

Compte rendu de TP

Sujet : Shifumie

Nom : Djefal

Prénom : Nawell

Enseignant : Mme. Beauvallet

Sommaire

1. Introduction.....	2
2. Règles du jeu.....	2
Règles du jeu.....	2
3. Objectif du projet.....	3
4. Technologies et outils utilisés.....	3
5. Étapes de développement.....	4
➤ Étape 1 : Définir le nombre de points.....	4
➤ Étape 2 : Choix du joueur.....	4
➤ Étape 3 : Choix aléatoire de l'ordinateur.....	5
➤ Étape 4 : Révélation du suspense.....	5
➤ Étape 5 : Détermination du gagnant de la manche.....	5
➤ Étape 6 : Boucle des manches.....	5
➤ Étape 7 : Fin de partie.....	6
➤ Étape 8 : Mise en forme du programme.....	6
➤ Étape 9 : Ajout du Puits.....	6
➤ Étape 10 : Choix du mode.....	7
6. Conclusion.....	7

1. Introduction

Ce travail dirigé (TD) a pour objectif de mettre en pratique les bases de la programmation à travers un projet ludique : la création du jeu Shifumi (Pierre-Feuille-Ciseau+ plus puis pars la suite) en Java.

Ce jeu permet de consolider les notions fondamentales de l'algorithmique telles que les boucles, les conditions, la génération aléatoire et la gestion des entrées utilisateurs.

L'activité est progressive et structurée afin d'amener l'étudiant à construire un programme complet, fonctionnel et interactif.

En plus de l'aspect technique, ce projet favorise la logique, la rigueur et le plaisir du code, tout en développant l'autonomie dans la conception d'un mini-jeu.




2. Règles du jeu

Le **Shifumi**, également connu sous le nom de **Pierre-Feuille-Ciseau**, est un jeu traditionnel opposant deux adversaires.

Chaque joueur choisit simultanément l'un des trois symboles suivants :

- **Pierre** (représente la force brute)
- **Feuille** (représente la souplesse et la ruse)
- **Ciseau** (représente la précision et la rapidité)

Le principe est simple :

- La pierre écrase le ciseau 
- Le ciseau coupe la feuille 
- La feuille enveloppe la pierre 
- Si les deux joueurs font le même choix, il y a égalité.

Dans la version numérique du jeu, l'utilisateur affronte l'ordinateur, qui effectue un choix aléatoire à chaque manche.

Le premier à atteindre le nombre de points définis en début de partie (3, 5 ou 10) est déclaré vainqueur.

Une variante avancée peut inclure une quatrième option, le Puits, qui renforce la stratégie du jeu :

Le Puits gagne contre la Pierre et le Ciseau, mais perd contre la Feuille.

3. Objectif du projet

L'objectif de ce TD est de concevoir et programmer un jeu interactif de Shifumi en Java. Ce projet vise à faire pratiquer les bases essentielles de la programmation tout en développant la logique et la rigueur algorithmique.

À travers ce travail, on apprend à :

- Manipuler les structures de contrôle (conditions `if`, boucles, `while` ou `for`) ;
- Gérer les entrées utilisateur et les valeurs aléatoires ;
- Mettre en place un système de score et de boucle de jeu ;
- Améliorer la présentation du programme grâce à des affichages lisibles et dynamiques ;
- Comprendre l'importance de tester et valider chaque étape du code avant d'avancer.

Ce projet permet également d'acquérir une méthode de travail proche de celle utilisée dans le développement professionnel :

analyser le besoin, concevoir une solution, coder, tester et corriger.

