Le programme *egon* a quelques problèmes comme méthode pour repérer les classes divines. À un certain seuil, *egon* donnera comme sortie beaucoup de classes, divines ou non.

Dans un cas où il n’y a pas de classe divine ou que le seuil est trop grand, certaines autres classes vont souvent ressortir. De la même façon, si le seuil est trop bas ou s’il y a peu de classes dans le dossier source, aucune classe ne sortira.

On remarque que l’une des types de classes qui sort souvent est le « main ». Dans notre cas JFreeChart.java est couplé plus souvent que les autres classes, soit 634 fois. On s’attend à ce qu’il soit très couplé, mais 634 semble être un immense nombre, il serait probablement une bonne idée de tout de même les analyser.

Dans un autre cas, des méthodes main ou semblables à un main auraient aussi pu sortir comme la plus couplé, sans être une classe divine. Par exemple, à un certain seuil, notre classe egon sera considéré comme divine selon sa méthode, malgré le fait qu’elle soit relativement simple. Certains types de classes semblable au main devront mentionner beaucoup d’autres classes, par exemple les contrôleurs. L’architecture d’une application Java est telle que le programme *egon* n’est pas toujours un outil précis pour trouver des classes divines. Lors de l’analyse, il faut porter une attention particulière pour être sûr qu’on ne déclare pas une classe comme divine si elle ne l’est pas. Une façon serait probablement de calculer le nombre de lignes moyen/médiane par classe et le nombre de couplage moyen/médiane par classe pour avoir un niveau de comparaison.

Le deuxième type de classe qui sortira souvent sont les classes utiles. Une classe utile est créer pour faciliter l’écriture du code et englobé des problèmes qui reviennent souvent. Par exemple, des classes qui travaillent des types de données spéciaux vont souvent revenir. Dans notre cas, nous avons TextUtils, SerialUtils et DatasetUtils qui sont sorties.

Des classes utiles peuvent parfois avoir une grande quantité de lignes, car elles doivent être bien documentés, pour faciliter l’utilisation et certaines d’entre elles nécessitent plus méthodes pour englober les spécifications. Il faudrait donc peut-être changer le calcul du nombre de ligne pour calcul de nombre de lignes de code non commentés.

Ceci nous apporte à un autre point important d’*egon*: les commentaires. Si un projet est réalisé par plusieurs développeurs qui ne commentent pas de la même façon (et que certain ne commentent pas), la courbe de répartition du nombre de lignes sera influencée.

En conclusion, *egon* a besoin de quelques améliorations pour être un bon indicateur des classes divines et ne sera probablement jamais parfais. Mais cette méthode peut toujours être utile pour donner une indication de quelles classes sont à risque d’être des classes divines si le développeur l’utilise avec un œil critique.