

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

Afin de réaliser ce projet, nous vous autorisons à utiliser une IA d'assistance à la génération de code telle que ChatGPT. Vous êtes libres du choix de cette IA.

L'autorisation d'utiliser ce type d'IA entraîne un certain nombre de conséquences.

Nos attentes vis-à-vis de la *qualité, quantité et pertinence du code* seront plus élevées que si vous deviez réaliser le projet sans l'assistance d'une IA.

Vous devez utiliser uniquement du code que vous comprenez parfaitement et que vous êtes capables d'expliquer durant les séances/soutenances. L'évaluation en tiendra compte de manière significative.

En effet, il faut noter que les assistants de génération de code produisent du code utilisant *toutes les fonctionnalités du langage*, y compris des fonctionnalités qui n'ont pas été abordées en L2. Vous pouvez bien sûr utiliser des fonctionnalités qui n'ont pas été vues en cours, à condition d'avoir pris le temps de vous former sur ces fonctionnalités (ce que c'est, à quoi ça sert, comment ça fonctionne, pourquoi c'est utilisé dans ce contexte précis) et d'être capable de répondre aux questions de l'enseignant qui vous interroge.

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 des deux premières itérations.

Les autres évolutions vous seront présentées au fur et à mesure de l'avancement du projet.

Tout au long du projet, vous devrez toujours respecter une architecture MVC et utiliser le cours de modélisation orientée objet pour mener votre analyse et votre conception avant de coder.

Itération 1 :

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

Il existe de nombreuses variantes des jeux de Nim dans lesquels deux joueurs jouent à tour de rôle avec des allumettes, des pièces, des graines ou toute autre type d'objets. On jouera, pour ce projet, avec des allumettes. En disposant ces allumettes les unes à côtés des autres, on forme un tas d'allumettes. On peut jouer avec un ou plusieurs tas. Les joueurs vont, chacun leur tour, prendre un certain nombre d'allumettes ; selon les variantes, le dernier à jouer gagne ou perd. Il y a forcément un gagnant, et donc un perdant.

Pour le projet, on fixe la disposition initiale des allumettes et les règles du jeu suivantes :

- Une fois défini le nombre de tas qui doit être supérieur ou égal à 1, on dispose un nombre impair d'allumettes par tas de la manière suivante : dans le $i^{\text{ième}}$ tas, on place $2 * i - 1$ allumettes.
- A chaque tour, un des joueurs sélectionne une ligne non vide et retire au moins une allumette.
- Le joueur gagnant est celui qui prend la dernière allumette.

Par exemple, si on choisit 3 tas, l'état initial de la partie sera :

Tas 1 -> 1 allumette

Tas 2 -> 3 allumettes

Tas 3 -> 5 allumettes

Affichage des tas :

```
|  
|||  
|||||
```

Avant de commencer à jouer, le programme demande le nombre de tas avec lequel on souhaite jouer. L'utilisateur saisit un nombre de tas qui doit être ≥ 1 .

Le programme demande le nom du joueur 1. Le joueur 1 saisit son nom. Le programme demande le nom du joueur 2. Le joueur 2 saisit son nom.

La partie peut alors commencer. Le joueur 1 commence.

Avant chaque coup, le programme affiche l'état de la partie et invite le joueur à jouer son coup en rappelant son nom et la forme d'un coup. Par exemple, on pourra afficher : " Dupont : à vous de

jouer un coup sous la forme 'm n' où m est le tas choisi et n le nombre d'allumettes à retirer dans ce tas".

Si le coup est invalide, on en informe le joueur et on lui propose de rejouer. Si le coup est valide, le programme doit tester si la partie est finie. Si elle n'est pas finie, c'est à l'autre joueur de jouer sinon le programme détermine le vainqueur, affiche son nom, puis propose de rejouer une partie.

Si les joueurs décident de rejouer une partie, le programme redémarre une partie avec le même nombre de tas. S'ils décident d'arrêter, le programme affiche le nombre de parties gagnées par chacun des joueurs et le nom du vainqueur ou "ex aequo" si les deux joueurs ont remportés le même nombre de parties.

Il est important de respecter avec précision le déroulement du jeu tel que décrit ci-dessus.

Choix des classes:

Les classes seront réparties dans trois packages qui seront nommés : `modele`, `vue`, `controleur`.

Classes du package `modele`:

Les classes du modèle découleront de votre analyse et de votre conception dont vous fournirez la documentation comme indiqué ci-dessous. Il doit y avoir une cohérence exacte entre les classes de votre programme, leurs attributs et leurs méthodes et les classes définies dans le modèle de conception.

Classe du package `vue`:

`Ihm` : interagit avec l'utilisateur en affichant des messages et en récupérant les informations qu'il saisit, informations qui sont récupérées par la classe du package `controleur`.

