

IFT 3913 - Tâche 2 – Graphhopper, tests unitaires automatiques

Binôme composé de :

Papa Moussa Diabate

Laurent Polzin

- Choix des classes :

Il faut dans un premier temps trouver des classes qui ne sont pas couvertes à 100%. Ensuite, il faut étudier si la classe est difficile à tester (des requêtes client-serveur ou la gestion de la mémoire RAM par exemple). Finalement, les classes qui ont été sélectionnées sont :

- **Transfers** (dans reader-gtfs). Cette classe s'occupe de trouver des correspondances entre un arrêt et une route donnée.
- **GHUtility** (dans core). Cette classe utilitaire offre en majorité des fonction utiles pour le test unitaire ou le debugging.
- **ArrayUtil** (dans core). Il s'agit ici d'une classe utilitaire pour les listes, on y trouve par exemple des fonctions qui retire les duplications, qui inverse une liste, qui mélange une liste, et d'autres encore.

- Les cas de tests :

- **Transfers** : la méthode testée (seule qui n'est pas 100% covered) est `getTransferFromStop`

```

List<Transfer> getTransfersFromStop(String fromStopId, String fromRouteId) {
    final List<Transfer> allOutboundTransfers = transfersFromStop.getDefault(fromStopId, Collections.emptyList());
    final Map<String, List<Transfer>> byToStop = allOutboundTransfers.stream()
        .filter(t -> t.transfer_type == 0 || t.transfer_type == 2)
        .filter(t -> t.from_route_id == null || (fromRouteId != null && fromRouteId.equals(t.from_route_id)))
        .collect(Collectors.groupingBy(t -> t.to_stop_id));
    final List<Transfer> result = new ArrayList<>();
    byToStop.forEach((toStop, transfers) -> {
        routesByStop.getDefault(toStop, Collections.emptySet()).forEach(toRouteId -> {
            final Transfer mostSpecificRule = findMostSpecificRule(transfers, fromRouteId, toRouteId);
            final Transfer myRule = new Transfer();
            myRule.to_route_id = toRouteId;
            myRule.from_route_id = fromRouteId;
            myRule.to_stop_id = mostSpecificRule.to_stop_id;
            myRule.from_stop_id = mostSpecificRule.from_stop_id;
            myRule.transfer_type = mostSpecificRule.transfer_type;
            myRule.min_transfer_time = mostSpecificRule.min_transfer_time;
            myRule.from_trip_id = mostSpecificRule.from_trip_id;
            myRule.to_trip_id = mostSpecificRule.to_trip_id;
            result.add(myRule);
        });
    });
    return result;
}

```

- TestFiltersGetTransferFromStop : On veut s'assurer que les deux filtres fonctionnent : uniquement les transferts de types 0 et 2 passent (4 types de transferts, de 0 à 3. 0 est le transfert recommandé, 1 il n'y a pas de correspondance permise, 2 transfert possible mais avec un temps minimum pour l'effectuer, 3 le transfert est coordonné entre les différentes instances (train et bus par exemple). On va ajouter une ligne dans transfers.txt, un fichier parmi d'autre dans un dossier permettant d'instancier plusieurs données gtfs de tests. On va spécifier un transfert_type = 1 par exemple. La variable byToStop devrait donc être vide, et result restera vide. On s'attend donc à une liste vide.

```

/// We here assure that the filter is working as expected, excluding transfer_type = 1.
@Test
public void testFiltersGetTransfersFromStop() {
    // 1st TEST : transfer_type=1 not included : BULLFROG,BULLFROG,AB,,1,100
    // Line added in transfers.txt, if changed the test won't pass
    // .filter(t -> t.transfer_type == 0 || t.transfer_type == 2) => FALSE
    // .filter(t -> t.from_route_id == null || fromRouteId.equals(t.from_route_id)) => FALSE
    assertEquals(Collections.emptyList(), sampleFeed.getTransfersFromStop(fromStopId:"BULLFROG", fromRouteId:"BULLFROG"));
}

```

- TestGetTransfersFromStopNotExistingFromStopId : on veut désormais jouer sur les paramètres. Commençons avec fromStopId. On va prendre un nom qui n'est pas dans les données gtfs de test, par exemple "TOTO". On va aussi faire varier fromRouteId pour voir la connivence des 2 paramètres. La première ligne de la méthode spécifie que si fromStopId n'est pas trouvé, la valeur par défaut est une liste vide. On s'attend donc que pour les 4 cas de tests, getTransfersFromStop nous renvoie une liste vide. Effectivement, même si la route d'où on vient existe (AB), si notre arrêt est imaginaire, la méthode ne peut nous renvoyer des correspondances possibles.

```

/// We assure that if fromStopID is wrong, not an existing stop or a null argument,
/// the function will return an empty list, as allOutboundTransfers is empty from the beginning.
@Test
public void testGetTransfersFromStopNotExistingFromStopID() {
    // 2nd TEST : not an existing fromStopID, or even null
    assertEquals(Collections.emptyList(), sampleFeed.getTransfersFromStop(fromStopId:"TOTO", fromRouteId:"AB"));
    assertEquals(Collections.emptyList(), sampleFeed.getTransfersFromStop(fromStopId:null, fromRouteId:"TOTO"));
    assertEquals(Collections.emptyList(), sampleFeed.getTransfersFromStop(fromStopId:"TOTO", fromRouteId:null));
    assertEquals(Collections.emptyList(), sampleFeed.getTransfersFromStop(fromStopId:null, fromRouteId:null));
}

