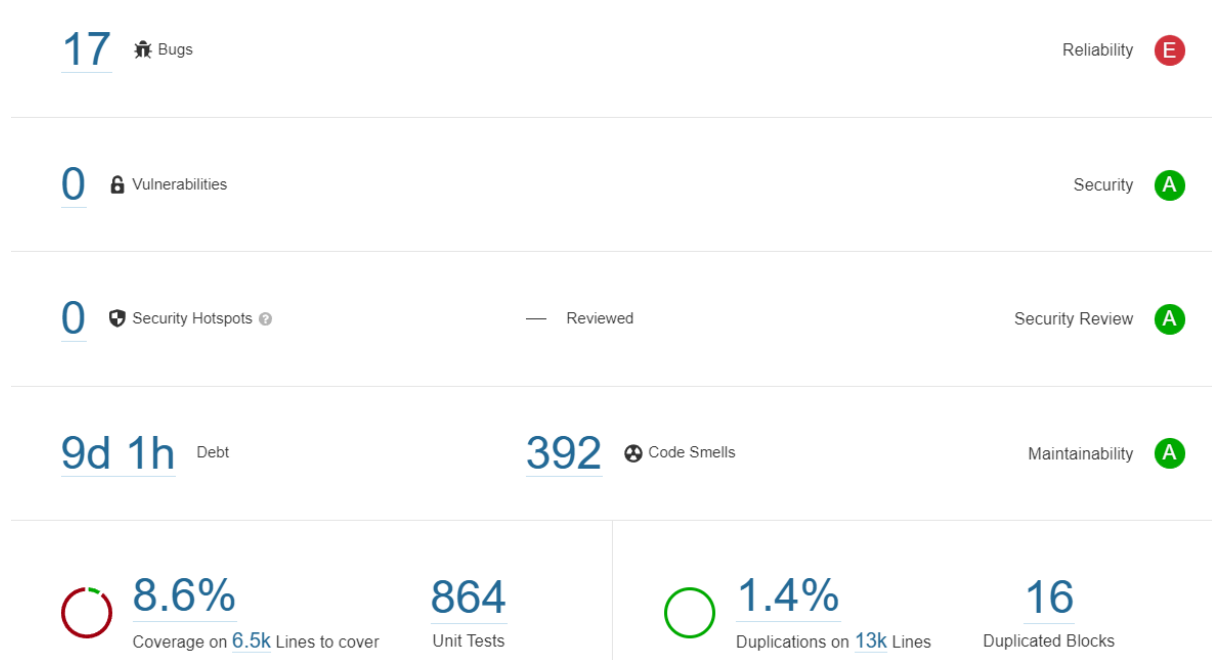


Projet Part 2

Analyse de base :



Code Review

Commit : réduction complexité cyclomatique dans UtilDeserialisationJson.
Réduction de la complexité cyclomatique (qui était à 10) en effectuant de la factorisation de code et la séparation de responsabilité (création de nouvelles fonctions pour déléguer certaines tâches).

```
public static <T> T deserialiserObjet(JsonReader jr, Class<T> clazz) throws Exception { Complexity is 10
```

qui passe à 5:

```
public static <T> T deserialiserObjet(JsonReader jr, Class<T> clazz) throws Exception { Complexity is 5 Everything is cool!
```

avec ajout de méthodes annexes:

Commit : Factorisation de code de

`timeBagOfPrimitivesReflectionStreaming`.

réduction de la complexité cognitive de `timeBagOfPrimitivesReflectionStreaming`.

```
public void timeBagOfPrimitivesReflectionStreaming(int reps) throws Exception { Complexity is 11 You must be kidding
```

qui pass à 3:

```
public void timeBagOfPrimitivesReflectionStreaming(int reps) throws Exception { Complexity is 3 Everything is cool!
```

Commit : suppression de la fonction de timeBagOfPrimitivesStreaming
suppression de la fonction de `timeBagOfPrimitivesStreaming` qui n'est utilisé nulle part dans le code du projet et qui avait une complexité de 11.

Commit : gestion duplication de code dans ConstructorConstructor.java
gestion de la duplication de code dans la classe ConstructorConstructor.java

ConstructorConstructor.java 103-112	ConstructorConstructor.java 115-124
<pre>@SuppressWarnings("unchecked") // type: final InstanceCreator<T> typeCreator = if (typeCreator != null) { return new ObjectConstructor<T>() { @Override public T construct() { return typeCreator.createInstance(); } }; }</pre>	<pre>@SuppressWarnings("unchecked") // types final InstanceCreator<T> rawTypeCreator = if (rawTypeCreator != null) { return new ObjectConstructor<T>() { @Override public T construct() { return rawTypeCreator.createInstance(); } }; }</pre>

en essayant de modifier le code pour limiter la duplication de code, on s'est rendu compte que la version java 7 qu'utilisait le projet ne prend pas en charge les fonctions anonymes (lambda), donc j'ai résolu le problème.

```
gson-parent [compile]: At 04/04/2024 23:46 with 6 sec, 847 ms
com.google.code.gson:gson:jar:2.10.2-SNAPSHOT 2 sec, 817 ms
compile 1 error 2 sec, 208 ms
ConstructorConstructor.java gson\src\main\java\com\googl
lambda expressions are not supported in -source 7 :157
```

et il n'y a plus de duplication de ce bloc de code dans la classe ConstructorConstructor.

