

R5.A8.D7 : Qualité de Développement Feuille TD-TP n° 1

Révisions sur les tests unitaires Java, IntelliJ, Gradle, JUnit, JAssert, jacoco, mockito

Objectifs :

- 1.- Reprise en main de l'outil de développement utilisé : IntelliJ + Java
- 2.- Révision des test unitaires vus dans la ressource R4.02-Qualité de développement

Sujet :

Il s'agit de développer un sous-ensemble d'une classe simple, nommée `Calculator`, réalisant quelques opérations basiques sur des éléments de type primitifs (par exemple, int, double) et personnalisés (par exemple rationnels).

Le développement de la classe sera accompagné des tests unitaires associés.

Cette feuille de TD-TP a pour but, non pas de développer cette classe de manière exhaustive, mais de se remémorer les acquis du module [R4.02-Qualité de développement](#) consacré aux tests unitaires et Intégration Continue utilisant JUnit, JAssert, Jacoco et Mockito, Gradle, Git/Github lors d'un développement en Java avec l'IDE IntelliJ.

Ressources à votre disposition :

- L'archive `junit5-jupiter-starter-gradle.zip`
C'est un projet 'Modèle' minimal pour le développement en Java avec IntelliJ et gradle. La fonction `main()` de son unique classe `Main` affiche « Hello world ».
Il devra être configuré pour le développement demandé.
- Le fichier `build.yml`, pour l'automatisation de tests sous la forme d'action Github. Il sera complété lors de la séance.

Préparation du travail

1.- Création du dossier consacré aux TDs et TP de cette ressource (R5.A.08 – R5.D.07)

Dans le dossier de votre espace réseau destiné à vos travaux pratiques, créer un sous-dossier destiné à recevoir tous les TP de la présente ressource. Nous l'appellerons, par exemple : `r5.A08.D07`

Dans ce dossier, créer un dossier `tdtp1`.

2.- Installation des fichiers

- a. Télécharger l'archive `junit5-jupiter-starter-gradle.zip` disponible sur eLearn. Décompresser l'archive.
- b. Déposer le dossier décompressé dans `r5.A08.D07\tdtp1` puis supprimer l'archive `.zip` téléchargée.
- c. Changer le nom du dossier/projet → `Calculator`
- d. Lancer IntelliJ, ouvrir le projet `Calculator`
- e. Si un JDK est associé par défaut à votre projet, Compiler, puis exécuter `main()`. Si ce n'est pas le cas, patienter jusqu'à l'étape suivante.

Vous êtes prêt à passer à l'étape suivante.

3.- Vérifier qu'un JDK est bien associé au projet IntelliJ – Configurer la variable d'environnement **JAVA_HOME**

- a. Dans IntelliJ, dans le menu **File/Project Structure/Project Settings/Project**, identifier la version du JDK présente dans la rubrique **SDK**. Elle est définie par un Nom (Name) et un chemin d'accès (JDK home path)

Exemple :

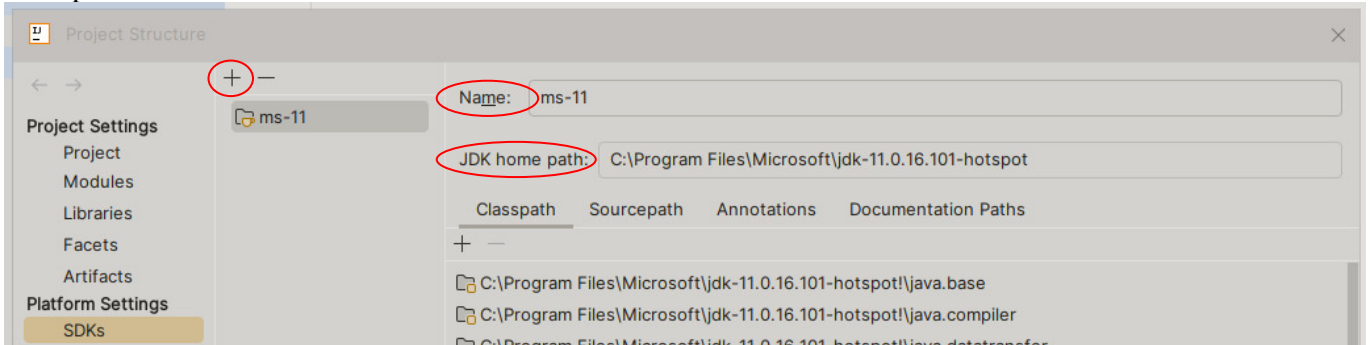


Figure 1 : JDK utilisé par défaut par IntelliJ

La Figure 1 montre une liste avec un seul JDK :

- Nom : ms-11

- Dossier d'installation : **C:\Program Files\Microsoft\jdk-11.0.16-hotspot**

C'est le JDK installé par défaut sur les bureaux virtuels de l'IUT. Vérifier que les fichiers sont bien présents sur la machine, sur **C:\Program Files\Microsoft**

Si aucun SDK n'est associé au projet, associer le JDK cité ci-dessus. S'il n'est pas disponible, télécharger la version de votre choix, grâce au bouton '+' situé au-dessus de la liste. Nous vous conseillons le JDK Amazon corretto 18.0.2 – Langage level 18).

Maintenant que vous disposez d'un JDK associé à votre projet, Compiler, puis exécuter **main()**.

Dans tous les cas, noter le chemin du **dossier d'installation** du JDK utilisé par votre IDE, vous en aurez besoin.

- b. Quitter IntelliJ
- c. A partir du Panneau de Configuration de Windows, créer une variable d'environnement **JAVA_HOME** liée à votre compte et préciser le chemin d'accès au dossier d'installation du JDK noté au point précédent.
- d. Lancer IntelliJ
- e. Dans IntelliJ, à partir du **Main menu**, puis dans le menu **Settings /Build, Execution, Deployment/Build Tools/Gradle**, renseigner la rubrique **Gradle JVM** avec la valeur **JAVA_HOME**.
- f. Sauvegarder.

Vous êtes prêt à passer à l'étape suivante.

4.- Préparer la structure du code (main et test)

- Dans `src/main/java`, créer un package java nommé `calculator`¹
- Déplacer la classe `Main` dans le package `calculator`

Dans le package `com.nomChoisi.calculator`

- Créer une classe `Calculator`
- Créer la classe de test associée (`CalculatorTest`)². Utiliser JUnit5.

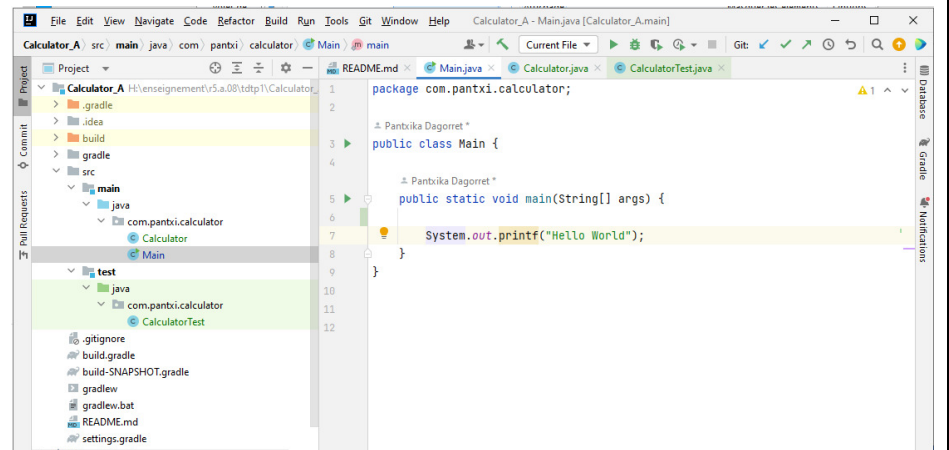


Figure 2 : Projet Calculator structuré, minimal

5.- Préparer la gestion des versions de votre projet (à l'aide de git et github)

- Depuis l'explorateur de fichiers de Windows, à la racine du dossier du projet Calculator, à l'aide de gitBash, créer un dépôt local (`git`)
- Créer un dépôt distant associé (`github`)
- Sur le dépôt local (`git`), engager le projet minimal (`git add...` puis `git commit...`) puis le pousser sur le dépôt distant (`git push origin main`)

Vous êtes prêt à travailler !

¹ Pour rappel, la structure du code et noms des packages en Java est la suivante :

- Le code source de l'application se trouve dans le dossier `src/main/java`
- Le code source des tests se trouve dans le dossier `src/test/java`
- La pratique de nommage des **packages** est la suivante :
`com.nomEntreprise.nomPackage` ou bien `com.nomProgrammeur.nomPackage`
Par exemple : `com.pantxi.calculator`

² Utiliser le raccourci : `Ctrl+Shift+T` après avoir positionné le curseur à l'intérieur de la classe. La liste des raccourcis de l'IDE IntelliJ est disponible sur : https://www.jetbrains.com/help/idea/reference-keymap-win-default.html#top_shortcuts

Travail à faire

6.- Coder et tester quelques opérations simples sur des types primitifs

a. Pour chacune des méthodes dont les signatures suivent :

```
public int add(int opG, int opD) ;  
public int divide(int opG, int opD) ;
```

- écrire sa définition, dans un premier temps sans tenir compte des possibles erreurs d'exécution pouvant advenir en raison de paramètres inappropriés
- écrire le/les test(s) associé(s) en utilisant la bibliothèque **jAssert** 3.4.5
- exécuter le(s) test(s) et visualiser le résultat sur la fenêtre dédiée de l'IDE
- le résultat est également sauvegardé, au format xml dans nomDuProjet/build/test-results/test/, et au format html dans le dossier nomDuProjet/build/reports/tests/
- isoler l'ajout de la méthode dans une nouvelle version, dans le dépôt local puis pousser vers le dépôt distant.