```

- TestGetTransfersFromStopNotExistingFromRouteId : ici, on veut étudier le cas où notre arrêt est correct mais la route sur laquelle on est n'existe pas. Le test ne passe pas ! Effectivement, on trouve dans la méthode fromRouteId.equals(...), ne prenant pas en compte la possibilité d'un nullPointer ! J'ai dû donc ajouter (fromRouteId != null && fromRouteId.equals(...)) pour pallier cette erreur. Notre arrêt étant correct, on nous propose des correspondances possibles. Effectivement, fromRouteId est facultatif.

```

@Test
public void testGetTransfersFromStopNotExistingFromRouteID() {
    // 3rd TEST : not an existing fromRouteID, but existing FromStopID
    assertNotEquals(Collections.emptyList(), sampleFeed.getTransfersFromStop(fromStopId:"BEATTY_AIRPORT", fromRouteId:null));
    // if fromRouteId == null, the program fails (fromRouteId.equals in the filter) !
    assertNotEquals(Collections.emptyList(), sampleFeed.getTransfersFromStop(fromStopId:"BEATTY_AIRPORT", fromRouteId:"titi"));
}

```

- TestGetTransferFromStopExistingToStop : on s'intéresse désormais au cas où on vient du même arrêt mais que nous ne sommes pas sur la même route. Il doit exister des correspondances (d'après les données gtfs, fichiers stops.txt, routes.txt, trips.txt...) pour BEATTY_AIRPORT. Dans le premier cas, on lui donne un fromStopId existant, ainsi, le programme peut retourner précisément le temps minimum pour faire la correspondance. Or, dans le second cas, on lui fournit une fromStopId inexistant, le programme va lui associer un temps minimum par défaut, 660 secondes. On ne devrait donc pas avoir les mêmes correspondances.

```

@Test
public void testGetTransfersFromStopGetExistingToStop() {
    // 4th test : testing different fromRouteId, transfers always existing if valid fromStopId
    List<Transfer> existingFromRouteID = sampleFeed.getTransfersFromStop(fromStopId:"BEATTY_AIRPORT", fromRouteId:"AB");
    List<Transfer> notExistingFromRouteID = sampleFeed.getTransfersFromStop(fromStopId:"BEATTY_AIRPORT", fromRouteId:"TOTO");
    assertNotEquals(Collections.emptyList(), existingFromRouteID);
    assertNotEquals(Collections.emptyList(), notExistingFromRouteID);
    // fromRouteId can be an imaginary route but cannot be null. We can admit coming from an unknown route as a valid case
    assertNotEquals(existingFromRouteID, notExistingFromRouteID);
}

```

- **GHUtility** : nous allons tester la méthode *getCommonNode*
Cette méthode retourne le nœud commun entre deux arêtes.

```

/**
 * @return the common node of two edges
 * @throws IllegalArgumentException if one of the edges doesn't exist or is a loop or the edges
 *         aren't connected at exactly one distinct node
 */
public static int getCommonNode(BaseGraph baseGraph, int edge1, int edge2) {
    EdgeIteratorState e1 = baseGraph.getEdgeIteratorState(edge1, Integer.MIN_VALUE);
    EdgeIteratorState e2 = baseGraph.getEdgeIteratorState(edge2, Integer.MIN_VALUE);
    if (e1.getBaseNode() == e1.getAdjNode())
        throw new IllegalArgumentException("edge1: " + edge1 + " is a loop at node " + e1.getBaseNode());
    if (e2.getBaseNode() == e2.getAdjNode())
        throw new IllegalArgumentException("edge2: " + edge2 + " is a loop at node " + e2.getBaseNode());

    if ((e1.getBaseNode() == e2.getBaseNode() && e1.getAdjNode() == e2.getAdjNode()) || (e1.getBaseNode() == e2.getAdjNode() && e1.getAdjNode() == e2.getBaseNode()))
        throw new IllegalArgumentException("edge1: " + edge1 + " and edge2: " + edge2 + " form a circle");
    else if (e1.getBaseNode() == e2.getBaseNode() || e1.getBaseNode() == e2.getAdjNode())
        return e1.getBaseNode();
    else if (e1.getAdjNode() == e2.getBaseNode() || e1.getAdjNode() == e2.getAdjNode())
        return e1.getAdjNode();
    else
        throw new IllegalArgumentException("edge1: " + edge1 + " and edge2: " + edge2 + " aren't connected");
}

```

- TestCommonNodeTrue1 : Il s'agit d'un simple test avec un graphe de 3 nœuds (0,1,2) et deux arêtes [(0,1) et (0,2)]. Nous voulons vérifier que la valeur retournée est bien 0, si nous lui passons les (0,1) et (0,2) en paramètres.
- TestCommonNodeTrue2 : Il s'agit d'un test plus ou moins similaire, mais de 3 nœuds (0,1,2) et deux arêtes [(0,1) et (2,1)]. Nous voulons vérifier que la valeur retournée est bien 1, si nous lui passons les (0,1) et (2,1) en paramètres.
- TestCommonNodeFalse : Nous cherchons à tester le cas où les deux arêtes passées en paramètres n'ont pas de nœud en commun. On instancie un graphe de 4 nœuds (0,1,2,3) avec trois arêtes [(0,1), (0,2) et (2,3)]. On appelle la fonction avec les arêtes (0,1) et (2,3) et nous vérifions que ça lance bien et bel une exception étant donné que les deux arêtes n'ont pas de nœud en commun.
- TestCircularEdge() : Nous voulons vérifier le comportement de la fonction lorsque nous lui passons une ou des arêtes de la forme (a,b) tel que a=b. Celle-ci devrait lever une exception. On instancie un graphe de 2 nœuds (0,1) avec deux arêtes [(0,1), (0,0)]. Étant donné la présence d'une arête circule une exception va être levée.
- TestEdgeThatFormACircle : Nous cherchons à tester le cas où les deux arêtes passées en paramètres forment un cercle. On instancie un graphe de 2 nœuds (0,1) avec trois arêtes [(0,1), (1,0)]. On appelle la fonction avec les arêtes (0,1) et (1,0) et nous vérifions que ça lance bien et bel une

exception étant donné que les deux arêtes forment un cercle.

