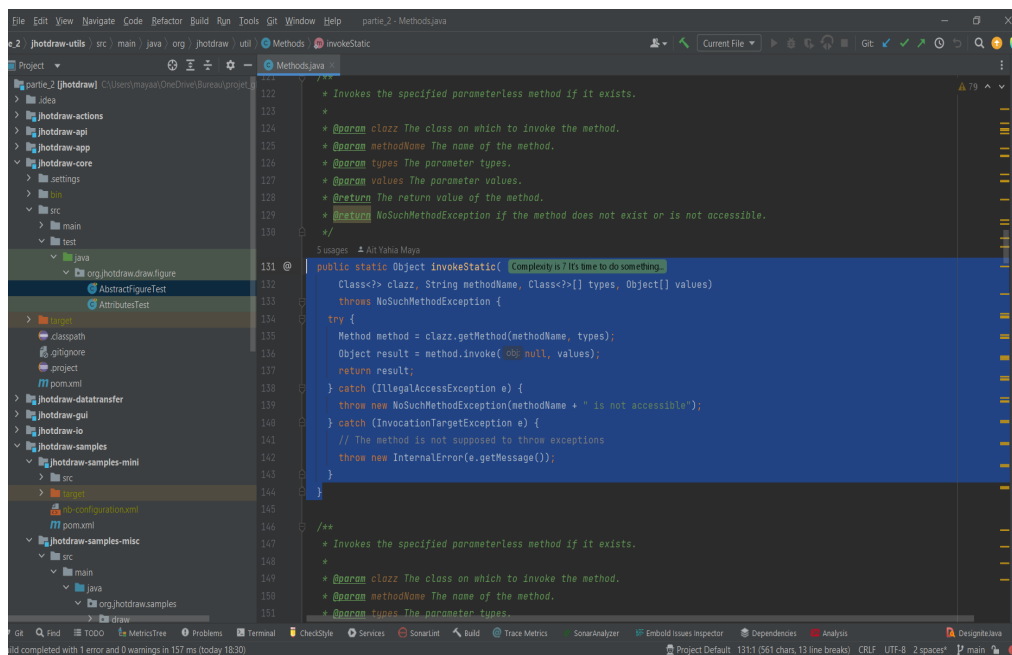
```
public void drawCanvas(Graphics2D g) {  
    if (attr().get(CANVAS_WIDTH) != null && attr().get(CANVAS_HEIGHT) != null) {  
        // Determine canvas color and opacity  
        Color canvasColor = attr().get(CANVAS_FILL_COLOR);  
        Double fillOpacity = attr().get(CANVAS_FILL_OPACITY);  
        if (canvasColor != null && fillOpacity > 0) {  
            canvasColor =  
                new Color((canvasColor.getRGB() & 0xffffffff) | ((int) (fillOpacity * 255) << 24), hasAlpha: true);  
            // Fill the canvas  
            Rectangle2D.Double r =  
                new Rectangle2D.Double(0, 0, attr().get(CANVAS_WIDTH), attr().get(CANVAS_HEIGHT));  
            g.setColor(canvasColor);  
            g.fillRect(r);  
        }  
    }  
}
```

Supprimer des classes static :

La suppression de classes statiques dépend de l'utilisation de ces classes dans le reste du code, parfois ca peut affecter la sémantique du code, donc il est important de tester soigneusement toutes les modifications apportées au code pour éviter tout effet secondaire indésirable

Par exemple, on a essayé de supprimer la classe invokeStatic qui se trouve dans jhotdraw-core/src/main/java/org/jhotdraw/util/Methods.java