Rappel : Test individuel – Batterie de tests

Les tests peuvent être lancés de manière :

- individuelle, via le menu contextuel associé à chaque méthode de test (Figure 3) :

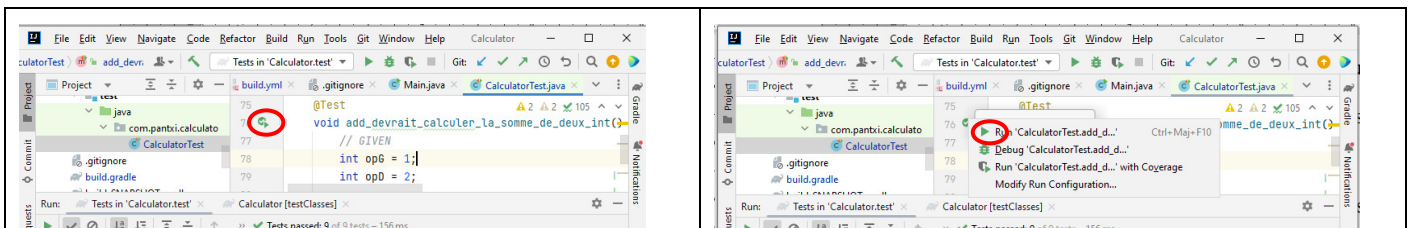


Figure 3 : Exécution d'un test individuel

- ou collective (tous simultanément)

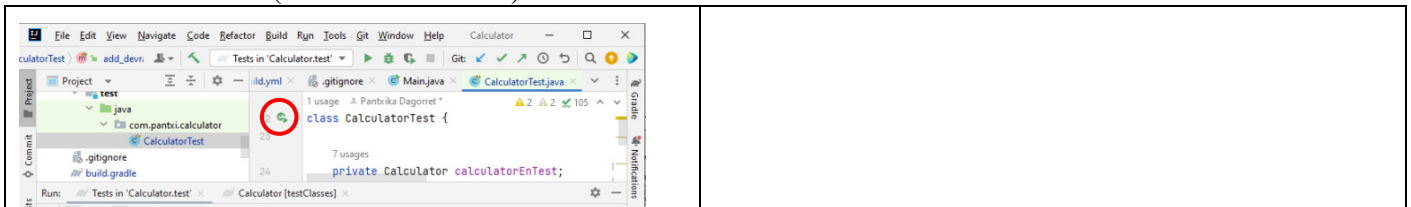


Figure 4 : Exécution collective de tous les tests de la classe (1)

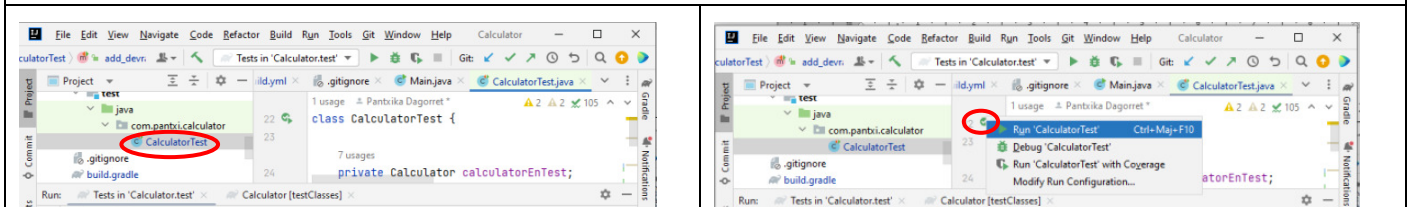


Figure 5 : Exécution collective de tous les tests de la classe (2)

³ Etendre les **dépendances** du fichier **build.gradle** avec la dernière version (3.26.3) de **assert-core**, à partir de l'url : <https://mvnrepository.com/artifact/org.assertj/assertj-core>

⁴ Les méthodes à importer pour écrire les assertions AssertJ appartiennent à la classe **Assertions**. Elles sont dans le package **org.assertj.core.api.Assertions** : <https://www.javadoc.io/static/org.assertj/assertj-core/3.26.3/org/assertj/core/api/Assertions.html>
<https://assertj.github.io/doc/#assertj-core-assertions-guide>

⁵ Appliquer les bonnes pratiques vues en R4.02 : bien nommer les tests, un test par situation à tester, notation snake_case, utiliser des import static pour les méthodes statiques, délimiter les zones du test : zones **GIVEN WHEN THEN**, ...

7.- Simplification d'un test : méthodes de montage / démontage

Les méthodes exécutées avant / après chaque test, et appelées méthodes **de montage** (set up) ou **démontage** (tear down), permettent de diminuer le nombre de lignes de chaque test et donc de le rendre plus lisible.

- a. Si cela est pertinent, utiliser les annotations `@BeforeEach` et `@AfterEach` de JUnit⁶ pour alléger chaque test avec les actions d'initialisation / nettoyage communes à tous les tests.
- b. Enregistrer, versionner.

8.- Montage et démontage - Cas des méthodes statiques

Il existe des situations où les méthodes écrites / testées sont statiques. Cela pourrait être le cas des méthodes `add()` et `divide()`.

- a. Modifier les méthodes et les tests en conséquence.
- b. Enregistrer, versionner.

⁶ JUnit : <https://junit.org/junit5/docs/current/user-guide/#overview-getting-started>

9.- Écrire des tests paramétrés

a. Pour la méthode suivante :

```
public int add(int opG, int opD) ;
```

– A l’aide de l’annotation `@ParameterizedTest` de JUnit⁷, ajouter un (nouveau) test paramétré puis exécuter ce test sur l’ensemble de valeurs simples ci-dessous, listées dans l’ordre opG, opD,

– ResultatAttendu :

0,	1,	1
1,	2,	3
-2,	2,	0
0,	0,	0
-1,	-2,	-3

– exécuter le test.

b. Enregistrer, versionner.

⁷ JUnit : <https://junit.org/junit5/docs/current/user-guide/#overview-getting-started>

Utiliser <https://junit.org/junit5/docs/current/user-guide/#writing-tests-parameterized-tests-sources-CsvSource>

10.- Gestion d'ensembles

- a. Ajouter à la classe **Calculator** la méthode suivante :

```
public Set8<Integer> ensembleChiffres(int pNombre)
```

qui, étant donné l'entier passé en paramètre, retourne l'ensemble (non ordonné) des chiffres composant ce nombre.

Exemples : `ensembleChiffres(7679)` retourne l'ensemble {6, 7, 9} (listés dans un ordre quelconque)
`ensembleChiffres(-11)` retourne l'ensemble {1}

- b. Écrire et exécuter le/les tests associés
- c. Enregistrer, versionner.

⁸ Interface Set : <https://docs.oracle.com/javase/8/docs/api/java/util/Set.html>
et Classe HashSet : <https://docs.oracle.com/javase/8/docs/api/java/util/HashSet.html>

11.- Tester les méthodes contenant une levée d'exception

a. Pour chacune des méthodes dont les signatures suivent :

```
public int add(int opG, int opD) ;  
public int divide(int opG, int opD) ;
```

- identifier une situation d'erreur donnant lieu à une levée d'exception
- modifier sa définition en vue de prendre en charge cette erreur⁹
- écrire un test associé en utilisant la bibliothèque **jAssert**
- exécuter le test.

b. Enregistrer, versionner.

⁹ Java, hiérarchie des classes d'exceptions :

<https://docs.oracle.com/javase/8/docs/api/java/lang/Throwable.html>

<https://docs.oracle.com/javase/8/docs/api/java/lang/Exception.html>

<https://docs.oracle.com/javase/8/docs/api/java/lang/RuntimeException.html>

<https://docs.oracle.com/javase/8/docs/api/java/lang/ArithmeticException.html>

12.- Contrôler la couverture de code du projet

La **couverture de code** est une mesure utilisée pour décrire le taux de code source exécuté d'un programme quand une suite de test est lancée. Un programme avec une haute couverture de code, mesurée en pourcentage, a davantage de code exécuté durant les tests ce qui laisse à penser qu'il a moins de chance de contenir de bugs logiciels non détectés, comparativement à un programme avec une faible couverture de code¹⁰ :

a. Rapport de couverture de code généré par IntelliJ

- Exécuter les tests simultanément avec l'option 'with Coverage' (clic droit sur src/test/java puis Run 'Tests in Calculator with Coverage' - Figure 6)

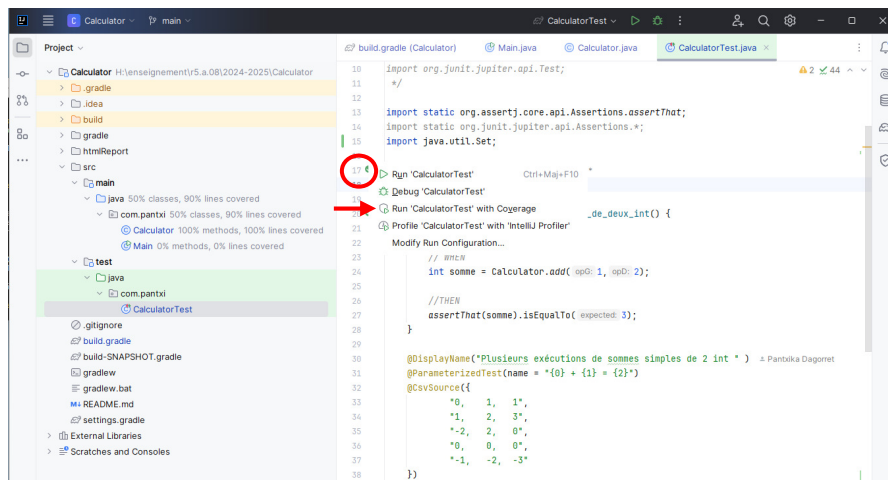


Figure 6 : Lancement d'un un test avec couverture de code

- Analyser les informations fournies par l'EDI : pourcentages de classes, de lignes, de méthodes, zones des fichiers sources couverts / non couverts par les tests (Figure 8)