- TestCommonNodeWithNonExistentEdge : Vérifie que si on appelle la méthode avec un identifiant d'arête qui n'existe pas ça lève une exception. On instancie un graphe de 2 nœuds (0,1) avec une arête [(0,1)] avec un identifiant d'arête qui n'existe pas 14. Et on appelle la fonction avec (0,1) et 14 ce qui devrait retourner une exception.
 - TestCommonNodeSymmetry : Vérifie que si on appelle la méthode avec les mêmes arêtes mais en changeant juste l'ordre le résultat reste inchangé. On instancie un graphe de 3 nœuds (0,1,2) avec deux arêtes [(0,1) (1,2)]. On vérifie que lorsqu'on appelle la méthode comme ceci : `getCommonNode(graph, (0,1), (1,2))` et `getCommonNode(graph, (1,2), (0,1))` le résultat est le même.
 -
- **ArrayUtil** : nous allons tester les méthodes `applyOrder`, `subList`, `calcSortOrder` et `removeConsecutiveDuplicates`.

```
public static int[] applyOrder(int[] arr, int[] order) { 2 usages  easbar
    if (order.length > arr.length)
        throw new IllegalArgumentException("sort order must not be shorter than array");
    int[] result = new int[order.length];
    for (int i = 0; i < result.length; ++i)
        result[i] = arr[order[i]];
    return result;
}
```

```
public static IntArrayList subList(IntArrayList list, int fromIndex, int toIndex) { 2 Andi
    IntArrayList result = new IntArrayList(expectedElements: toIndex - fromIndex);
    for (int i = fromIndex; i < toIndex; i++)
        result.add(list.get(i));
    return result;
}
```

```

public static int[] calcSortOrder(final int[] arr1, final int[] arr2, int length) { 7 usages
    if (arr1.length < length || arr2.length < length)
        throw new IllegalArgumentException("Arrays must not be shorter than given length");
    IndirectComparator comp = (int indexA, int indexB) -> {
        final int arr1cmp = Integer.compare(arr1[indexA], arr1[indexB]);
        return arr1cmp != 0 ? arr1cmp : Integer.compare(arr2[indexA], arr2[indexB]);
    };
    return IndirectSort.mergesort(start:0, length, comp);
}

```

```

public static int removeConsecutiveDuplicates(int[] arr, int end) { 7 usages easbar
    if (end < 0)
        throw new IllegalArgumentException("end less than 0");
    if (end == 0)
        return 0;
    int curr = 0;
    for (int i = 1; i < end; ++i) {
        if (arr[i] != arr[curr])
            arr[++curr] = arr[i];
    }
    return curr + 1;
}

```

- TestSublist0 : Nous cherchons à tester le cas où on tente d'extraire une sous-liste à partir d'une liste vide. On utilise un tableau vide [] et on tente d'extraire une sous-liste entre les indices 3 et 5. On appelle la fonction avec cette liste vide et les indices (3,5) et nous vérifions que ça lance bien une exception étant donné qu'on ne peut pas extraire de sous-liste d'une liste vide avec des indices qui dépassent sa taille.

- TestSublist1 : Il s'agit d'un simple test avec un tableau de 10 éléments [1,2,3,4,5,6,7,8,9,10] et une extraction de sous-liste entre les indices 3 et 5. Nous voulons vérifier que la valeur retournée est bien [4,5], si nous lui passons le tableau et les indices (3,5) en paramètres.
- TestSublist2 : Il s'agit d'un test similaire, mais avec les indices de début et de fin égaux. On utilise le même tableau [1,2,3,4,5,6,7,8,9,10] et on extrait une sous-liste entre les indices (0,0). Nous voulons vérifier que la valeur retournée est bien un tableau vide [], représentant une plage d'extraction nulle.
- TestApplyOrder2 : Nous cherchons à tester le cas où le tableau d'ordre passé en paramètre est plus grand que le tableau source. On utilise un tableau source [1,2,3,4,5,6,7,8,9,10] de taille 10 et un tableau d'ordre [1,2,3,4,5,6,7,8,9,10,11] de taille 11. On appelle la fonction avec ces deux tableaux et nous vérifions que ça lance bien une exception étant donné que le tableau d'ordre est plus grand que le tableau source.
- TestCalcSortOrder2 : Nous voulons vérifier le comportement de la fonction lorsque le paramètre length passé en argument est plus grand que la taille des tableaux. On utilise deux tableaux de taille 10 : [1,2,3,4,5,6,7,8,9,10] et [12,42,387,41,45,69,71,58,98,10], avec un paramètre length=11. Étant donné que la longueur spécifiée dépasse la taille des tableaux, une exception va être levée.
- TestCalcSortOrder3 : Nous cherchons à tester le cas où les deux tableaux passés en paramètres sont de tailles inégales. On utilise un premier tableau [1,2,3,4,5,6,7,8,9,10,11] de taille 11 et un deuxième tableau [12,42,387,41,45,69,71,58,98,10] de taille 10. On appelle la fonction avec ces deux tableaux et nous vérifions que ça lance bien une exception étant donné que les tableaux ont des tailles différentes.
- TestRemoveConsecutiveDuplicates : Nous voulons vérifier le comportement de la fonction lorsque le paramètre end passé en argument est inférieur à zéro. On utilise un tableau [1,2,3,4,5,6,7,8,9,10] et un paramètre end=-1. Étant donné que l'indice de fin est négatif, une exception va être levée.

- Pitest et analyse de mutation des tests :

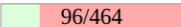
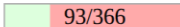
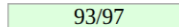
Pitest est un module permettant de tester la résistance aux bugs d'un programme. Il va générer des petites erreurs au sein même du programme. Plus il y a de mutants "vivants", moins le test est rigoureux. Autrement dit, il faut que le programme test tue le plus de mutants possibles. Pour utiliser pitest, il faut commencer par l'ajouter dans le pom.xml. On a le choix entre l'ajouter au pom.xml à la racine du projet et faire des appels sur les sous-répertoires, ou l'ajouter aux deux pom.xml, celui de la classe **core** et celui de la classe **reader-gtfs**. On va ici l'ajouter au pom.xml de la racine et faire des appels sur les sous-répertoires.