4. Technologies et outils utilisés

Pour la réalisation de ce projet, les outils suivants sont utilisés :

- **Langage de programmation : Java**
 - Version recommandée : **JDK 17 ou supérieure**
 - Java est un langage orienté objet, idéal pour l'apprentissage des bases de la programmation grâce à sa syntaxe claire et à sa large communauté de développeurs.
- **Environnement de développement : Eclipse IDE**
 - Eclipse est un environnement complet et convivial permettant d'écrire, exécuter et déboguer facilement le code Java.
 - Il offre une interface intuitive, la coloration syntaxique, et une console intégrée pour visualiser les résultats du programme.
- **Matériel requis :**
 - Un ordinateur équipé du **JDK** (Java Development Kit).
 - L'accès à un terminal ou à l'interface d'Eclipse pour exécuter le programme.

Ces outils offrent une base solide pour comprendre le fonctionnement d'un projet Java, du code source jusqu'à l'exécution du programme.

5. Étapes de développement

Étape 1 : Définir le nombre de points

```
// Étape 1 : définir le nombre de points
do {
    System.out.println("\nEn combien de points se déroule la partie ? (3, 5 ou 10)");
    while (!sc.hasNextInt()) {
        System.out.println("⚠ Entrez un nombre valide (3, 5 ou 10) :");
        sc.next();
    }
    nbPoints = sc.nextInt();
} while (nbPoints != 3 && nbPoints != 5 && nbPoints != 10);
```

Cette étape permet de demander à l'utilisateur le nombre de points nécessaires pour remporter la partie. Une vérification est effectuée pour s'assurer que la valeur saisie est valide (3, 5 ou 10).

Étape 2 : Choix du joueur

```
// Étape 2 : choix du joueur
do {
    System.out.println("\nChoisissez : Pierre (P), Feuille (F), Ciseau (C)"
        + (modeAvecPuits ? ", ou Puits (U)" : "") + " :");
    choixJoueur = sc.next().toUpperCase().charAt(0);
} while (!estChoixValide(choixJoueur, modeAvecPuits));
```

L'utilisateur saisit son choix parmi Pierre, Feuille ou Ciseau (et éventuellement Puits). Une boucle contrôle la validité de la saisie afin d'éviter toute entrée incorrecte.

Étape 3 : Choix aléatoire de l'ordinateur

```
// Étape 3 : choix aléatoire de l'ordinateur
int aleatoire = (int) (Math.random() * (modeAvecPuits ? 4 : 3)) + 1;
if (aleatoire == 1)
    choixOrdi = 'P';
else if (aleatoire == 2)
    choixOrdi = 'F';
else if (aleatoire == 3)
    choixOrdi = 'C';
else
    choixOrdi = 'U'; // uniquement si mode avec puits
```

Le programme génère un choix aléatoire pour l'ordinateur à l'aide de la fonction `Math.random()`, simulant ainsi un adversaire virtuel.

Étape 4 : Révélation du suspense

```
// Étape 4 : suspense
System.out.println("\nL'ordinateur réfléchit...");
try {
    Thread.sleep(3000); // pause 3s
} catch (InterruptedException e) {
    e.printStackTrace();
}
```

Une pause de trois secondes est introduite avec `Thread.sleep()` avant d'afficher le choix de l'ordinateur, pour renforcer l'aspect ludique du jeu.

Étape 5 : Détermination du gagnant de la manche

```
// Étape 5 : déterminer le gagnant
if (choixJoueur == choixOrdi) {
    System.out.println("⚖ Égalité !");
} else if (joueurGagne(choixJoueur, choixOrdi, modeAvecPuits)) {
    System.out.println("🏆 Vous gagnez cette manche !");
    scoreJoueur++;
} else {
    System.out.println("🏆 L'ordinateur gagne cette manche !");
    scoreOrdi++;
}

System.out.println("📊 Score actuel → Joueur : " + scoreJoueur + " | Ordinateur : " + scoreOrdi);
}
```

Les choix du joueur et de l'ordinateur sont comparés selon les règles du Shifumi afin d'attribuer un point au vainqueur ou d'indiquer une égalité.