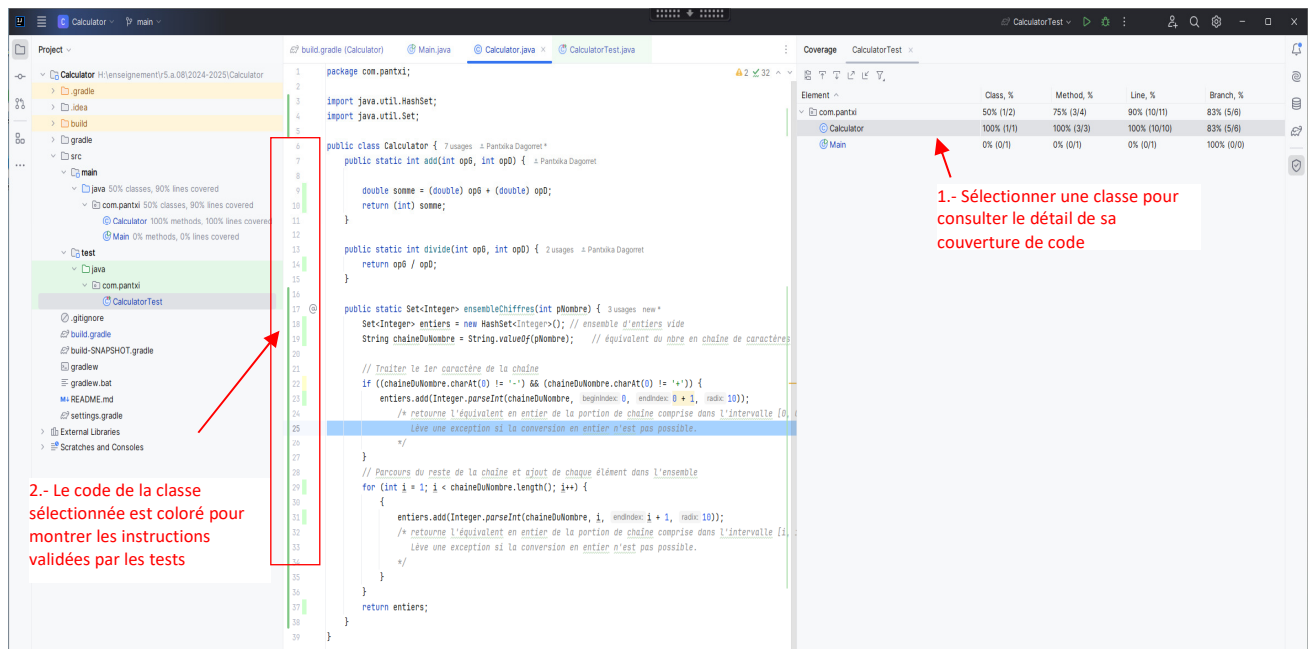


Figure 7 : Couverture de code illustrée directement sur l'EDI

¹⁰ https://fr.wikipedia.org/wiki/Couverture_de_code

- IntelliJ peut générer un rapport de test et couverture de code sous la forme de fichier html. Le nom et le dossier de rangement du rapport sont modifiables (Figure 8, Figure 9)

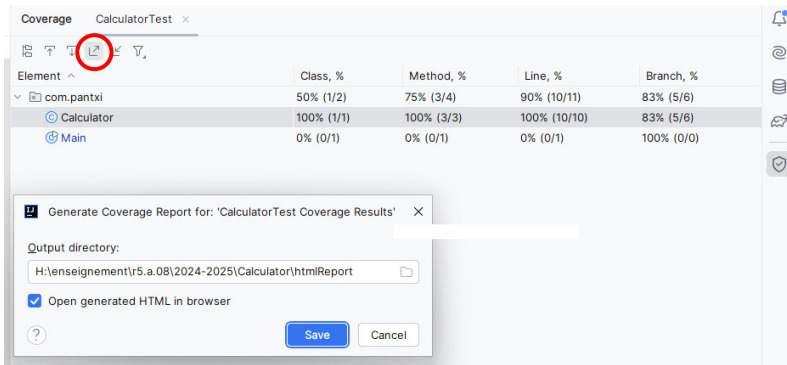


Figure 8 : Couverture de code générée par IntelliJ – reporting exporté au format HTML (1/2)

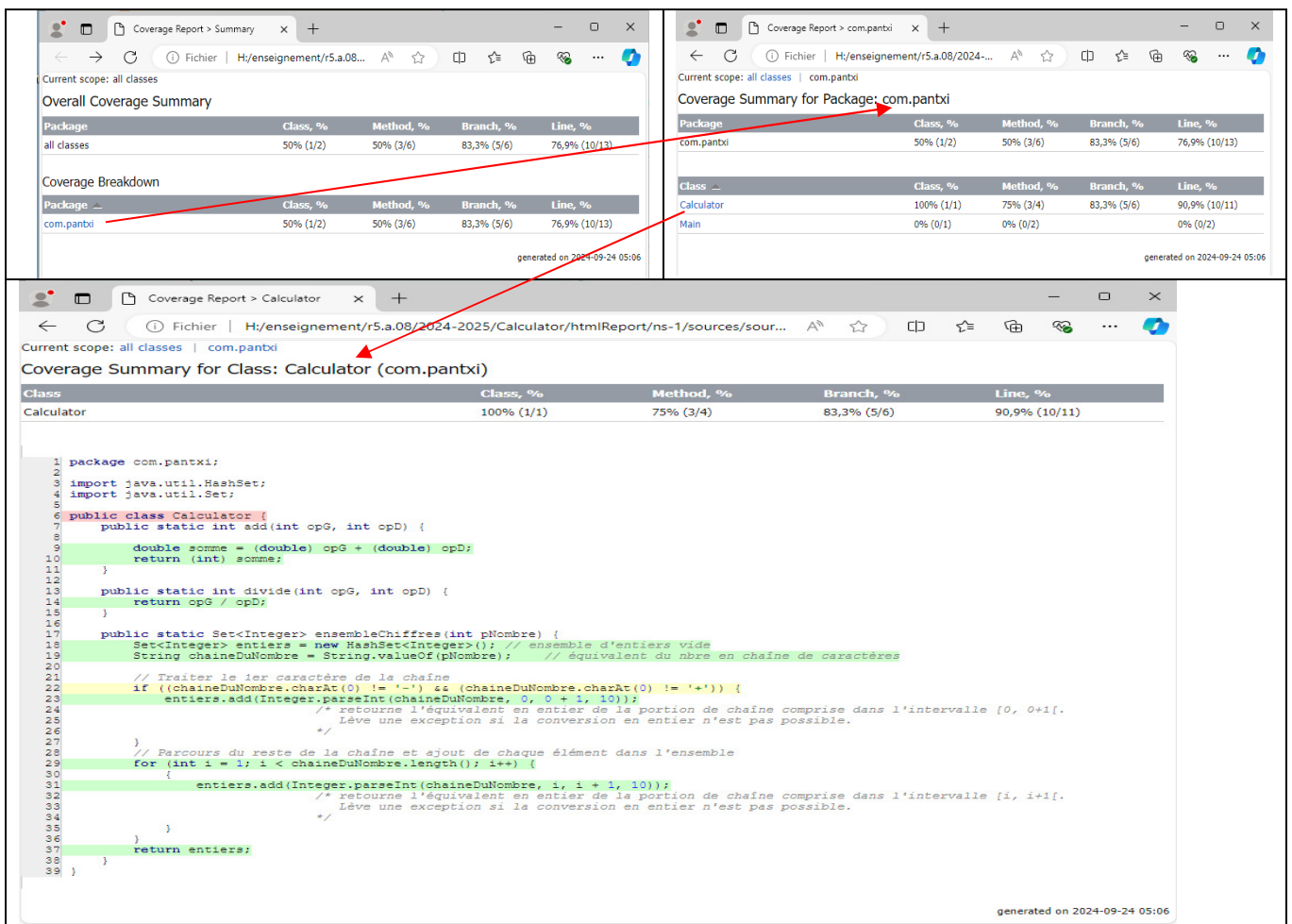


Figure 9 : Couverture de code générée par IntelliJ – reporting exporté au format HTML (1/2) (2/2)

b. Rapport de couverture de code .html généré par Jacoco ¹¹

Le rapport de couverture de code peut aussi être généré par JaCoCo (Java Code Coverage), un outil spécialisé. Le rapport généré est au format html. Le nom et le dossier de rangement du rapport sont modifiables

- Compléter le fichier **build.gradle** avec les directives concernant :
 - o Ajout de jacoco comme plugin de reporting (rubrique plugins : ajouter id 'jacoco')
 - o Finalisation du test par un rapport de test jacoco (rubrique test : ajouter finalizedBy jacocoTestReport)
 - o Préciser le dossier où sera généré le rapport :

```
jacocoTestReport {
    dependsOn test
    reports {
        xml.required = true
        html.outputLocation=layout.buildDirectory.dir('reports/jacoco/test')
        // dossier et fichier de rangement par défaut, mais modifiable, comme par exemple :
        // html.outputLocation=layout.projectDirectory.dir('reports/jacoco/test')
    }
}
```
- Pourquoi, d'après vous, le rapport de couverture de code est-il stocké par défaut dans le dossier **build** ?
- Depuis l'IDE, exécuter à nouveau les tests de la classe avec l'option 'with Coverage' (clic droit sur src/test/java puis Run 'Tests in Calculator with Covage' - Figure 6). L'EDI montre les tests générés par IntelliJ, mais un rapport .html est également généré par jacoco dans le dossier demandé.
- Enregistrer, versionner.

c. Créer un test (script) Jacoco générant un rapport de couverture de code au format .html.

- Sélectionner le menu Run / Edit Configuration(cf. Figure 10 à Figure 13).