Voici les scores de mutation obtenus, avant et après l'ajout de nos tests :

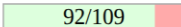
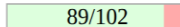
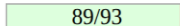
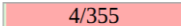
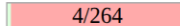
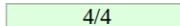
Pit Test Coverage Report

Package Summary

com.graphhopper.util

Number of Classes	Line Coverage	Mutation Coverage	Test Strength
2	21% 	25% 	96% 

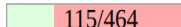
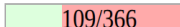
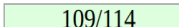
Breakdown by Class

Name	Line Coverage	Mutation Coverage	Test Strength
ArrayUtil.java	84% 	87% 	96% 
GHUtility.java	1% 	2% 	100% 

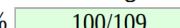
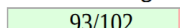
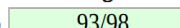
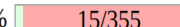
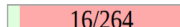
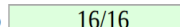
Pit Test Coverage Report

Package Summary

com.graphhopper.util

Number of Classes	Line Coverage	Mutation Coverage	Test Strength
2	25% 	30% 	96% 

Breakdown by Class

Name	Line Coverage	Mutation Coverage	Test Strength
ArrayUtil.java	92% 	91% 	95% 
GHUtility.java	4% 	6% 	100% 

Pour la classe GHUtility.java, il n'y a pas de nouveaux mutants.

Pour la classe ArrayUtil.java il y a un nouveau mutant. Qui n'était pas couvert avant l'ajout des tests mais qui a survécu après l'ajout des tests.

