# Étapes précédant les tests

**Diagramme Relation Objet**

Pour établir l’ORD, nous avons débuté avec le fichier qui nous intéressait, soit BipartiteXExtended. Nous avons recherché toutes les classes qui font partie du package algs4. À chaque fois que nous en trouvions, nous notions le type d’association. Après avoir complété la première classe, nous procédions à celles trouvées précédemment. Cela se poursuit jusqu’à tomber sur classe primitive. Voici le résultat de nos démarches :

A close up of a map

Description automatically generated

Figure A: ORD pour une partie du package algs4

Un constat important dans ce graphique : comme il y avait énormément de classes utilisées par la classe In, nous avons décidé de ne pas les mettre dans le diagramme. De plus, la classe Iterable qui y figure n’est pas à tester puisqu’il s’agit d’un type commun. Nous l’avons mis pour bien illustrer que trois des classes l’utilisent comme interface.

**CFW**

Une fois le diagramme ORD établi, nous sommes en mesure de bien cibler l’impact d’un changement de classe aura sur les autres. Il suffit de regarder quelles sont les classes qui pointent sur la classe modifiée de façon récursive. Voici le résultat de nos démarches :

|  |  |
| --- | --- |
| **Classe étudiée** | **CFW de la classe étudiée** |
| BipartiteXExtended |  |
| BipartiteX | BipartiteXExtended |
| Graph | BipartiteXExtended  BipartiteX |
| In | BipartiteXExtended  BipartiteX  Graph |
| Queue | BipartiteXExtended  BipartiteX |
| Stack | BipartiteXExtended  BipartiteX  Graph |
| Bag | BipartiteXExtended  BipartiteX  Graph |

Tableau 1: CFW des classes étudiées

Il est à noter que nous ne retrouvons pas la classe Iterable dans le tableau 1. Nous l’avons exclu puisqu’il s’agit d’une classe commune dont les preuves sont faites. Il en est de même pour toutes les relations que la classe In possède avec les autres classes telles que Scanner, File, BufferedInputStream et etc.

**Ordre topologique et ordre de tests**

En regardant attentivement le tableau 1, nous constatons que les classes In, Queue, Stack et Bag n’apparaissent pas dans le CFW d’aucune classe. Ensuite, nous remonterons dans le tableau pour retrouver l’ordre. Nous avons conscience qu’il s’agit d’une coïncidence que les classes dans le tableau sont l’ordre inverse de celui des tests. Voici la représentation graphique de l’ordre topologique que l’on peut déduire de la figure A :

A close up of text on a white surface

Description automatically generated

Figure B: Ordre topologique

On constate que la figure B correspond au résultat attendu de notre lecture du tableau 1. Nous pouvons facilement en retirer l’ordre de tests en s’assurant d’effectuer le test d’une classe une fois toutes les autres classes pointant sur elle ont été testées. Voici l’ordre des tests :

|  |  |
| --- | --- |
| Niveau | Classes à tester |
| 1 | Queue, Stack, Bag, In |
| 2 | Graph |
| 3 | BipartiteX |
| 4 | BipartiteXExtended |

Tableau 2: Ordre des tests:

Établir le lien entre le tableau 2 et la figure B est trivial : nous avions déjà illustrer les niveaux de test selon le décalage vertical. Nous pouvons maintenant procéder aux tests.

# Tests