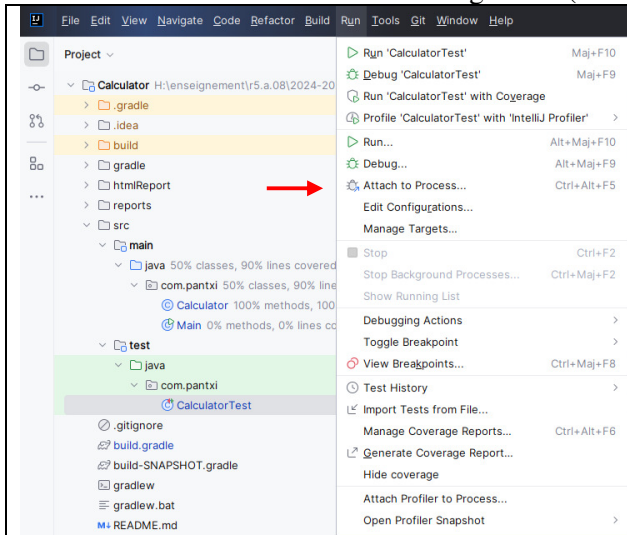


Figure 10 : Création d'un test avec rapport Jacoco (1/4)

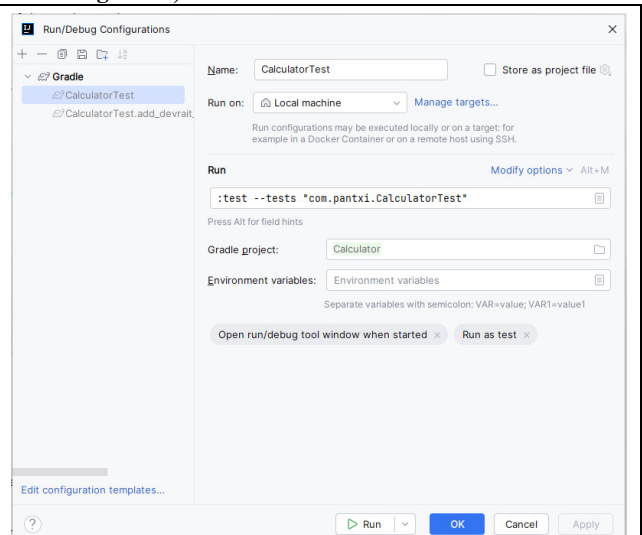


Figure 11 : Création d'un test avec rapport Jacoco (2/4)

¹¹ <https://www.jacoco.org/jacoco/trunk/doc/counters.html>

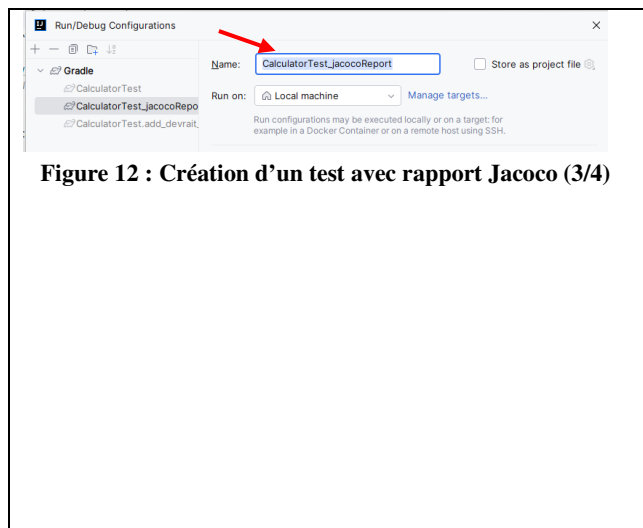


Figure 12 : Création d'un test avec rapport Jacoco (3/4)

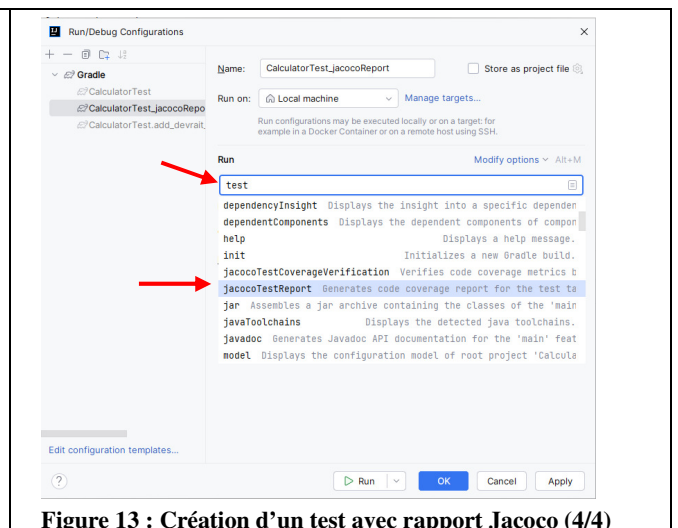


Figure 13 : Création d'un test avec rapport Jacoco (4/4)

- Exécuter le test avec rapport de couverture Jacoco créé (cf.)

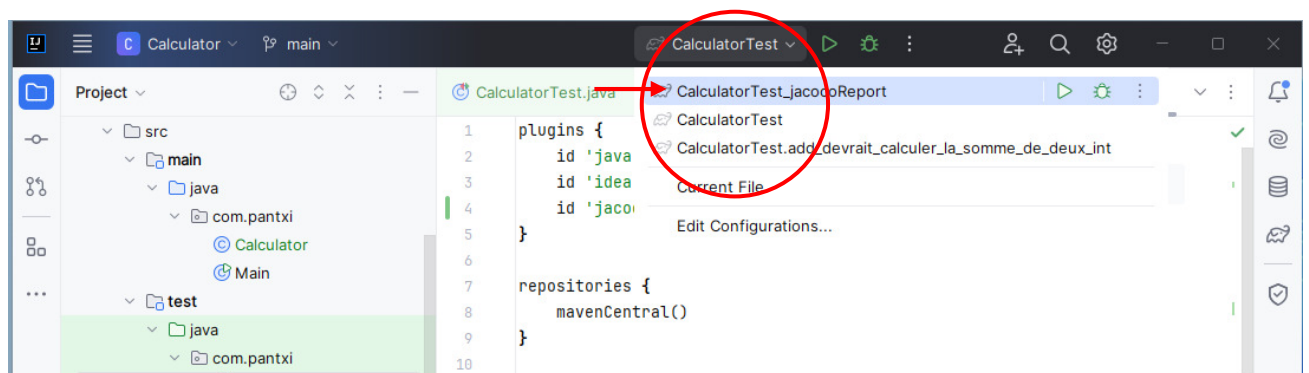


Figure 14 : Lancement du test créé

- Consulter les rapports générés par Jacoco dans le dossier nomDuProjet/reports (Figure 15)

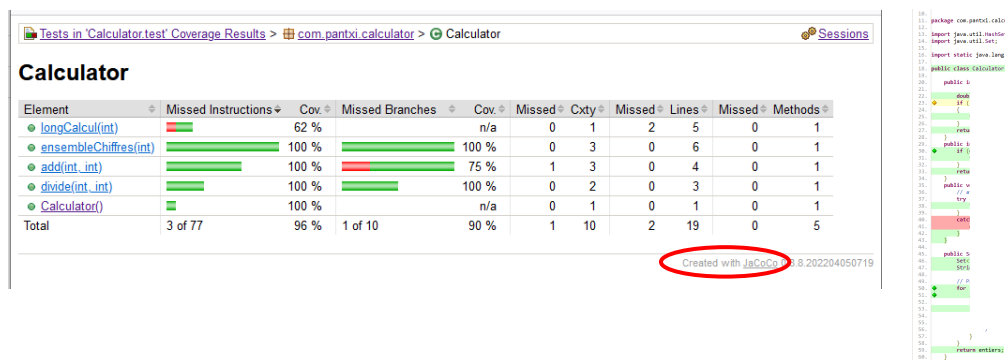


Figure 15 : Test avec Couverture de test – reporting fait par jacoco

- Enregistrer, versionner.

13.- Automatiser la construction du projet et l'exécution des tests lors du push : action Github

L'exécution fréquente des tests est nécessaire afin de s'assurer que tout nouvel ajout de fonctionnalité n'altère pas la validité du code déjà produit.

Afin de décharger le programmeur de cette tâche répétitive, il convient de l'automatiser. Cette pratique relève de ce que l'on appelle **Intégration Continue**¹².

Gradle est le **moteur de production**¹³ utilisé dans notre projet par IntelliJ. Il organise toutes les étapes de construction du projet, dont l'exécution des tests et les mesures de couverture de code.