Voir la ligne 232

Mutations	Mutations
28 1. removed call to java/util/Arrays::fill - KILLED	28 1. removed call to java/util/Arrays::fill - KILLED
40 1. replaced return value with null for com/graphhopper/util/ArrayUtil::constant - KILLED	40 1. replaced return value with null for com/graphhopper/util/ArrayUtil::constant - KILLED
49 1. replaced return value with null for com/graphhopper/util/ArrayUtil::zero - NO COVERAGE	49 1. replaced return value with null for com/graphhopper/util/ArrayUtil::zero - NO COVERAGE
56 1. replaced return value with null for com/graphhopper/util/ArrayUtil::iota - KILLED	56 1. replaced return value with null for com/graphhopper/util/ArrayUtil::iota - KILLED
63 1. Replaced integer subtraction with addition - KILLED	63 1. Replaced integer subtraction with addition - KILLED
64 1. Replaced integer subtraction with addition - KILLED	64 1. Replaced integer subtraction with addition - KILLED
65 1. changed conditional boundary - KILLED	65 1. changed conditional boundary - KILLED
66 1. negated conditional - KILLED	66 1. negated conditional - KILLED
67 1. Replaced integer addition with subtraction - KILLED	67 1. Replaced integer addition with subtraction - KILLED
74 1. replaced return value with null for com/graphhopper/util/ArrayUtil::range - KILLED	74 1. replaced return value with null for com/graphhopper/util/ArrayUtil::range - KILLED
75 1. Replaced integer addition with subtraction - KILLED	75 1. Replaced integer addition with subtraction - KILLED
83 1. replaced return value with null for com/graphhopper/util/ArrayUtil::permutation - KILLED	83 1. replaced return value with null for com/graphhopper/util/ArrayUtil::permutation - KILLED
89 1. negated conditional - KILLED	89 1. negated conditional - KILLED
90 1. changed conditional boundary - KILLED	90 1. changed conditional boundary - KILLED
91 1. negated conditional - KILLED	91 1. negated conditional - KILLED
92 1. replaced boolean return with true for com/graphhopper/util/ArrayUtil::isPermutation - KILLED	92 1. replaced boolean return with true for com/graphhopper/util/ArrayUtil::isPermutation - KILLED
93 1. removed call to com/carrotsearch/hppc/BitSet::set - KILLED	93 1. removed call to com/carrotsearch/hppc/BitSet::set - KILLED
94 1. replaced boolean return with false for com/graphhopper/util/ArrayUtil::isPermutation - KILLED	94 1. replaced boolean return with false for com/graphhopper/util/ArrayUtil::isPermutation - KILLED
95 1. changed increment from 1 to 1 - KILLED	95 1. changed increment from 1 to 1 - KILLED
104 1. Replaced integer subtraction with addition - KILLED	104 1. Replaced integer subtraction with addition - KILLED
104 2. changed conditional boundary - SURVIVED Coverage tests	104 2. changed conditional boundary - SURVIVED Coverage tests
104 3. negated conditional - KILLED	104 3. negated conditional - KILLED
104 4. changed increment from 1 to 1 - KILLED	104 4. changed increment from 1 to 1 - KILLED
110 1. replaced return value with null for com/graphhopper/util/ArrayUtil::reverse - KILLED	110 1. replaced return value with null for com/graphhopper/util/ArrayUtil::reverse - KILLED
117 1. Replaced integer division with multiplication - KILLED	117 1. Replaced integer division with multiplication - KILLED
118 1. changed conditional boundary - SURVIVED Coverage tests	118 1. changed conditional boundary - SURVIVED Coverage tests
119 1. negated conditional - KILLED	119 1. negated conditional - KILLED
120 1. Replaced integer addition with subtraction - KILLED	120 1. Replaced integer addition with subtraction - KILLED
124 1. replaced return value with null for com/graphhopper/util/ArrayUtil::shuffle - KILLED	124 1. replaced return value with null for com/graphhopper/util/ArrayUtil::shuffle - KILLED
133 1. negated conditional - KILLED	133 1. negated conditional - KILLED
134 1. changed conditional boundary - KILLED	134 1. changed conditional boundary - KILLED
135 1. negated conditional - KILLED	135 1. negated conditional - KILLED
136 1. negated conditional - KILLED	136 1. negated conditional - KILLED
137 1. changed conditional boundary - KILLED	137 1. changed conditional boundary - KILLED
138 1. negated conditional - KILLED	138 1. negated conditional - KILLED
140 1. changed increment from 1 to 1 - KILLED	140 1. changed increment from 1 to 1 - KILLED
141 1. replaced int return with 0 for com/graphhopper/util/ArrayUtil::removeConsecutiveDuplicates - KILLED	141 1. replaced int return with 0 for com/graphhopper/util/ArrayUtil::removeConsecutiveDuplicates - KILLED
142 1. Replaced integer addition with subtraction - KILLED	142 1. Replaced integer addition with subtraction - KILLED
149 1. negated conditional - KILLED	149 1. negated conditional - KILLED
151 1. replaced return value with null for com/graphhopper/util/ArrayUtil::withoutConsecutiveDuplicates - KILLED	151 1. replaced return value with null for com/graphhopper/util/ArrayUtil::withoutConsecutiveDuplicates - KILLED
152 1. removed call to com/carrotsearch/hppc/IntArrayList::add - KILLED	152 1. removed call to com/carrotsearch/hppc/IntArrayList::add - KILLED
153 1. negated conditional - KILLED	153 1. negated conditional - KILLED
154 1. changed conditional boundary - KILLED	154 1. changed conditional boundary - KILLED
155 1. negated conditional - KILLED	155 1. negated conditional - KILLED
156 1. removed call to com/carrotsearch/hppc/IntArrayList::add - KILLED	156 1. removed call to com/carrotsearch/hppc/IntArrayList::add - KILLED
157 1. replaced return value with null for com/graphhopper/util/ArrayUtil::withoutConsecutiveDuplicates - KILLED	157 1. replaced return value with null for com/graphhopper/util/ArrayUtil::withoutConsecutiveDuplicates - KILLED
160 1. changed conditional boundary - KILLED	160 1. changed conditional boundary - KILLED
161 1. negated conditional - KILLED	161 1. negated conditional - KILLED
162 1. replaced return value with null for com/graphhopper/util/ArrayUtil::calcSortOrder - KILLED	162 1. replaced return value with null for com/graphhopper/util/ArrayUtil::calcSortOrder - KILLED
172 1. negated conditional - KILLED	172 1. negated conditional - KILLED
173 1. changed conditional boundary - KILLED	173 1. changed conditional boundary - KILLED
174 1. negated conditional - KILLED	174 1. negated conditional - KILLED
175 1. changed conditional boundary - KILLED	175 1. changed conditional boundary - KILLED
188 1. negated conditional - KILLED	188 1. negated conditional - KILLED
192 1. replaced int return with 0 for com/graphhopper/util/ArrayUtil::lambdaCalcSortOrder\$0 - KILLED	192 1. replaced int return with 0 for com/graphhopper/util/ArrayUtil::lambdaCalcSortOrder\$0 - KILLED
194 1. replaced return value with null for com/graphhopper/util/ArrayUtil::calcSortOrder - KILLED	194 1. replaced return value with null for com/graphhopper/util/ArrayUtil::calcSortOrder - KILLED
202 1. changed conditional boundary - KILLED	202 1. changed conditional boundary - KILLED
203 1. negated conditional - KILLED	203 1. negated conditional - KILLED
204 1. changed conditional boundary - KILLED	204 1. changed conditional boundary - KILLED
205 1. negated conditional - KILLED	205 1. negated conditional - KILLED
206 1. changed conditional boundary - KILLED	206 1. changed conditional boundary - KILLED
207 1. replaced return value with null for com/graphhopper/util/ArrayUtil::applyOrder - KILLED	207 1. replaced return value with null for com/graphhopper/util/ArrayUtil::applyOrder - KILLED
217 1. removed call to java/util/Arrays::fill - KILLED	217 1. removed call to java/util/Arrays::fill - KILLED
218 1. negated conditional - KILLED	218 1. negated conditional - KILLED
220 1. changed conditional boundary - KILLED	220 1. changed conditional boundary - KILLED
221 1. replaced return value with null for com/graphhopper/util/ArrayUtil::invert - KILLED	221 1. replaced return value with null for com/graphhopper/util/ArrayUtil::invert - KILLED
226 1. negated conditional - NO COVERAGE	226 1. negated conditional - NO COVERAGE
227 1. changed conditional boundary - NO COVERAGE	227 1. changed conditional boundary - NO COVERAGE
228 1. replaced return value with null for com/graphhopper/util/ArrayUtil::invert - NO COVERAGE	228 1. replaced return value with null for com/graphhopper/util/ArrayUtil::invert - NO COVERAGE
232 1. negated conditional - NO COVERAGE	232 1. negated conditional - NO COVERAGE
233 1. changed conditional boundary - NO COVERAGE	233 1. changed conditional boundary - NO COVERAGE
234 1. removed call to com/carrotsearch/hppc/IntArrayList::add - NO COVERAGE	234 1. removed call to com/carrotsearch/hppc/IntArrayList::add - NO COVERAGE
235 1. replaced return value with null for com/graphhopper/util/ArrayUtil::subList - NO COVERAGE	235 1. replaced return value with null for com/graphhopper/util/ArrayUtil::subList - NO COVERAGE
244 1. negated conditional - KILLED	244 1. negated conditional - KILLED
245 1. Replaced integer addition with subtraction - SURVIVED Coverage tests	245 1. Replaced integer addition with subtraction - SURVIVED Coverage tests
246 1. replaced return value with null for com/graphhopper/util/ArrayUtil::merge - KILLED	246 1. replaced return value with null for com/graphhopper/util/ArrayUtil::merge - KILLED
248 1. Replaced integer addition with subtraction - KILLED	248 1. Replaced integer addition with subtraction - KILLED
250 1. negated conditional - KILLED	250 1. negated conditional - KILLED
251 1. changed conditional boundary - KILLED	251 1. changed conditional boundary - KILLED
252 1. changed increment from 1 to 1 - KILLED	252 1. changed increment from 1 to 1 - KILLED
253 1. changed increment from 1 to 1 - KILLED	253 1. changed increment from 1 to 1 - KILLED
254 1. changed increment from 1 to 1 - KILLED	254 1. changed increment from 1 to 1 - KILLED
255 1. negated conditional - KILLED	255 1. negated conditional - KILLED
256 1. removed call to java/lang/System::arraycopy - KILLED	256 1. removed call to java/lang/System::arraycopy - KILLED
257 1. Replaced integer subtraction with addition - KILLED	257 1. Replaced integer subtraction with addition - KILLED
258 1. Replaced integer addition with subtraction - KILLED	258 1. Replaced integer addition with subtraction - KILLED
259 1. Replaced integer subtraction with addition - KILLED	259 1. Replaced integer subtraction with addition - KILLED
260 1. removed call to java/lang/System::arraycopy - KILLED	260 1. removed call to java/lang/System::arraycopy - KILLED
261 1. Replaced integer subtraction with addition - KILLED	261 1. Replaced integer subtraction with addition - KILLED
262 1. Replaced integer addition with subtraction - KILLED	262 1. Replaced integer addition with subtraction - KILLED
264 1. replaced return value with null for com/graphhopper/util/ArrayUtil::merge - KILLED	264 1. replaced return value with null for com/graphhopper/util/ArrayUtil::merge - KILLED

