Objectif du projet : Création d'une application en Java offrant la possibilité de jouer à deux jeux, le jeu de Nim ou le jeu Puissance 4.

Le projet est à réaliser par trinôme. Il suivra un développement itératif et incrémental. Les itérations seront de deux semaines, les butées temporelles correspondront aux séances de TP au cours desquelles vous devrez faire une démonstration du fonctionnement de votre projet. Il est important d'avoir un projet opérationnel à présenter à chaque séance même s'il ne couvre pas toutes les fonctionnalités.

Le découpage en itérations sera le suivant :

Itération 1 : Développer une application permettant de jouer au jeu de Nim à deux joueurs.

Itération 2 : Développer une application permettant de jouer au jeu de puissance 4 à deux joueurs.

Itération 3 : Développer une application proposant deux jeux au choix, le jeu de Nim ou le jeu Puissance 4 à deux joueurs avec intégration de contraintes sur les deux jeux.

Itération 4 : Intégrer à l'application la possibilité de jouer à deux joueurs ou de jouer seul contre l'ordinateur.

Itération 5 : Intégrer plusieurs stratégies pour l'ordinateur.

Ce sujet vous présente le détail de la troisième itération.

Tout au long du projet, vous devrez toujours respecter une architecture MVC et maintenir à jour la Javadoc de vos méthodes.

Itération 3 :

Vous devez développer une application Java permettant à deux joueurs humains de jouer au jeu de Nim ou au jeu de Puissance4 en mode console.

L'objectif de cette itération est de fusionner les deux applications que vous avez créées pour en créer une qui permette de choisir au lancement de l'application si on souhaite jouer au jeu de Nim ou au jeu Puissance4. Si les joueurs choisissent Nim, il pourront faire autant de parties qu'ils souhaitent du jeu de Nim, s'ils choisissent Puissance4, il pourront faire autant de parties qu'ils souhaitent du jeu Puissance4.

Vous avez pu constater qu'il y a des similitudes dans le déroulement du scénario pour un jeu de Nim ou un jeu Puissance4 puisqu'il s'agit dans les deux cas d'un jeu de société dans lequel les parties se font à deux joueurs humains qui jouent à tour rôle jusqu'à la fin de la partie. A la fin de la partie, on

affiche le nom du vainqueur puis on propose de rejouer. Le jeu se termine avec l'affichage des scores.

Travail à faire :

Première partie

1. Créer un diagramme de classe de conception décrivant l'application. Nous avons étudié en cours de MO deux design patterns comportementaux, **stratégie** et **patron de méthode**. Utilisez le design pattern qui vous semble le plus adapté pour répondre au problème de mutualisation de certaines parties de code et de gestion des parties qui varient. L'idée qui doit vous guider est qu'on pourrait vouloir ajouter d'autres jeux de société à deux joueurs par la suite à notre application. L'ajout devrait pouvoir se faire facilement en respectant le principe ouvert/fermé.

2. Coder et tester votre application. C'est dans le `main`, que l'`ihm` demande au joueur s'il veut jouer au jeu de Nim ou au jeu Puissance4 et ensuite, le jeu demandé commence.

Deuxième partie

On souhaite maintenant apporter des évolutions aux deux jeux :

Evolution pour le jeu Puissance4 : les rotations

Désormais, **au début de chaque partie**, le programme demande si on souhaite intégrer la rotation dans cette partie. Ainsi à chaque tour de jeu, un joueur peut soit faire glisser un jeton dans une colonne, soit effectuer une rotation à 90° de la grille du jeu.

Deux sens de rotation sont possibles : la première à droite et la seconde à gauche. La figure ci-dessous illustre l'enchaînement suivant : rotation à droite de la situation initiale, suivie d'une rotation à gauche.

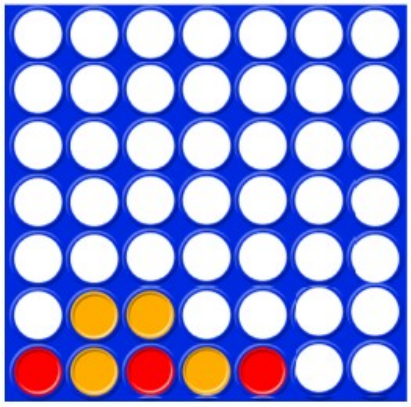	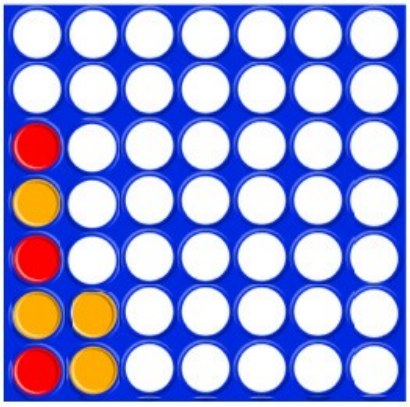	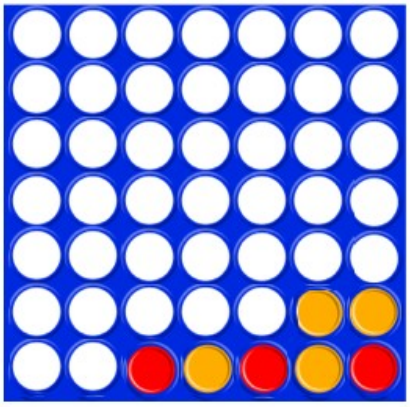
Figure 1 : Situation initiale	Figure 2 rotation à droite de la figure 1	Figure 3 : rotation à droite de la figure 2

Le nombre de rotations par partie est limité à quatre par joueur. Notez que le choix du type de partie (avec ou sans rotation) se fait en début de partie. On peut donc faire une partie avec rotation puis enchaîner sur une autre sans rotation et ainsi de suite jusqu'à la fin complète du jeu.

Evolution pour le jeu de Nim : contraintes sur le nombre d'allumettes à retirer

Désormais, **au début de chaque partie**, le programme demande si on souhaite jouer avec une contrainte sur le nombre maximum d'allumettes que l'on peut retirer à chaque coup.
L'utilisateur saisit un entier positif correspondant à ce nombre maximum ou 0 si les joueurs ne souhaitent pas jouer avec une contrainte sur le nombre maximum d'allumettes. Notez que le choix du type de partie (avec ou sans contrainte sur le nombre maximum d'allumettes à retirer) se fait en début de partie. On peut donc faire une partie avec contrainte puis enchaîner sur une autre sans contrainte et ainsi de suite jusqu'à la fin complète du jeu.

Intégrer ces changements à votre précédent code.

Vous déposerez sur CELENE dans le dépôt correspondant à l'itération 3 un dossier contenant le diagramme de classe de conception et le code de l'application finale.