- Utilisation de Gradle depuis l'IDE : onglet Gradle (Figure 16) :

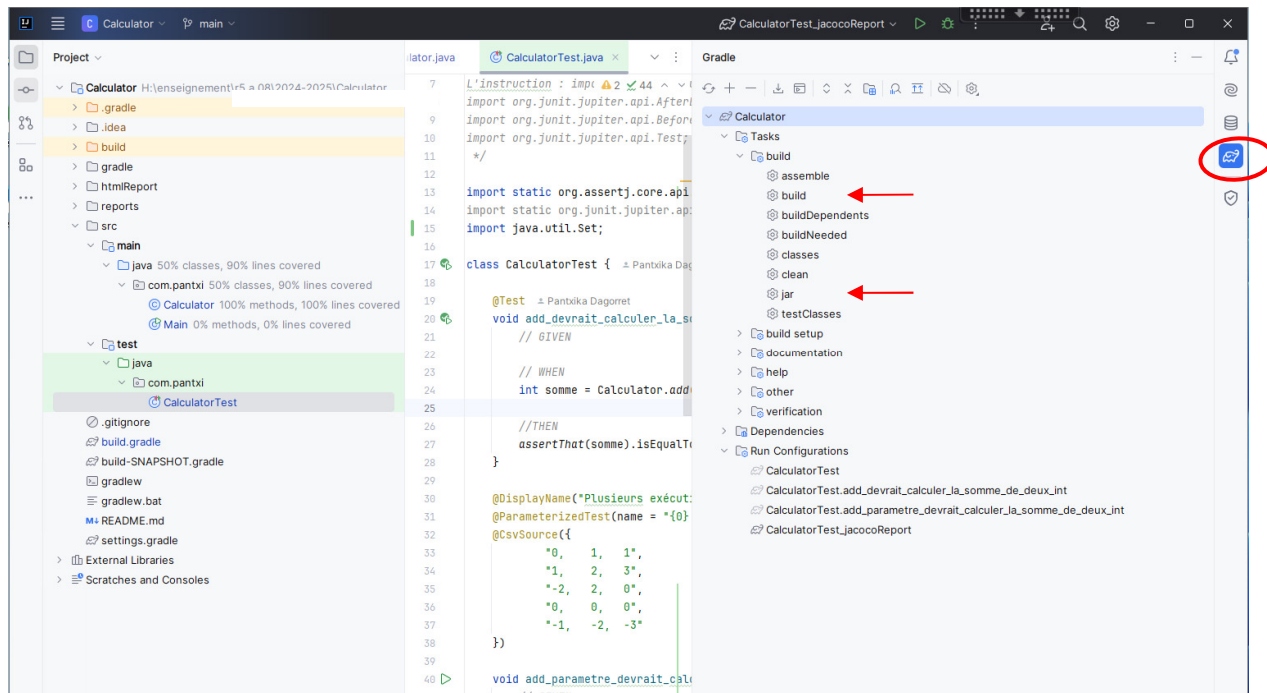


Figure 16 : Gradle sur l'IDE IntelliJ

- Utilisation en mode ligne de commande :

```
C:\Calculator> .\gradlew.bat test
C:\Calculator> .\gradlew.bat build
```

Il est possible de configurer Gradle afin que la construction du projet et exécution des tests soient exécutées **automatiquement lors de chaque push** vers le dépôt distant du projet (Github). Cela se fait grâce aux **actions Github**.

La procédure pour le faire est la suivante.

- Désactiver l'usage de Jacoco pour la génération du rapport de couverture → revenir au script de test nommé **calculatorTest** réalisant le test (et un éventuel reporting via IntelliJ) (menu Run/Edit Configurations : cf. 12.- Contrôler la couverture de code)
- S'assurer (en ligne de commande ou via l'interface Gradle de l'IDE) que l'exécution des tests est correcte (les tests s'exécutent) et que la construction du projet produit bien un fichier .jar (dans le dossier build/libs/).

Attention la variable d'environnement JAVA_HOME doit être configurée si vous utilisez un autre JDK que celui installé par défaut sur les machines virtuelles !

¹² https://fr.wikipedia.org/wiki/Int%C3%A9gration_continue

¹³ https://fr.wikipedia.org/wiki/Moteur_de_production

- c. Créer une branche sur le dépôt local (par exemple `github-action-build-tests-auto`)
- d. Dans cette branche, créer une **action Github** pour construire l'application et lancer les tests automatiquement à chaque dépôt. Pour ce faire :
 - À la racine du projet, créer un dossier `.github/workflows`
 - Dans ce dossier, copier le fichier `build.yml` fourni sur eLearn [qui vous convient](#) en fonction du JDK que vous avez utilisé, en prenant soin de le renommer en `build.yml`
 - Pousser sur Github, puis, sur GitHub, vérifier dans le volet Actions que l'action Github s'est bien terminée. Noter que le fichier `.jar` n'est pas présent sur le dépôt distant. Pourquoi ?
- e. Lorsque l'action Github s'est bien terminée, dans le dépôt local, fusionner (merge) la branche `github-action-build-tests-auto` avec la branche principale.
- f. Supprimer la branche `github-action-build-tests-auto` du dépôt local.

Enregistrer, versionner.

14.- Automatiser jusqu'à la production d'un rapport de tests consultable sur Github

- a. Créer une branche sur le dépôt local (par exemple github-action-rapport-test-auto)
- b. Ajouter l'étape suivante au fichier build.yml, utilisant l'action **test-reporter**.

```
- name: Rapport de tests
  uses: dorny/test-reporter@v1
  if: success() || failure()
  with:
    name: JUnit Tests
    path: build/test-results/test/TEST-*.xml14
    reporter: java-junit
```

- c. Sauvegarder et pousser sur GitHub. Vérifier que le rapport de test est bien généré.
- d. Fusionner (merge) la branche lorsque l'action github s'est bien terminée
- e. Supprimer la branche
- f. Enregistrer, versionner.

¹⁴ Le test-reporter se nourrit du fichier résultat de test. Justifier le nom et l'emplacement de ce fichier.

15.- Automatiser jusqu'à la production d'un rapport de couverture de code (Jacoco) consultable sur Github

La mise en place de cette action comporte deux volets :

- Utilisation du plugin jacoco par Gradle, qui génère des rapports de couverture de code dans divers formats : .html, .xml, ... pour être consultés soit par un humain, soit pour être traités automatiquement par un moteur de production. Cela supposera re-configurer le fichier build.gradle.
 - Création d'une action github pour récupérer le rapport de couverture (au format .xml) et générer son équivalent au format .html sur Github.
- a. Créer une branche sur le dépôt local (par exemple github-action-rapport-jacoco)
 - b. Modifier le fichier build.gradle¹⁵ :
 - compléter la rubrique **plugin** pour ajouter le plugin jacoco à la liste des plugins installés
 - compléter la rubrique **test** : le test doit se finir par la création d'un rapport de test jacocoTestReport
 - créer une rubrique **jacocoTestReport** décrivant le rapport de test avec les informations suivantes :
 - xml.required = true
 - les rapport de couverture de code doivent être généré dans le dossier nomDuProjet/build/reports/jacoco/test.
 - Mettre à jour le fichier gradle (cliquer sur l'icone 'éléphant' apparaissant dans la zone client de la fenêtre du fichier build.gradle)
 - c. Dans le fichier build.yml : enlever le commentaire de la ligne : `run: ./gradlew build #JacocoTestReport`
 - d. Exécuter les tests avec l'option 'with Coverage', consulter le rapport (.html) généré par Jacoco