Classe du package `controleur` :

`ControleurJeuNim` : gère le jeu, c'est à dire l'enchaînement des parties jusqu'à ce que les joueurs décident d'arrêter. A sa création, le `controleurJeuNim` reçoit un objet de type `Ihm`. Il communique avec les objets du modèle pour la gestion de chaque partie.

A vous de prévoir les méthodes que doit fournir le `controleur`.

On ajoute un package `main` contenant une classe `Main` qui contient le `main`. Voici le code de la méthode `main` :

```
public static void main(String[] args) {
    Ihm ihm = new Ihm();
    ControleurJeuNim controleurJeuNim=new ControleurJeuNim(ihm);
    controleurJeuNim.jouer();
}
```

On notera que seules les méthodes de l'objet de type `Ihm` permettront d'interagir avec l'utilisateur. ***En dehors de cette classe, il n'y aura aucune utilisation de `System.out.println` ou d'un objet de type `Scanner` `sc = new Scanner(System.in)`.***

Travail à rendre à la fin de l'itération1 : vous déposerez sur CELENE une archive au format zip contenant l'analyse de l'application, un modèle de conception, le code de votre projet dont vous ferez la démonstration pendant la séance de TP ainsi qu'un document détaillant la répartition des tâches au sein du trinôme et expliquant de quelle manière vous avez utilisé l'IA dans votre projet.

Pour l'analyse, vous fournirez un diagramme de cas d'utilisation avec la rédaction détaillée du ou des scénarios avec les extensions associées.

Vous devez prévoir les situations où l'utilisateur saisit des données invalides comme par exemple, un caractère à la place d'un entier, un entier négatif à la place d'un entier positif, un coup invalide par rapport à l'état de la partie. Le jeu doit permettre de saisir de nouvelles valeurs et de poursuivre.

C'est cette analyse qui vous permettra d'écrire un code fiable et robuste et de prévoir vos cas de test pour tester votre application.

Pour le modèle de conception, vous fournirez le diagramme de classe de vos classes logicielles avec les attributs, les opérations et les associations entre les classes en précisant le sens de navigation.

Dans le document expliquant votre utilisation de l'IA, vous pourrez répondre aux questions suivantes :

- En quoi vous a-t-elle été utile dans chaque partie, analyse, conception, codage ?
- Est-ce qu'elle vous a fait gagner du temps ? Si oui sur quelles parties.
- Avez-vous rencontré des difficultés d'utilisation ? Si oui, lesquelles ?

Itération 2 :

Vous devez développer une application Java permettant à deux joueurs humains de jouer au jeu Puissance 4 en mode console toujours en respectant une architecture MVC.

Chaque joueur dispose de jetons : rouges pour le premier joueur et jaunes pour le second.

Une grille rigide comptant 7 lignes et 7 colonnes est à disposition des deux adversaires dont le but est d'être le premier à aligner 4 jetons de même couleur horizontalement, verticalement ou diagonalement.

À chaque tour, chaque joueur choisit une colonne de la grille et lâche son jeton au sommet de celle-ci. Ce dernier est alors attiré par la gravité et tombe jusqu'en bas si la colonne est vide ou se positionne juste au-dessus du dernier jeton de la colonne, si celle-ci n'est pas vide.

La partie se termine dès qu'un joueur aligne 4 jetons. Il est alors déclaré gagnant. Si la grille est pleine sans qu'aucun des joueurs ne soit arrivé à aligner 4 jetons, alors la partie est déclarée nulle.

Déroulement du jeu:

Le programme demande le nom du joueur 1. Le joueur 1 saisit son nom. Le programme demande le nom du joueur 2. Le joueur 2 saisit son nom.

La partie peut alors commencer. Le joueur 1 commence.

Avant chaque coup, le programme affiche l'état de la partie et invite le joueur à jouer son coup en rappelant son nom. Par exemple, on pourra afficher : " Dupont : à vous de jouer .

Le joueur saisit un numéro correspondant à une colonne de la grille (de 1 à 7).

Si le numéro de colonne est invalide, on affiche un message et on propose au joueur de saisir de nouveau son coup.

Si le coup est valide, le programme doit tester si la partie est finie. Si elle n'est pas finie, c'est à l'autre joueur de jouer sinon le programme détermine le vainqueur, affiche son nom, puis propose de rejouer une partie.

Si les joueurs décident de rejouer une partie, le programme redémarre une partie. S'ils décident d'arrêter, le programme affiche le nombre de parties gagnées par chacun des joueurs et le nom du vainqueur ou "ex aequo" si les deux joueurs ont remportés le même nombre de parties.

Travail à rendre à la fin de l'itération2 : vous déposerez sur CELENE une archive au format zip contenant l'analyse de l'application, un modèle de conception et le code de votre projet dont vous ferez la démonstration pendant la séance de TP ainsi qu'un document détaillant la répartition des tâches au sein du trinôme et expliquant de quelle manière vous avez utilisé l'IA dans votre projet..