Nous remarquons que le mutant est le remplacement de la soustraction par l'addition dans la méthode subList (voir plus haut le code de subList). Or subList n'était pas testé, raison pour laquelle le mutant n'était pas couvert. Mais après avoir inséré des tests pour subList on voit que le mutant a survécu (il passe quand même les tests). Car l'addition apporte simplement un changement au nombre d'éléments attendus du tableau qui n'est pas la taille réelle du tableau, car la taille c'est le nombre d'éléments dedans, qui sera inchangé car on copie à partir de fromIndex jusqu'à toIndex. Ce qui fait qu'on aura toujours le même nombre d'éléments et vu que la taille est le nombre d'éléments les tests suivants réussissent quand même. Les tableaux ont les mêmes éléments et la même taille.

com.graphhopper.util.ArrayUtilTest.[engine:junit-

jupiter]/[class:com.graphhopper.util.ArrayUtilTest]/[method:TestSublist2()]
com.graphhopper.util.ArrayUtilTest.[engine:junit-
jupiter]/[class:com.graphhopper.util.ArrayUtilTest]/[method:TestSublist1()]
com.graphhopper.util.ArrayUtilTest.[engine:junit-
jupiter]/[class:com.graphhopper.util.ArrayUtilTest]/[method:TestSublist0()]

Pit Test Coverage Report

Package Summary

com.graphhopper.gtfs

Number of Classes	Line Coverage	Mutation Coverage	Test Strength
1	77% <div><div>77/100</div></div>	51% <div><div>32/63</div></div>	60% <div><div>32/53</div></div>

Breakdown by Class

Name	Line Coverage	Mutation Coverage	Test Strength
Transfers.java	77% <div><div>77/100</div></div>	51% <div><div>32/63</div></div>	60% <div><div>32/53</div></div>

Pit Test Coverage Report

Package Summary

com.graphhopper.gtfs

Number of Classes	Line Coverage	Mutation Coverage	Test Strength
1	99% <div><div>99/100</div></div>	59% <div><div>38/64</div></div>	59% <div><div>38/64</div></div>

Breakdown by Class

Name	Line Coverage	Mutation Coverage	Test Strength
Transfers.java	99% <div><div>99/100</div></div>	59% <div><div>38/64</div></div>	59% <div><div>38/64</div></div>

Pour la classe Transfers, voici les mutations avant et après l'ajout des tests :