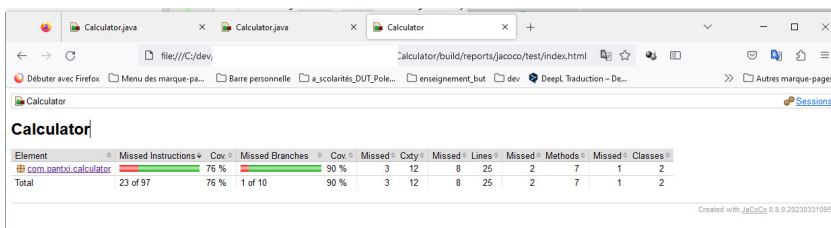


Figure 17 : Test avec Couverture de test – reporting fait par jacoco

```
10 package com.partei.calculator;
11 import java.util.HashMap;
12 import java.util.Map;
13 import static java.lang.Double.valueOf;
14
15 public class Calculator {
16
17     public static double add(int a, int b) {
18         return a + b;
19     }
20
21     public static double subtract(int a, int b) {
22         return a - b;
23     }
24
25     public static double multiply(int a, int b) {
26         return a * b;
27     }
28
29     public static double divide(int a, int b) {
30         return a / b;
31     }
32
33     public static double modulo(int a, int b) {
34         return a % b;
35     }
36
37     public static double power(int a, int b) {
38         return Math.pow(a, b);
39     }
40
41     public static double squareRoot(int a) {
42         return Math.sqrt(a);
43     }
44
45     public static double factorial(int a) {
46         return factorial(a);
47     }
48
49     public static double fibonacci(int a) {
50         return fibonacci(a);
51     }
52
53     public static double gcd(int a, int b) {
54         return gcd(a, b);
55     }
56
57     public static double lcm(int a, int b) {
58         return lcm(a, b);
59     }
60
61     public static double isPrime(int a) {
62         return isPrime(a);
63     }
64
65     public static double isPalindrome(int a) {
66         return isPalindrome(a);
67     }
68
69     public static double isArmstrong(int a) {
70         return isArmstrong(a);
71     }
72
73     public static double isHappy(int a) {
74         return isHappy(a);
75     }
76
77     public static double isStrong(int a) {
78         return isStrong(a);
79     }
80
81     public static double isCarmichael(int a) {
82         return isCarmichael(a);
83     }
84
85     public static double isSmarandache(int a) {
86         return isSmarandache(a);
87     }
88
89     public static double isPronic(int a) {
90         return isPronic(a);
91     }
92
93     public static double isTriangular(int a) {
94         return isTriangular(a);
95     }
96
97     public static double isPentagonal(int a) {
98         return isPentagonal(a);
99     }
100
101     public static double isHexagonal(int a) {
102         return isHexagonal(a);
103     }
104
105     public static double isOctagonal(int a) {
106         return isOctagonal(a);
107     }
108
109     public static double isNonagonal(int a) {
110         return isNonagonal(a);
111     }
112
113     public static double isDecagonal(int a) {
114         return isDecagonal(a);
115     }
116
117     public static double isDodecagonal(int a) {
118         return isDodecagonal(a);
119     }
120
121     public static double isTetrahedral(int a) {
122         return isTetrahedral(a);
123     }
124
125     public static double isSquare(int a) {
126         return isSquare(a);
127     }
128
129     public static double isCube(int a) {
130         return isCube(a);
131     }
132
133     public static double isFibonacci(int a) {
134         return isFibonacci(a);
135     }
136
137     public static double isLucas(int a) {
138         return isLucas(a);
139     }
140
141     public static double isFibonacciPrime(int a) {
142         return isFibonacciPrime(a);
143     }
144
145     public static double isLucasPrime(int a) {
146         return isLucasPrime(a);
147     }
148
149     public static double isFibonacciSquare(int a) {
150         return isFibonacciSquare(a);
151     }
152
153     public static double isLucasSquare(int a) {
154         return isLucasSquare(a);
155     }
156
157     public static double isFibonacciCube(int a) {
158         return isFibonacciCube(a);
159     }
160
161     public static double isLucasCube(int a) {
162         return isLucasCube(a);
163     }
164
165     public static double isFibonacciPentagonal(int a) {
166         return isFibonacciPentagonal(a);
167     }
168
169     public static double isLucasPentagonal(int a) {
170         return isLucasPentagonal(a);
171     }
172
173     public static double isFibonacciHexagonal(int a) {
174         return isFibonacciHexagonal(a);
175     }
176
177     public static double isLucasHexagonal(int a) {
178         return isLucasHexagonal(a);
179     }
180
181     public static double isFibonacciOctagonal(int a) {
182         return isFibonacciOctagonal(a);
183     }
184
185     public static double isLucasOctagonal(int a) {
186         return isLucasOctagonal(a);
187     }
188
189     public static double isFibonacciNonagonal(int a) {
190         return isFibonacciNonagonal(a);
191     }
192
193     public static double isLucasNonagonal(int a) {
194         return isLucasNonagonal(a);
195     }
196
197     public static double isFibonacciDecagonal(int a) {
198         return isFibonacciDecagonal(a);
199     }
200
201     public static double isLucasDecagonal(int a) {
202         return isLucasDecagonal(a);
203     }
204
205     public static double isFibonacciDodecagonal(int a) {
206         return isFibonacciDodecagonal(a);
207     }
208
209     public static double isLucasDodecagonal(int a) {
210         return isLucasDodecagonal(a);
211     }
212
213     public static double isFibonacciTetrahedral(int a) {
214         return isFibonacciTetrahedral(a);
215     }
216
217     public static double isLucasTetrahedral(int a) {
218         return isLucasTetrahedral(a);
219     }
220
221     public static double isFibonacciSquare(int a) {
222         return isFibonacciSquare(a);
223     }
224
225     public static double isLucasSquare(int a) {
226         return isLucasSquare(a);
227     }
228
229     public static double isFibonacciCube(int a) {
230         return isFibonacciCube(a);
231     }
232
233     public static double isLucasCube(int a) {
234         return isLucasCube(a);
235     }
236
237     public static double isFibonacciPentagonal(int a) {
238         return isFibonacciPentagonal(a);
239     }
240
241     public static double isLucasPentagonal(int a) {
242         return isLucasPentagonal(a);
243     }
244
245     public static double isFibonacciHexagonal(int a) {
246         return isFibonacciHexagonal(a);
247     }
248
249     public static double isLucasHexagonal(int a) {
250         return isLucasHexagonal(a);
251     }
252
253     public static double isFibonacciOctagonal(int a) {
254         return isFibonacciOctagonal(a);
255     }
256
257     public static double isLucasOctagonal(int a) {
258         return isLucasOctagonal(a);
259     }
260
261     public static double isFibonacciNonagonal(int a) {
262         return isFibonacciNonagonal(a);
263     }
264
265     public static double isLucasNonagonal(int a) {
266         return isLucasNonagonal(a);
267     }
268
269     public static double isFibonacciDecagonal(int a) {
270         return isFibonacciDecagonal(a);
271     }
272
273     public static double isLucasDecagonal(int a) {
274         return isLucasDecagonal(a);
275     }
276
277     public static double isFibonacciDodecagonal(int a) {
278         return isFibonacciDodecagonal(a);
279     }
280
281     public static double isLucasDodecagonal(int a) {
282         return isLucasDodecagonal(a);
283     }
284
285     public static double isFibonacciTetrahedral(int a) {
286         return isFibonacciTetrahedral(a);
287     }
288
289     public static double isLucasTetrahedral(int a) {
290         return isLucasTetrahedral(a);
291     }
292
293     public static double isFibonacciSquare(int a) {
294         return isFibonacciSquare(a);
295     }
296
297     public static double isLucasSquare(int a) {
298         return isLucasSquare(a);
299     }
300
301     public static double isFibonacciCube(int a) {
302         return isFibonacciCube(a);
303     }
304
305     public static double isLucasCube(int a) {
306         return isLucasCube(a);
307     }
308
309     public static double isFibonacciPentagonal(int a) {
310         return isFibonacciPentagonal(a);
311     }
312
313     public static double isLucasPentagonal(int a) {
314         return isLucasPentagonal(a);
315     }
316
317     public static double isFibonacciHexagonal(int a) {
318         return isFibonacciHexagonal(a);
319     }
320
321     public static double isLucasHexagonal(int a) {
322         return isLucasHexagonal(a);
323     }
324
325     public static double isFibonacciOctagonal(int a) {
326         return isFibonacciOctagonal(a);
327     }
328
329     public static double isLucasOctagonal(int a) {
330         return isLucasOctagonal(a);
331     }
332
333     public static double isFibonacciNonagonal(int a) {
334         return isFibonacciNonagonal(a);
335     }
336
337     public static double isLucasNonagonal(int a) {
338         return isLucasNonagonal(a);
339     }
340
341     public static double isFibonacciDecagonal(int a) {
342         return isFibonacciDecagonal(a);
343     }
344
345     public static double isLucasDecagonal(int a) {
346         return isLucasDecagonal(a);
347     }
348
349     public static double isFibonacciDodecagonal(int a) {
350         return isFibonacciDodecagonal(a);
351     }
352
353     public static double isLucasDodecagonal(int a) {
354         return isLucasDodecagonal(a);
355     }
356
357     public static double isFibonacciTetrahedral(int a) {
358         return isFibonacciTetrahedral(a);
359     }
360
361     public static double isLucasTetrahedral(int a) {
362         return isLucasTetrahedral(a);
363     }
364
365     public static double isFibonacciSquare(int a) {
366         return isFibonacciSquare(a);
367     }
368
369     public static double isLucasSquare(int a) {
370         return isLucasSquare(a);
371     }
372
373     public static double isFibonacciCube(int a) {
374         return isFibonacciCube(a);
375     }
376
377     public static double isLucasCube(int a) {
378         return isLucasCube(a);
379     }
380
381     public static double isFibonacciPentagonal(int a) {
382         return isFibonacciPentagonal(a);
383     }
384
385     public static double isLucasPentagonal(int a) {
386         return isLucasPentagonal(a);
387     }
388
389     public static double isFibonacciHexagonal(int a) {
390         return isFibonacciHexagonal(a);
391     }
392
393     public static double isLucasHexagonal(int a) {
394         return isLucasHexagonal(a);
395     }
396
397     public static double isFibonacciOctagonal(int a) {
398         return isFibonacciOctagonal(a);
399     }
400
401     public static double isLucasOctagonal(int a) {
402         return isLucasOctagonal(a);
403     }
404
405     public static double isFibonacciNonagonal(int a) {
406         return isFibonacciNonagonal(a);
407     }
408
409     public static double isLucasNonagonal(int a) {
410         return isLucasNonagonal(a);
411     }
412
413     public static double isFibonacciDecagonal(int a) {
414         return isFibonacciDecagonal(a);
415     }
416
417     public static double isLucasDecagonal(int a) {
418         return isLucasDecagonal(a);
419     }
420
421     public static double isFibonacciDodecagonal(int a) {
422         return isFibonacciDodecagonal(a);
423     }
424
425     public static double isLucasDodecagonal(int a) {
426         return isLucasDodecagonal(a);
427     }
428
429     public static double isFibonacciTetrahedral(int a) {
430         return isFibonacciTetrahedral(a);
431     }
432
433     public static double isLucasTetrahedral(int a) {
434         return isLucasTetrahedral(a);
435     }
436
437     public static double isFibonacciSquare(int a) {
438         return isFibonacciSquare(a);
439     }
440
441     public static double isLucasSquare(int a) {
442         return isLucasSquare(a);
443     }
444
445     public static double isFibonacciCube(int a) {
446         return isFibonacciCube(a);
447     }
448
449     public static double isLucasCube(int a) {
450         return isLucasCube(a);
451     }
452
453     public static double isFibonacciPentagonal(int a) {
454         return isFibonacciPentagonal(a);
455     }
456
457     public static double isLucasPentagonal(int a) {
458         return isLucasPentagonal(a);
459     }
460
461     public static double isFibonacciHexagonal(int a) {
462         return isFibonacciHexagonal(a);
463     }
464
465     public static double isLucasHexagonal(int a) {
466         return isLucasHexagonal(a);
467     }
468
469     public static double isFibonacciOctagonal(int a) {
470         return isFibonacciOctagonal(a);
471     }
472
473     public static double isLucasOctagonal(int a) {
474         return isLucasOctagonal(a);
475     }
476
477     public static double isFibonacciNonagonal(int a) {
478         return isFibonacciNonagonal(a);
479     }
480
481     public static double isLucasNonagonal(int a) {
482         return isLucasNonagonal(a);
483     }
484
485     public static double isFibonacciDecagonal(int a) {
486         return isFibonacciDecagonal(a);
487     }
488
489     public static double isLucasDecagonal(int a) {
490         return isLucasDecagonal(a);
491     }
492
493     public static double isFibonacciDodecagonal(int a) {
494         return isFibonacciDodecagonal(a);
495     }
496
497     public static double isLucasDodecagonal(int a) {
498         return isLucasDodecagonal(a);
499     }
500
501     public static double isFibonacciTetrahedral(int a) {
502         return isFibonacciTetrahedral(a);
503     }
504
505     public static double isLucasTetrahedral(int a) {
506         return isLucasTetrahedral(a);
507     }
508
509     public static double isFibonacciSquare(int a) {
510         return isFibonacciSquare(a);
511     }
512
513     public static double isLucasSquare(int a) {
514         return isLucasSquare(a);
515     }
516
517     public static double isFibonacciCube(int a) {
518         return isFibonacciCube(a);
519     }
520
521     public static double isLucasCube(int a) {
522         return isLucasCube(a);
523     }
524
525     public static double isFibonacciPentagonal(int a) {
526         return isFibonacciPentagonal(a);
527     }
528
529     public static double isLucasPentagonal(int a) {
530         return isLucasPentagonal(a);
531     }
532
533     public static double isFibonacciHexagonal(int a) {
534         return isFibonacciHexagonal(a);
535     }
536
537     public static double isLucasHexagonal(int a) {
538         return isLucasHexagonal(a);
539     }
540
541     public static double isFibonacciOctagonal(int a) {
542         return isFibonacciOctagonal(a);
543     }
544
545     public static double isLucasOctagonal(int a) {
546         return isLucasOctagonal(a);
547     }
548
549     public static double isFibonacciNonagonal(int a) {
550         return isFibonacciNonagonal(a);
551     }
552
553     public static double isLucasNonagonal(int a) {
554         return isLucasNonagonal(a);
555     }
556
557     public static double isFibonacciDecagonal(int a) {
558         return isFibonacciDecagonal(a);
559     }
560
561     public static double isLucasDecagonal(int a) {
562         return isLucasDecagonal(a);
563     }
564
565     public static double isFibonacciDodecagonal(int a) {
566         return isFibonacciDodecagonal(a);
567     }
568
569     public static double isLucasDodecagonal(int a) {
570         return isLucasDodecagonal(a);
571     }
572
573     public static double isFibonacciTetrahedral(int a) {
574         return isFibonacciTetrahedral(a);
575     }
576
577     public static double isLucasTetrahedral(int a) {
578         return isLucasTetrahedral(a);
579     }
580
581     public static double isFibonacciSquare(int a) {
582         return isFibonacciSquare(a);
