Approche globale

Après plusieurs discussions au sein de l'équipe, il a été entendu de ne pas utiliser trop de design pattern. Puisque nous n'étions pas à l’aise avec l’implantation des design pattern, nous avions l'impression que ce serait trop long. En effet, notre but était d'opter pour la simplicité et d'éviter de perdre du temps compte tenu des délais. Dans notre optique, nous respections le principe Keep it simple, stupid.

Choix de conception

Nous avons utilisé l’héritage ainsi que le design pattern **factory** pour découper l’architecture des formes. Concernant le choix de pattron, nous avons utilisé ce pattern pour éviter de dupliquer la méthode creationForme.

Améliorations possibles dans la conception

Canvas : ne respecte pas le "single responsability principle". Nous aurions pu la diviser en plusieurs classe. Dans un monde idéal, nous aurions pu utiliser le pattron d’architecture MVC. Du coup, la classe caneva aurait pu être divisée dans une classe modèle qui contiendrait notre liste de forme et notre vue contiendrait notre fenêtre ainsi que nos events listners.

Dans une optique d’un projet qui évoluera, l’idéal aurait été d’utiliser une **stragégie** pour les différentes options d’exportations. Par exemple, nous aurions pu avoir une stratégie par type de fichier ( .jpeg, .gif etc.).

Pour l’exportation, en utilisant le pattern **décorateur** pour gérer les actions des formes, nous aurions pu enlever la responsabilité de cette action à la classe canevas et ainsi respecter single responsability principle.

Constats

Un des constats est que nous avons commencé à coder trop rapidement sans trop avoir de plan établi. C’est ce qui explique les super classes, le peu de design patterns utilisés et la difficulté à intégrer le travail de chacun. À la fin du projet, nous nous sommes aperçu qu’il y aurait des amélioration possible. Par exemple, des noms des classes et variables aurait pu être plus parlant.

Il est difficile d’utiliser git à ce stade de la formation. En effet, il serait peut-être mieux d’expérimenter git en équipe de deux et sur de plus petits projets. Également, il serait bien qu’il y ait une seule personne qui est responsable du master (branche develop). Une autre observation que nous avons fait est qu’il peut devenir compliqué de merger si on as pas committé très régulièrement.