Mutations		Mutations	
37	1: replaced return value with "" for com/graphhopper/gtfs/Transfers::lambda\$new\$9 - KILLED	37	1: replaced return value with "" for com/graphhopper/gtfs/Transfers::lambda\$new\$9 - KILLED
38	1: replaced return value with "" for com/graphhopper/gtfs/Transfers::lambda\$new\$1 - SURVIVED Coverage Tests	38	1: replaced return value with "" for com/graphhopper/gtfs/Transfers::lambda\$new\$1 - SURVIVED Coverage Tests
40	1: replaced return value with "" for com/graphhopper/gtfs/Transfers::lambda\$new\$2 - SURVIVED Coverage Tests	40	1: replaced return value with "" for com/graphhopper/gtfs/Transfers::lambda\$new\$2 - SURVIVED Coverage Tests
41	1: replaced return value with "" for com/graphhopper/gtfs/Transfers::lambda\$new\$3 - SURVIVED Coverage Tests	41	1: replaced return value with "" for com/graphhopper/gtfs/Transfers::lambda\$new\$3 - SURVIVED Coverage Tests
45	1: replaced return value with Stream.empty for com/graphhopper/gtfs/Transfers::explodeTransfers - KILLED	45	1: replaced return value with Stream.empty for com/graphhopper/gtfs/Transfers::explodeTransfers - KILLED
48	1: negated conditional - KILLED	48	1: negated conditional - KILLED
49	1: replaced return value with Stream.empty for com/graphhopper/gtfs/Transfers::lambda\$explodeTransfers\$7 - KILLED	49	1: replaced return value with Stream.empty for com/graphhopper/gtfs/Transfers::lambda\$explodeTransfers\$7 - KILLED
50	1: replaced boolean return with true for com/graphhopper/gtfs/Transfers::lambda\$explodeTransfers\$4 - SURVIVED Coverage Tests	50	1: replaced boolean return with true for com/graphhopper/gtfs/Transfers::lambda\$explodeTransfers\$4 - SURVIVED Coverage Tests
51	2: negated conditional - KILLED	51	2: negated conditional - KILLED
51	1: replaced boolean return with true for com/graphhopper/gtfs/Transfers::lambda\$explodeTransfers\$5 - KILLED	51	1: replaced boolean return with true for com/graphhopper/gtfs/Transfers::lambda\$explodeTransfers\$5 - KILLED
52	2: replaced boolean return with false for com/graphhopper/gtfs/Transfers::lambda\$explodeTransfers\$5 - KILLED	52	2: replaced boolean return with false for com/graphhopper/gtfs/Transfers::lambda\$explodeTransfers\$5 - KILLED
55	1: replaced return value with null for com/graphhopper/gtfs/Transfers::lambda\$explodeTransfers\$6 - KILLED	55	1: replaced return value with null for com/graphhopper/gtfs/Transfers::lambda\$explodeTransfers\$6 - KILLED
58	1: replaced return value with Stream.empty for com/graphhopper/gtfs/Transfers::lambda\$explodeTransfers\$7 - SURVIVED Coverage Tests	58	1: replaced return value with Stream.empty for com/graphhopper/gtfs/Transfers::lambda\$explodeTransfers\$7 - SURVIVED Coverage Tests
63	1: negated conditional - KILLED	63	1: negated conditional - KILLED
64	1: replaced return value with Stream.empty for com/graphhopper/gtfs/Transfers::lambda\$explodeTransfers\$11 - KILLED	64	1: replaced return value with Stream.empty for com/graphhopper/gtfs/Transfers::lambda\$explodeTransfers\$11 - KILLED
65	1: replaced boolean return with true for com/graphhopper/gtfs/Transfers::lambda\$explodeTransfers\$8 - SURVIVED Coverage Tests	65	1: replaced boolean return with true for com/graphhopper/gtfs/Transfers::lambda\$explodeTransfers\$8 - SURVIVED Coverage Tests
65	2: negated conditional - KILLED	65	2: negated conditional - KILLED
66	1: replaced boolean return with false for com/graphhopper/gtfs/Transfers::lambda\$explodeTransfers\$9 - KILLED	66	1: replaced boolean return with false for com/graphhopper/gtfs/Transfers::lambda\$explodeTransfers\$9 - KILLED
67	2: replaced boolean return with true for com/graphhopper/gtfs/Transfers::lambda\$explodeTransfers\$9 - SURVIVED Coverage Tests	67	2: replaced boolean return with true for com/graphhopper/gtfs/Transfers::lambda\$explodeTransfers\$9 - SURVIVED Coverage Tests
70	1: replaced return value with null for com/graphhopper/gtfs/Transfers::lambda\$explodeTransfers\$10 - KILLED	70	1: replaced return value with null for com/graphhopper/gtfs/Transfers::lambda\$explodeTransfers\$10 - KILLED
73	1: replaced return value with Stream.empty for com/graphhopper/gtfs/Transfers::lambda\$explodeTransfers\$11 - SURVIVED Coverage Tests	73	1: replaced return value with Stream.empty for com/graphhopper/gtfs/Transfers::lambda\$explodeTransfers\$11 - SURVIVED Coverage Tests
81	1: replaced boolean return with true for com/graphhopper/gtfs/Transfers::lambda\$getTransfers\$toStop\$12 - SURVIVED Coverage Tests	81	1: replaced boolean return with true for com/graphhopper/gtfs/Transfers::lambda\$getTransfers\$toStop\$12 - SURVIVED Coverage Tests
83	2: negated conditional - KILLED	83	2: negated conditional - KILLED
83	3: negated conditional - SURVIVED Coverage Tests	83	3: negated conditional - SURVIVED Coverage Tests
84	1: negated conditional - SURVIVED Coverage Tests	84	1: negated conditional - SURVIVED Coverage Tests
84	2: negated conditional - KILLED	84	2: negated conditional - KILLED
85	1: replaced boolean return with true for com/graphhopper/gtfs/Transfers::lambda\$getTransfers\$toStop\$13 - SURVIVED Coverage Tests	85	1: replaced boolean return with true for com/graphhopper/gtfs/Transfers::lambda\$getTransfers\$toStop\$13 - SURVIVED Coverage Tests
87	1: removed call to java.util.Map::forEach - KILLED	87	1: removed call to java.util.Map::forEach - KILLED
87	1: negated conditional - KILLED	87	1: negated conditional - KILLED
88	1: negated conditional - KILLED	88	1: negated conditional - KILLED
91	1: removed call to java.util.Set::forEach - KILLED	91	1: removed call to java.util.Set::forEach - KILLED
98	1: replaced boolean return with true for com/graphhopper/gtfs/Transfers::lambda\$getTransfers\$toStop\$17 - KILLED	98	1: replaced boolean return with true for com/graphhopper/gtfs/Transfers::lambda\$getTransfers\$toStop\$17 - KILLED
113	2: replaced boolean return with false for com/graphhopper/gtfs/Transfers::lambda\$getTransfers\$toStop\$17 - KILLED	113	2: replaced boolean return with false for com/graphhopper/gtfs/Transfers::lambda\$getTransfers\$toStop\$17 - KILLED
113	3: negated conditional - KILLED	113	3: negated conditional - KILLED
119	1: replaced return value with Collections.emptyList for com/graphhopper/gtfs/Transfers::getTransfers\$toStop - KILLED	119	1: replaced return value with Collections.emptyList for com/graphhopper/gtfs/Transfers::getTransfers\$toStop - KILLED
125	1: negated conditional - NO_COVERAGE	125	1: negated conditional - KILLED
125	2: replaced boolean return with true for com/graphhopper/gtfs/Transfers::lambda\$getTransfers\$fromStop\$18 - NO_COVERAGE	125	2: replaced boolean return with true for com/graphhopper/gtfs/Transfers::lambda\$getTransfers\$fromStop\$18 - SURVIVED Coverage Tests
125	3: negated conditional - NO_COVERAGE	125	3: negated conditional - SURVIVED Coverage Tests
126	1: negated conditional - NO_COVERAGE	126	1: negated conditional - SURVIVED Coverage Tests
126	2: negated conditional - NO_COVERAGE	126	2: replaced boolean return with true for com/graphhopper/gtfs/Transfers::lambda\$getTransfers\$fromStop\$19 - SURVIVED Coverage Tests
126	3: replaced boolean return with true for com/graphhopper/gtfs/Transfers::lambda\$getTransfers\$fromStop\$19 - NO_COVERAGE	126	3: negated conditional - SURVIVED Coverage Tests
127	1: replaced return value with "" for com/graphhopper/gtfs/Transfers::lambda\$getTransfers\$fromStop\$20 - NO_COVERAGE	127	1: replaced return value with "" for com/graphhopper/gtfs/Transfers::lambda\$getTransfers\$fromStop\$20 - KILLED
129	1: removed call to java.util.Map::forEach - NO_COVERAGE	129	1: removed call to java.util.Map::forEach - KILLED
130	1: removed call to java.util.Set::forEach - NO_COVERAGE	130	1: removed call to java.util.Set::forEach - KILLED
144	1: replaced return value with Collections.emptyList for com/graphhopper/gtfs/Transfers::getTransfers\$fromStop - NO_COVERAGE	144	1: replaced return value with Collections.emptyList for com/graphhopper/gtfs/Transfers::getTransfers\$fromStop - KILLED
149	1: removed call to java.util.ArrayList::sort - SURVIVED Coverage Tests	149	1: removed call to java.util.ArrayList::sort - SURVIVED Coverage Tests
151	1: negated conditional - SURVIVED Coverage Tests	151	1: negated conditional - SURVIVED Coverage Tests
152	1: Changed increment from 1 to -1 - SURVIVED Coverage Tests	152	1: Changed increment from 1 to -1 - SURVIVED Coverage Tests
154	1: negated conditional - SURVIVED Coverage Tests	154	1: negated conditional - SURVIVED Coverage Tests
155	1: Changed increment from 1 to -1 - SURVIVED Coverage Tests	155	1: Changed increment from 1 to -1 - SURVIVED Coverage Tests
157	1: replaced int return with 0 for com/graphhopper/gtfs/Transfers::lambda\$findMostSpecificRule\$23 - SURVIVED Coverage Tests	157	1: replaced int return with 0 for com/graphhopper/gtfs/Transfers::lambda\$findMostSpecificRule\$23 - SURVIVED Coverage Tests
157	2: removed negation - SURVIVED Coverage Tests	157	2: removed negation - SURVIVED Coverage Tests
159	1: negated conditional - KILLED	159	1: negated conditional - KILLED
162	1: replaced return value with null for com/graphhopper/gtfs/Transfers::findMostSpecificRule - KILLED	162	1: replaced return value with null for com/graphhopper/gtfs/Transfers::findMostSpecificRule - KILLED
163	1: negated conditional - KILLED	163	1: negated conditional - KILLED
166	2: replaced boolean return with true for com/graphhopper/gtfs/Transfers::hashCodeSpecificDepartureTransferRules - SURVIVED Coverage Tests	166	2: replaced boolean return with true for com/graphhopper/gtfs/Transfers::hashCodeSpecificDepartureTransferRules - SURVIVED Coverage Tests
166	3: replaced boolean return with true for com/graphhopper/gtfs/Transfers::hashCodeSpecificDepartureTransferRules\$24 - SURVIVED Coverage Tests	166	3: replaced boolean return with true for com/graphhopper/gtfs/Transfers::hashCodeSpecificDepartureTransferRules\$24 - SURVIVED Coverage Tests
167	4: replaced boolean return with false for com/graphhopper/gtfs/Transfers::hashCodeSpecificDepartureTransferRules - KILLED	167	4: replaced boolean return with false for com/graphhopper/gtfs/Transfers::hashCodeSpecificDepartureTransferRules - KILLED
168	1: replaced boolean return with true for com/graphhopper/gtfs/Transfers::hashCodeSpecificArrivalTransferRules\$25 - KILLED	168	1: replaced boolean return with true for com/graphhopper/gtfs/Transfers::hashCodeSpecificArrivalTransferRules\$25 - KILLED
170	2: replaced boolean return with true for com/graphhopper/gtfs/Transfers::hashCodeSpecificArrivalTransferRules - KILLED	170	2: replaced boolean return with true for com/graphhopper/gtfs/Transfers::hashCodeSpecificArrivalTransferRules - KILLED
170	3: replaced boolean return with false for com/graphhopper/gtfs/Transfers::hashCodeSpecificArrivalTransferRules - KILLED	170	3: replaced boolean return with false for com/graphhopper/gtfs/Transfers::hashCodeSpecificArrivalTransferRules - KILLED
170	4: negated conditional - KILLED	170	4: negated conditional - KILLED

