Travail de preuve de concept pour un programme de dessin vectoriel

Le design principal de l’application repose sur la présence de forme distincte dans un panneau de dessin. Nous voulions créer le moins de couplage possible entre notre panneau de dessin et les éléments qui vont composer la fenêtre de dessin. Pour ce faire nous avons décider de gérer une seule sorte d’objet que nous avons appelé des « Element ». Ces éléments peuvent représenter tous les cas de dessin que nous déterminé et doit pouvoir permettre l’ajout de nouvelles formes sans avoir à changer le code des interfaces graphiques. Pour ce faire nous avons pensé à un patron de conception soit un constructeur. Il nous permet la flexibilité de créer des éléments d’ajouter des formes diverses sans trop de modification. Pour nous permettre de passer les paramètres des différents panneaux nous avons pensé à introduire un Singleton afin créer une instance unique à l’application afin d’appeler le constructeur. Ainsi, nous avons un découplage complet et le panneau de dessin ne gère que des objets du type « Element ». Le danger ici du Singleton est de s’assurer que l’instance ne peut-être accéder simultanément par des objets de l’application. Si nous avions eu plus de temps, nous aurions aimé se débarrasser du Singleton afin pouvoir gérer la construction autrement.

Pour la simplicité de l’élaboration de la preuve de concept nous avons fait le choix de réduire les options de couleurs et de grosseurs de trait en incluant seulement quelques options à l’application. Pour se faire, nous avons créé trois énumérations qui contiennent les valeurs que nous avons déterminé. La possibilité de les éliminé est assez facile et va demander que quelques modifications puisque les données membre des « Element » sont de type java.awt.Color ou des primitif float. Par contre si nous maintenons le Singleton tel quel, il faudra revoir le « GenerateurElement » puisque celui-ci utilise explicitement les données des énumérations.