**Rapport A31/32 : Choix de conception**

L’architecture de logiciel que nous avons choisis correspond au MVC. Le logiciel est organisée en trois parties : le Modèle (model), contenant les données, la Vue (view) qui les affiche et le Contrôleur (controller) qui les manipule.

Nous avons mis en place dans notre Controller une classe Facade : elle sélectionne la vue en fonction de ses demandes et en manipulant les données du modèle. Facade permet d’encapsuler la complexité entre le modèle et la vue, n’en rajoutant pas inutilement dans ces deux parties.

Pour limiter la dépendance entre la classe Echec du modèle et la classe echecWindow de la vue, nous avons décidé d’utiliser le design pattern de l’Observer. Nous avons créé une interface echecObserver qui surveille (« observe ») la classe Echec et permet à echecWindow de réagir en fonction des changements des changements survenus sur Echec. Les différentes modifications demandant du controller demandant des changements sur la view sont le choix d’un mouvement par un joueur, une modification dans les mouvements qu’il peut faire ou la mise en échec/échec et mat d’un joueur. Chacune de ces situations correspond à une méthode dans echecObserver.

Les différentes pièces ont plusieurs fonctionnalités et propriétés communes : elles ont chacune la possibilité de capturer une autre pièce, une liste de déplacements possibles, une couleur… Il a été décidé de faire hériter chaque type de pièce d’une super classe « Piece » contenant toutes ces propriétés communes. Cela revient à suivre le principe DRY : si nous n’avons pas mutualisé le code, nous nous serons répété.

Les différents types de pièces sont ensuite créées par pieceFactory, faisant appelle à la Factory Method : nous pouvons donc instancier chaque Pièce depuis la classe Echec, sans qu’Echec n’ait besoin de connaitre leurs classes concrètes. Pour cela, la classe Echec utilise la méthode setPiece(), faisant appel à notre pieceFactory qui créera toutes les pièces de l’échiquier.

Nous avons décidé de signaler la couleur des Pièces avec un énumérateur Color, contenant deux possibilités : BLACK et WHITE. Cela nous permet d’éviter les confusions qu’aurait pu entrainer l’utilisation d’un booléen ou d’un entier pour identifier la couleur d’une pièce.