583     }
584
585     public static double isLucasSquare(int a) {
586         return isLucasSquare(a);
587     }
588
589     public static double isFibonacciCube(int a) {
590         return isFibonacciCube(a);
591     }
592
593     public static double isLucasCube(int a) {
594         return isLucasCube(a);
595     }
596
597     public static double isFibonacciPentagonal(int a) {
598         return isFibonacciPentagonal(a);
599     }
600
601     public static double isLucasPentagonal(int a) {
602         return isLucasPentagonal(a);
603     }
604
605     public static double isFibonacciHexagonal(int a) {
606         return isFibonacciHexagonal(a);
607     }
608
609     public static double isLucasHexagonal(int a) {
610         return isLucasHexagonal(a);
611     }
612
613     public static double isFibonacciOctagonal(int a) {
614         return isFibonacciOctagonal(a);
615     }
616
617     public static double isLucasOctagonal(int a) {
618         return isLucasOctagonal(a);
619     }
620
621     public static double isFibonacciNonagonal(int a) {
622         return isFibonacciNonagonal(a);
623     }
624
625     public static double isLucasNonagonal(int a) {
626         return isLucasNonagonal(a);
627     }
628
629     public static double isFibonacciDecagonal(int a) {
630         return isFibonacciDecagonal(a);
631     }
632
633     public static double isLucasDecagonal(int a) {
634         return isLucasDecagonal(a);
635     }
636
637     public static double isFibonacciDodecagonal(int a) {
638         return isFibonacciDodecagonal(a);
639     }
640
641     public static double isLucasDodecagonal(int a) {
642         return isLucasDodecagonal(a);
643     }
644
645     public static double isFibonacciTetrahedral(int a) {
646         return isFibonacciTetrahedral(a);
647     }
648
649     public static double isLucasTetrahedral(int a) {
650         return isLucasTetrahedral(a);
651     }
652
653     public static double isFibonacciSquare(int a) {
654         return isFibonacciSquare(a);
655     }
656
657     public static double isLucasSquare(int a) {
658         return isLucasSquare(a);
659     }
660
661     public static double isFibonacciCube(int a) {
662         return isFibonacciCube(a);
663     }
664
665     public static double isLucasCube(int a) {
666         return isLucasCube(a);
667     }
668
669     public static double isFibonacciPentagonal(int a) {
670         return isFibonacciPentagonal(a);
671     }
672
673     public static double isLucasPentagonal(int a) {
674         return isLucasPentagonal(a);
675     }
676
677     public static double isFibonacciHexagonal(int a) {
678         return isFibonacciHexagonal(a);
679     }
680
681     public static double isLucasHexagonal(int a) {
682         return isLucasHexagonal(a);
683     }
684
685     public static double isFibonacciOctagonal(int a) {
686         return isFibonacciOctagonal(a);
687     }
688
689     public static double isLucasOctagonal(int a) {
690         return isLucasOctagonal(a);
691     }
692
693     public static double isFibonacciNonagonal(int a) {
694         return isFibonacciNonagonal(a);
695     }
696
697     public static double isLucasNonagonal(int a) {
698         return isLucasNonagonal(a);
699     }
700
701     public static double isFibonacciDecagonal(int a) {
702         return isFibonacciDecagonal(a);
703     }
704
705     public static double isLucasDecagonal(int a) {
706         return isLucasDecagonal(a);
707     }
708
709     public static double isFibonacciDodecagonal(int a) {
710         return isFibonacciDodecagonal(a);
711     }
712
713     public static double isLucasDodecagonal(int a) {
714         return isLucasDodecagonal(a);
715     }
716
717     public static double isFibonacciTetrahedral(int a) {
718         return isFibonacciTetrahedral(a);
719     }
720
721     public static double isLucasTetrahedral(int a) {
722         return isLucasTetrahedral(a);
723     }
724
725     public static double isFibonacciSquare(int a) {
726         return isFibonacciSquare(a);
727     }
728
729     public static double isLucasSquare(int a) {
730         return isLucasSquare(a);
731     }
732
733     public static double isFibonacciCube(int a) {
734         return isFibonacciCube(a);
735     }
736
737     public static double isLucasCube(int a) {
738         return isLucasCube(a);
739     }
740
741     public static double isFibonacciPentagonal(int a) {
742         return isFibonacciPentagonal(a);
743     }
744
745     public static double isLucasPentagonal(int a) {
746         return isLucasPentagonal(a);
747     }
748
749     public static double isFibonacciHexagonal(int a) {
750         return isFibonacciHexagonal(a);
751     }
752
753     public static double isLucasHexagonal(int a) {
754         return isLucasHexagonal(a);
755     }
756
757     public static double isFibonacciOctagonal(int a) {
758         return isFibonacciOctagonal(a);
759     }
760
761     public static double isLucasOctagonal(int a) {
762         return isLucasOctagonal(a);
763     }
764
765     public static double isFibonacciNonagonal(int a) {
766         return isFibonacciNonagonal(a);
767     }
768
769     public static double isLucasNonagonal(int a) {
770         return isLucasNonagonal(a);
771     }
772
773     public static double isFibonacciDecagonal(int a) {
774         return isFibonacciDecagonal(a);
775     }
776
777     public static double isLucasDecagonal(int a) {
778         return isLucasDecagonal(a);
779     }
780
781     public static double isFibonacciDodecagonal(int a) {
782         return isFibonacciDodecagonal(a);
783     }
784
785     public static double isLucasDodecagonal(int a) {
786         return isLucasDodecagonal(a);
787     }
788
789     public static double isFibonacciTetrahedral(int a) {
790         return isFibonacciTetrahedral(a);
791     }
792
793     public static double isLucasTetrahedral(int a) {
794         return isLucasTetrahedral(a);
795     }
796
797     public static double isFibonacciSquare(int a) {
798         return isFibonacciSquare(a);
799     }
800
801     public static double isLucasSquare(int a) {
802         return isLucasSquare(a);
803     }
804
805     public static double isFibonacciCube(int a) {
806         return isFibonacciCube(a);
807     }
808
809     public static double isLucasCube(int a) {
810         return isLucasCube(a);
811     }
812
813     public static double isFibonacciPentagonal(int a) {
814         return isFibonacciPentagonal(a);
815     }
816
817     public static double isLucasPentagonal(int a) {
818         return isLucasPentagonal(a);
819     }
820
821     public static double isFibonacciHexagonal(int a) {
822         return isFibonacciHexagonal(a);
823     }
824
825     public static double isLucasHexagonal(int a) {
826         return isLucasHexagonal(a);
827     }
828
829     public static double isFibonacciOctagonal(int a) {
830         return isFibonacciOctagonal(a);
831     }
832
833     public static double isLucasOctagonal(int a) {
834         return isLucasOctagonal(a);
835     }
836
837     public static double isFibonacciNonagonal(int a) {
838         return isFibonacciNonagonal(a);
839     }
840
841     public static double isLucasNonagonal(int a) {
842         return isLucasNonagonal(a);
843     }
844
845     public static double isFibonacciDecagonal(int a) {
846         return isFibonacciDecagonal(a);
847     }
848
849     public static double isLucasDecagonal(int a) {
850         return isLucasDecagonal(a);
851     }
852
853     public static double isFibonacciDodecagonal(int a) {
854         return isFibonacciDodecagonal(a);
855     }
856
857     public static double isLucasDodecagonal(int a) {
858         return isLucasDodecagonal(a);
859     }
860
861     public static double isFibonacciTetrahedral(int a) {
862         return isFibonacciTetrahedral(a);
863     }
864
865     public static double isLucasTetrahedral(int a) {
866         return isLucasTetrahedral(a);
867     }
868
869     public static double isFibonacciSquare(int a) {
870         return isFibonacciSquare(a);
871     }
872
873     public static double isLucasSquare(int a) {
874         return isLucasSquare(a);
875     }
876
877     public static double isFibonacciCube(int a) {
878         return isFibonacciCube(a);
879     }
880
881     public static double isLucasCube(int a) {
882         return isLucasCube(a);
883     }
884
885     public static double isFibonacciPentagonal(int a) {
886         return isFibonacciPentagonal(a);
887     }
888
889     public static double isLucasPentagonal(int a) {
890         return isLucasPentagonal(a);
891     }
892
893     public static double isFibonacciHexagonal(int a) {
894         return isFibonacciHexagonal(a);
895     }
896
897     public static double isLucasHexagonal(int a) {
898         return isLucasHexagonal(a);
899     }
900
901     public static double isFibonacciOctagonal(int a) {
902         return isFibonacciOctagonal(a);
903     }
904
905     public static double isLucasOctagonal(int a) {
906         return isLucasOctagonal(a);
907     }
908
909     public static double isFibonacciNonagonal(int a) {
910         return isFibonacciNonagonal(a);
911     }
912
913     public static double isLucasNonagonal(int a) {
914         return isLucasNonagonal(a);
915     }
916
917     public static double isFibonacciDecagonal(int a) {
918         return isFibonacciDecagonal(a);
919     }
920
921     public static double isLucasDecagonal(int a) {
922         return isLucasDecagonal(a);
923     }
924
925     public static double isFibonacciDodecagonal(int a) {
926         return isFibonacciDodecagonal(a);
927     }
928
929     public static double isLucasDodecagonal(int a) {
930         return isLucasDodecagonal(a);
931     }
932
933     public static double isFibonacciTetrahedral(int a) {
934         return isFibonacciTetrahedral(a);
935     }
936
937     public static double isLucasTetrahedral(int a) {
938         return isLucasTetrahedral(a);
939     }
940
941     public static double isFibonacciSquare(int a) {
942         return isFibonacciSquare(a);
943     }
944
945     public static double isLucasSquare(int a) {
946         return isLucasSquare(a);
947     }
948
949     public static double isFibonacciCube(int a) {
950         return isFibonacciCube(a);
951     }
952
953     public static double isLucasCube(int a) {
954         return isLucasCube(a);
955     }
956
957     public static double isFibonacciPentagonal(int a) {
958         return isFibonacciPentagonal(a);
959     }
960
961     public static double isLucasPentagonal(int a) {
962         return isLucasPentagonal(a);
963     }
964
965     public static double isFibonacciHexagonal(int a) {
966         return isFibonacciHexagonal(a);
967     }
968
969     public static double isLucasHexagonal(int a) {
970         return isLucasHexagonal(a);
971     }
972
973     public static double isFibonacciOctagonal(int a) {
974         return isFibonacciOctagonal(a);
975     }
976
977     public static double isLucasOctagonal(int a) {
978         return isLucasOctagonal(a);
979     }
980
981     public static double isFibonacciNonagonal(int a) {
982         return isFibonacciNonagonal(a);
983     }
984
985     public static double isLucasNonagonal(int a) {
986         return isLucasNonagonal(a);
987     }
988
989     public static double isFibonacciDecagonal(int a) {
990         return isFibonacciDecagonal(a);
991     }
992
993     public static double isLucasDecagonal(int a) {
994         return isLucasDecagonal(a);
995     }
996
997     public static double isFibonacciDodecagonal(int a) {
998         return isFibonacciDodecagonal(a);
999     }
1000    public static double isLucasDodecagonal(int a) {
1001        return isLucasDodecagonal(a);
1002    }
1003
1004    public static double isFibonacciTetrahedral(int a) {
1005        return isFibonacciTetrahedral(a);
1006    }
1007
1008    public static double isLucasTetrahedral(int a) {
1009        return isLucasTetrahedral(a);
1010    }
1011
1012    public static double isFibonacciSquare(int a) {
1013        return isFibonacciSquare(a);
1014    }
1015
1016    public static double isLucasSquare(int a) {
1017        return isLucasSquare(a);
1018    }
1019
1020    public static double isFibonacciCube(int a) {
1021        return isFibonacciCube(a);
1022    }
1023
1024    public static double isLucasCube(int a) {
1025        return isLucasCube(a);
1026    }
1027
1028    public static double isFibonacciPentagonal(int a) {
1029        return isFibonacciPentagonal(a);
1030    }
1031
1032    public static double isLucasPentagonal(int a) {
1033        return isLucasPentagonal(a);
1034    }
1035
1036    public static double isFibonacciHexagonal(int a) {
1037        return isFibonacciHexagonal(a);
1038    }
1039
1040    public static double isLucasHexagonal(int a) {
1041        return isLucasHexagonal(a);
1042    }
1043
1044    public static double isFibonacciOctagonal(int a) {
1045        return isFibonacciOctagonal(a);
1046    }
1047
1048    public static double isLucasOctagonal(int a) {
1049        return isLucasOctagonal(a);
1050    }
1051
1052    public static double isFibonacciNonagonal(int a) {
1053        return isFibonacciNonagonal(a);
1054    }
1055
1056    public static double isLucasNonagonal(int a) {
1057        return isLucasNonagonal(a);
1058    }
1059
1060    public static double isFibonacciDecagonal(int a) {
1061        return isFibonacciDecagonal(a);
1062    }
1063
1064    public static double isLucasDecagonal(int a) {
1065        return isLucasDecagonal(a);
1066    }
1067
1068    public static double isFibonacciDodecagonal(int a) {
1069        return isFibonacciDodecagonal(a);
1070    }
1071
1072    public static double isLucasDodecagonal(int a) {
1073        return isLucasDodecagonal(a);
1074    }
1075
1076    public static double isFibonacciTetrahedral(int a) {
1077        return isFibonacciTetrahedral(a);
1078    }
1079
1080    public static double isLucasTetrahedral(int a) {
1081        return isLucasTetrahedral(a);
1082    }
1083
1084    public static double isFibonacciSquare(int a) {
1085        return isFibonacciSquare(a);
1086    }
1087
1088    public static double isLucasSquare(int a) {
1089        return isLucasSquare(a);
1090    }
1091
1092    public static double isFibonacciCube(int a) {
1093        return isFibonacciCube(a);
1094    }
1095
1096    public static double isLucasCube(int a) {
1097        return isLucasCube(a);
1098    }
1099
1100    public static double isFibonacciPentagonal(int a) {
1101        return isFibonacciPentagonal(a);
1102    }
1103
1104    public static double isLucasPentagonal(int a) {
1105        return isLucasPentagonal(a);
1106    }
1107
1108    public static double isFibonacciHexagonal(int a) {
1109        return isFibonacciHexagonal(a);
1110    }
1111
1112    public static double isLucasHexagonal(int a) {
1113        return isLucasHexagonal(a);
1114    }
1115
1116    public static double isFibonacciOctagonal(int a) {
1117        return isFibonacciOctagonal(a);
1118    }
1119
1120    public static double isLucasOctagonal(int a) {
1121        return isLucasOctagonal(a);
1122    }
1123
1124    public static double isFibonacciNonagonal(int a) {
1125        return isFibonacciNonagonal(a);
1126    }
1127
1128    public static double isLucasNonagonal(int a) {
1129        return isLucasNonagonal(a);
1130    }
1131
1132    public static double isFibonacciDecagonal(int a) {
1133        return isFibonacciDecagonal(a);
1134    }
1135
1136    public static double isLucasDecagonal(int a)
```