Certaines mutations sont devenues vertes simplement par le fait d'avoir ajouté de la couverture de code. Les quelques mutants qui ont survécus, lorsque les valeurs booléennes sont modifiées, n'ont pas été tués car mes tests ne vérifient pas la taille des listes résultantes mais simplement si elles sont vides ou non. Ainsi, même si le transfert de type 1 (BULLFROG) est ajouté à la liste (filtré précédemment), on aura le même retour, avec ou sans mutants.

- Java Faker :

Après avoir ajouté l'adresse du github de javafaker au pom.xml et après avoir recompilé, nous pouvons désormais créer de fausses données de tests, utile pour la classe Transfers. Effectivement, nous pourrions utiliser un faker pour créer des données gtfs valides.

```

@Test
public void testWithFaker() {
    Faker faker = new Faker();

    // arrange

    // initialization of random values
    String randomRouteId = faker.letterify(letterString:"RANDOM_????");
    String knownStopId = "BEATTY_AIRPORT";
    Transfers transfers = sampleFeed;

    // act

    // real stop, random route.
    List<Transfer> randomTransfers = transfers.getTransfersFromStop(knownStopId, randomRouteId);

    // asserts

    // as seen before, it should return a non-empty transfer list
    assertEquals(Collections.emptyList(), randomTransfers);

    // every transfer came from the same stop.
    assertTrue(randomTransfers.stream()
        .allMatch(t -> knownStopId.equals(t.from_stop_id)),
        "Every transfers must come from " + knownStopId);

    // Just to know what was created.
    System.out.println("Random route: " + randomRouteId);
    System.out.println("Generated transfers: " + randomTransfers.size());
}

```