Étape 6 : Boucle des manches

```
// Étape 6 : boucle des manches
while (scoreJoueur < nbPoints && scoreOrdi < nbPoints) {
    char choixJoueur;
    char choixOrdi;
}
```

Le programme répète les manches tant qu'aucun joueur n'a atteint le nombre de points défini. La boucle **while** contrôle la continuité du jeu.

Étape 7 : Fin de partie

```
// Étape 7 : fin de partie
if (scoreJoueur > scoreOrdi)
    System.out.println("\n🏆 Félicitations, vous remportez la partie !");
else
    System.out.println("\n🏆 L'ordinateur remporte la partie... Revanche ?");

System.out.println("Voulez-vous rejouer ? (O/N)");
char reponse = sc.next().toUpperCase().charAt(0);
if (reponse != 'O') {
    rejouer = false;
    System.out.println("\nMerci d'avoir joué ! À bientôt 🍀");
}
}

sc.close();
}

private static void afficher(String string) {
    // TODO Auto-generated method stub
}
}
```

Lorsque l'un des participants atteint le score cible, le programme affiche le gagnant et propose de recommencer une nouvelle partie

Étape 8 : Mise en forme du programme

```
// Étape 8 : affichage mis en forme
System.out.println("Vous : " + symbole(choixJoueur) + " (" + nom(choixJoueur) + ")");
System.out.println("Ordinateur : " + symbole(choixOrdi) + " (" + nom(choixOrdi) + ")");
```

Les résultats et les symboles sont présentés de manière visuelle et attrayante (

- ✎ pour Pierre,
- _ pour Feuille,
- > pour Ciseau, etc.),

améliorant l'expérience utilisateur.

Étape 9 : Ajout du Puits

```
// Étape 9 : intégration de la nouvelle option "Puits"
// Le Puits bat Pierre et Ciseau, mais perd contre Feuille
public static boolean joueurGagne(char joueur, char ordi, boolean modeAvecPuits) {
    return (joueur == 'P' && ordi == 'C') ||
           (joueur == 'F' && ordi == 'P') ||
           (joueur == 'C' && ordi == 'F') ||

           // Étape 9 : règles spécifiques au Puits
           (modeAvecPuits && joueur == 'U' && (ordi == 'P' || ordi == 'C')) ||
           (modeAvecPuits && joueur == 'F' && ordi == 'U');
}
```

Une nouvelle option « Puits » est intégrée, modifiant les règles du jeu : le Puits bat Pierre et Ciseau, mais perd contre Feuille. Cette extension enrichit la logique de comparaison.

Étape 10 : Choix du mode

```
// Étape 10 : choix du mode
System.out.println("Souhaitez-vous jouer en mode avec Puits ? (O/N)");
boolean modeAvecPuits = sc.next().toUpperCase().charAt(0) == 'O';

int scoreJoueur = 0;
int scoreOrdi = 0;
int nbPoints;
```

En début de partie, l'utilisateur choisit de jouer en mode classique ou avec la variante « Puits », permettant de personnaliser l'expérience de jeu.

6. Conclusion

La réalisation du jeu Shifumi en Java m'a permis de mettre en pratique les bases essentielles de la programmation : les boucles, les conditions, les fonctions aléatoires et la gestion des entrées utilisateur.

Ce projet m'a aidé à mieux comprendre la logique algorithmique et le déroulement d'un programme interactif.

J'ai rencontré quelques difficultés lors de l'ajout du mode "Puits", notamment pour adapter les règles du jeu et intégrer correctement cette nouvelle option dans le code. Cependant, avec de la recherche, de la réflexion et plusieurs essais, j'ai finalement réussi à le faire fonctionner.

Cette expérience m'a appris l'importance de la persévérance et du débogage, deux compétences essentielles en programmation.

Au final, ce TD m'a non seulement permis d'approfondir mes connaissances en Java, mais aussi de renforcer ma confiance en ma capacité à résoudre des problèmes par moi-même.