16.- Conditionner la construction du projet à la qualité de la couverture de code

La construction du projet est gérée par Gradle. Il faut donc modifier le fichier build.gradle et y ajouter une condition de fin de construction, de sorte à faire échouer la construction du projet si un certain niveau de couverture n'est pas atteint.

Vous pouvez vous inspirer des résultats obtenus au point 12.- précédent, ou bien fixer un seuil arbitraire pour vérifier que cette nouvelle fonctionnalité est opérationnelle. Dans l'illustration ci-dessous, le seuil a été fixé à 0,9.

```
C:\dev\xxx\Calculator>.\gradlew.bat build

> Task :test
CalculatorTest > longCalcul_devrait_durer_moins_d_1_seconde() PASSED
CalculatorTest > divide_devrait_lever_une_exception_quand_diviseur_est_0() PASSED
CalculatorTest > add_devrait_lever_une_exception_si_somme_hors_intervalle_des_int() PASSED
CalculatorTest > add_devrait_calculer_la_somme_de_deux_int() PASSED
CalculatorTest > Plusieurs tests de sommes simples de 2 int > 0 + 1 = 1 PASSED
CalculatorTest > Plusieurs tests de sommes simples de 2 int > 1 + 2 = 3 PASSED
CalculatorTest > Plusieurs tests de sommes simples de 2 int > -2 + 2 = 0 PASSED
CalculatorTest > Plusieurs tests de sommes simples de 2 int > 0 + 0 = 0 PASSED
CalculatorTest > Plusieurs tests de sommes simples de 2 int > -1 + -2 = -3 PASSED

> Task :jacocoTestCoverageVerification FAILED
[ant:jacocoReport] Rule violated for bundle Calculator: instructions covered ratio is 0.4, but expected
minimum is 0.9

FAILURE: Build failed with an exception.

* What went wrong:
Execution failed for task ':jacocoTestCoverageVerification'.
> Rule violated for bundle Calculator: instructions covered ratio is 0.4, but expected minimum is 0.9

BUILD FAILED in 19s
6 actionable tasks: 1 executed, 5 up-to-date

C:\dev\xxx\Calculator>
```

- Créer une branche sur le dépôt local (par exemple github-action-seuil-couverture)
- Compléter le fichier build.gradle¹⁷ avec une rubrique **jacocoTestCoverageVerification** + le seuil choisi, et une rubrique **check.dependsOn**.
Ne pas oublier de mettre à jour le fichier gradle (cliquer sur l'icone 'éléphant' apparaissant dans la zone client de la fenêtre du fichier build.gradle)
- Exécuter le build par les moyens de votre choix (ligne de commande, onglet Gradle dans IDE, ou bien en poussant le code sur Github), et vérifier que la construction est bien stoppée.
- Fusionner (merge) la branche lorsque l'opération est terminée.
- Supprimer la branche.
- Enregistrer, versionner.

¹⁷ https://docs.gradle.org/current/userguide/jacoco_plugin.html
<https://medium.com/codeops/code-coverage-with-gradle-and-jacoco-8b2e7d580d2a>