Commit : utilisation de private dans une méthode privée (rédundance)

```
private static final <T> void assertIterationOrder(Iterable<T> actual, T... expected) {
```

corrigée :

```
private static <T> void assertIterationOrder(Iterable<T> actual, T... expected) {
```

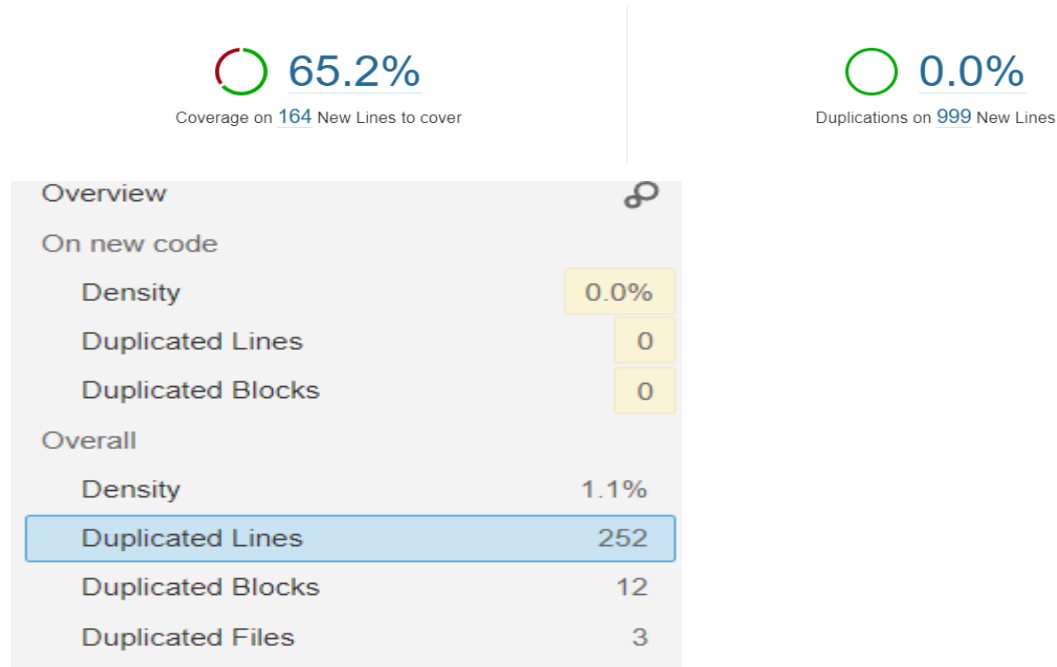
Commit: utilisation de assertThrows pour ne pas avoir un bloc de catch vide

```
try {
    keySet.add("c");
    fail();
} catch (UnsupportedOperationException e) {
}
```

Commit: utilisation de String.repeat(...) au lieu d'un for (Java 11 requis)

utilisation de String.repeat(times) dans JsonParserTest mais une version java 11 où ultérieur est requise pour pouvoir utiliser String.repeat(..).

Duplication line and block on new code and coverage:



Essais Réduction nombre de break dans la boucle:

j'ai essayé de réduire le nombre de break dans cette fonction et la complexité par la même occasion, mais après les changements les tests s'exécutent indéfiniment, donc j'ai dû enlever les changements et laissé comme c'était.

```
private static Type resolve( Complexity is 43 Bloody hell...  
  
    if (toResolve == typeVariable) {  
1 break;  
    }  
  
    } else if (toResolve instanceof Class && ((Class<?>) toResolve).isArray()) {  
        Class<?> original = (Class<?>) toResolve;  
        Type componentType = original.getComponentType();  
        Type newComponentType =  
            resolve(context, contextRawType, componentType, visitedTypeVariables);  
        toResolve = equal(componentType, newComponentType) ? original : arrayOf(newComponentType);  
2 break;  
  
    } else if (toResolve instanceof GenericArrayType) {  
        GenericArrayType original = (GenericArrayType) toResolve;  
        Type componentType = original.getGenericComponentType();  
        Type newComponentType =  
            resolve(context, contextRawType, componentType, visitedTypeVariables);  
        toResolve = equal(componentType, newComponentType) ? original : arrayOf(newComponentType);  
3 break;  
  
    } else if (toResolve instanceof ParameterizedType) {  
        ParameterizedType original = (ParameterizedType) toResolve;  
        Type ownerType = original.getOwnerType();  
        Type newOwnerType = resolve(context, contextRawType, ownerType, visitedTypeVariables);
```