Et donc les tests ne passent plus car ils dépendent de la présence de cette méthode.



```
122  * Invokes the specified parameterless method if it exists.
123  *
124  * @param clazz The class on which to invoke the method.
125  * @param methodName The name of the method.
126  * @param types The parameter types.
127  * @param values The parameter values.
128  * @return The return value of the method.
129  * @throws NoSuchMethodException if the method does not exist or is not accessible.
130  */
131  public static Object invokeStatic( /* Complexity is 7/Its time to do something.
132      Class<?> clazz, String methodName, Class<?>[] types, Object[] values)
133      throws NoSuchMethodException {
134      try {
135          Method method = clazz.getMethod(methodName, types);
136          Object result = method.invoke(null, values);
137          return result;
138      } catch (IllegalAccessException e) {
139          throw new NoSuchMethodException(methodName + " is not accessible");
140      } catch (InvocationTargetException e) {
141          // The method is not supposed to throw exceptions
142          throw new InternalError(e.getMessage());
143      }
144  }
145
146  /**
147   * Invokes the specified parameterless method if it exists.
148   *
149   * @param clazz The class on which to invoke the method.
150   * @param methodName The name of the method.
151   * @param types The parameter types.
```

```

MINGW64/C:/Users/mayaa/OneDrive/Bureau/projet_g1/partie_2
method org.jhotdraw.util.Methods.invokeStatic(java.lang.String,java.lang.String,java.lang.Class<?>[],java.lang.Object[]) is not applicable
(argument mismatch; java.lang.Class<capture#5 of ?> cannot be converted to java.lang.String)
[INFO] 3 errors
[INFO] -----
[INFO] Reactor Summary for jhotdraw 10.0-SNAPSHOT:
[INFO] jhotdraw ..... SUCCESS [ 7.504 s]
[INFO] jhotdraw-utils ..... FAILURE [ 11.148 s]
[INFO] jhotdraw-datatransfer ..... SKIPPED
[INFO] jhotdraw-api ..... SKIPPED
[INFO] jhotdraw-core ..... SKIPPED
[INFO] jhotdraw-actions ..... SKIPPED
[INFO] jhotdraw-gui ..... SKIPPED
[INFO] jhotdraw-app ..... SKIPPED
[INFO] jhotdraw-vml ..... SKIPPED
[INFO] jhotdraw-io ..... SKIPPED
[INFO] jhotdraw-samples ..... SKIPPED
[INFO] jhotdraw-samples-misc ..... SKIPPED
[INFO] jhotdraw-samples-mini ..... SKIPPED
[INFO] -----
[INFO] BUILD FAILURE
[INFO] -----
[INFO] Total time: 19.008 s
[INFO] Finished at: 2023-04-10T21:13:12+02:00
[INFO] -----
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-compiler-plugin:3.10.1:compile (default-compile) on project jhotdraw-utils: Compilation failure: Compilation failure:
[ERROR] /C:/Users/mayaa/OneDrive/Bureau/projet_g1/partie_2/jhotdraw-utils/src/main/java/org/jhotdraw/util/Methods.java:[118,12] no suitable method found for invokeStatic(java.lang.C
[ERROR] g.String,java.lang.Class<?>[],java.lang.Object[])
[ERROR] method org.jhotdraw.util.Methods.invokeStatic(java.lang.Class<?>,java.lang.String,java.lang.Class<?>[],java.lang.Object) is not applicable
[ERROR] (argument mismatch; java.lang.Class<?>[] cannot be converted to java.lang.Class<?>)
[ERROR] method org.jhotdraw.util.Methods.invokeStatic(java.lang.String,java.lang.String,java.lang.Class<?>[],java.lang.Object[]) is not applicable
[ERROR] (argument mismatch; java.lang.Class<capture#1 of ?> cannot be converted to java.lang.String)
[ERROR] /C:/Users/mayaa/OneDrive/Bureau/projet_g1/partie_2/jhotdraw-utils/src/main/java/org/jhotdraw/util/Methods.java:[146,14] no suitable method found for invokeStatic(java.lang.C
[ERROR] g.String,java.lang.Class<?>[],java.lang.Object[])
[ERROR] method org.jhotdraw.util.Methods.invokeStatic(java.lang.Class<?>,java.lang.String,java.lang.Class<?>[],java.lang.Object) is not applicable
[ERROR] (argument mismatch; java.lang.Class<?>[] cannot be converted to java.lang.Class<?>)
[ERROR] method org.jhotdraw.util.Methods.invokeStatic(java.lang.String,java.lang.String,java.lang.Class<?>[],java.lang.Object[]) is not applicable
[ERROR] (argument mismatch; java.lang.Class<capture#2 of ?> cannot be converted to java.lang.String)
[ERROR] /C:/Users/mayaa/OneDrive/Bureau/projet_g1/partie_2/jhotdraw-utils/src/main/java/org/jhotdraw/util/Methods.java:[166,14] no suitable method found for invokeStatic(java.lang.C
[ERROR] g.String,java.lang.Class<?>[],java.lang.Object[])
[ERROR] method org.jhotdraw.util.Methods.invokeStatic(java.lang.Class<?>,java.lang.String,java.lang.Class<?>[],java.lang.Object) is not applicable
[ERROR] (argument mismatch; java.lang.Class<?>[] cannot be converted to java.lang.Class<?>)
[ERROR] method org.jhotdraw.util.Methods.invokeStatic(java.lang.String,java.lang.String,java.lang.Class<?>[],java.lang.Object[]) is not applicable
[ERROR] (argument mismatch; java.lang.Class<capture#3 of ?> cannot be converted to java.lang.String)
[ERROR] -> [Help 1]
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles:
[ERROR] [Help 1] http://wiki.apache.org/confluence/display/MAVEN/MojoFailureException
[ERROR]
[ERROR] After correcting the problems, you can resume the build with the command
[ERROR] mvn <args> -rf :jhotdraw-utils
mayaa@LAPTOP-C53DMTC4 MINGW64 ~/OneDrive/Bureau/projet_g1/partie_2 (main)
$

```

Fusionner des classes :

On a fusionné les deux classes FloatingTextField et FloatingTextArea qui se trouvent dans le paquetage jhotdraw-core/src/main/java/org/jhotdraw/text en une seule classe FloatingText.

Cela évite de séparer les classes qui font les mêmes fonctionnalités même si c'est pour réduire la taille et la complexité d'une seule classe.

Supprimer des packages contenant peu de classes :

La suppression des packages peut :

- Simplifier et réduire la complexité du code en réduisant le nombre de classes et en simplifiant la structure globale du projet
- Améliorer les performances car les paquetages contenant peu de classes peuvent ralentir le temps de compilation et de chargement du projet
- Réduire la quantité de code. Cela rendra la maintenance du code plus facile et plus efficace.

Comme le package print dispose d'une seule classe qui est DrawingPageable.java, on l'a supprimé et déplacé la classe dans un autre package (locator)

