**-Hamed Ouattara**

# Description des tests

## Load data from memory Tests

Test#1 loadDatafromMemory\_AreDatasIntact : Ce test vise a vérifier si les données sont bien ajoutées, enregistrées et que les données sont bien intactes tout en s’assurant que le nombre total de points est exacte.

Test#2 loadDotsFromMemory\_Empty\_Exception : Si le fichier de points est vide, normalement la fonction devrait lever une exception de type runTime. Le but du programme est de trouver des carrés grâce a des points donc je trouve qu’arrêter le programme pour cela est pertinent.

Test#3 loadDotsFromMemory\_InvalidInts\_Ignored : Ce texte à pour but de vérifier si la fonction filtre bien les données en gardant juste les points qui sont constitués de nombre. Je présume que nous avons un graph de nombre pas alphanumérique.

Test#4 loadDotsFromMemory\_AllInvalid\_Exception : imaginons que nous avons un fichier avec que des points de lettres. Ce test vérifie si la fonction lèvera une exception.

Test #5 loadDotsFromMemory\_UwantedFormat\_Ignored : encore une fois on ne sait jamais, ça se peut que les points soie générer par un humain et comme l’erreur est humaine, je test voir si le format des points est correct avant de les rajouté.

## Tests AddDot

Test #6 AddDot\_SingleDot\_StoredCorrectly : je fais un premier test simple pour voir si la méthode est capable d’enregistrer les points sans problème. Pour commencer je vérifie si le nombre de point est exacte puis je vérifie si les coordonnées sont correctes.

Test #7 AddDot\_MultipleDots\_StoredInOrder : ce test vise tout simplement à vérifier si les points sont ajoutés dans le bon ordre. Je commence par en ajouter les points dans le tableau puis je vérifie les coordonnées du premier point.

Test #8 AddDot\_DuplicateDots\_Accepted : puisque rien n’est spécifié dans l’énoncé, je déduis qu’au chargement des points ceux en double sont acceptés. Mon code n’en tient pas compte de toute façon, mais par contre vérifie si les points ne sont pas semblables pour la formation des carrés. Ce test vérifie tous simplement s’il n’y a pas de problème à ce niveau.

Test #9 AddDot\_InitialEmptyArray\_AddsOneDot : j’ai voulu tester un cas limite classique où aucun point n’a encore été ajouté et on appelle addDot . Je commence par vérifier si le tableau est bien vide, j’ajoute un point pour vois s’il a été rajouté.

## Test Find Square Brute Force

Test #10 FindSquaresBruteForce\_NoSquare\_Found : ce test à pour but de vérifier s’il n’y a effectivement pas de combinaisons de points qui forment un carré. Je donne des points qui ne forme pas un carré a la fonction et je vérifie si elle ne trouve pas de carré.

Test #11 FindSquaresBruteForce\_TwoValidSquares\_Found : ce test vérifie s’il y’a deux ou plusieurs carrés ayant les mêmes dimensions mais différentes coordonnées, celle-ci ne seront pas considéré comme des carrés semblables et seront rajoutés dans la liste de carrés.

Test #12 FindSquaresBruteForce\_LessThanFourDots\_Throws : on se sait jamais, on pourrait tomber sur un fichier qui contient que trois points, donc je trouve raisonnable de lever une exception pour cela.

Test #13 FindSquaresBruteForce\_AllDotsIdentical\_NoSquare : pour former un carré il faut quatre points différents. Donc ce test vérifie que la fonction ignore ce genre de combinaisons de points.

Test #14 FindSquaresBruteForce\_LinearDots\_NoSquare : l’ors de la recherche, on peut tomber sur des combinaisons de point qui ont les mêmes longueurs mais ne forme pas un carré. Ce test vise simplement à vérifier que la fonction ne considère pas des droites qui sont l’une à côté de l’autre.

Test #15 FindSquaresBruteForce\_NonAlignedDots\_NoSquare : même test que le précédent mais cette fois ci avec des côtés qui ne sont pas alignés.

Test # 16 FindSquaresBruteForce\_OddNumberOfDots\_OneSquare\_UsingIsSquareInList : il peut souvent avoir un nombre impair de points, donc de test vise à vérifier que la fonction est quand même capable de trouver les points

## Test Find Square Optimized

Test #17 FindSquaresOptimized\_ThreDistinctSquares\_Found : ce test vise a vérifier si la fonction est capable d’ajouter plusieurs points. Les points générés forme trois carrés l’un à la suite de l’autre.

Test #18 FindSquaresOptimized\_LessThanFourDots\_Throws : un carré est formé de 4 points différends donc au cas ou nous avons juste trois points dans le fichier, la fonction devrait lancer une exception.

Test #19 FindSquaresOptimized\_OddNumberOfDots\_NoSquare : ce test rajoute un nombre de impairs qui ne forment pas de carrés. Cela test si la fonction ne crée pas des carrés par hasard.

Test #20 FindSquaresOptimized\_UnorderedPoints\_StillDetectsSquare : disons que la fonction reçois des points qui forment un carré mais quis sont en désordre. Ce test vise à tester ce cas dans la fonction.

Test #21 FindSquaresOptimized\_TwoDistinctSquares\_FoundUslessDots : il est possible d’avoir des points inutiles dans la liste de points. Ce test vise a vérifier si la fonction est capable de les filtrer et de trouver les points qui forment un carré.

### Source

Pour générer des points conditionnées j’ai eu recours à ChatGpt.