

# IN-B1-UE09 : Programmation orientée objet

## Laboratoire 1

**Durée prévue :** 4 heures

### Acquis d'apprentissage visés

À la fin du laboratoire, tu seras capable de :

- Créer des objets avec l'opérateur new
- Appeler des méthodes d'objet
- Déclarer des méthodes de classe manipulant des objets
- Écrire des tests unitaires validant des fonctions

Plus généralement, tu seras capable de créer des objets et de les manipuler en Java.

Pour y parvenir, nous supposons que tu as pris connaissance des concepts vus pendant les deux premières séances de POO et que tu as fait les exercices correspondants 😊.

## Préalable – reprise en main d'Eclipse

⌚ Durée : 5 min

Avant d'importer le projet, tu dois créer un espace de travail, qui est un conteneur de projets. Nous allons créer un espace de travail pour les labos de POO.

### Créer l'espace de travail

Pour créer un espace de travail, ouvre Eclipse. Par défaut, le lanceur d'Eclipse te propose de choisir un répertoire comme espace de travail (Figure 1). Choisis un chemin que tu retrouveras facilement avec ton explorateur de fichiers.

Choisis, par exemple :

- c:\Users\[username]\Documents\Ecole\Pri\Poo sous Windows.
- /Users/[username]/Documents/Ecole/Pri/Poo sous Mac.
- /home/[username]/Documents/Ecole/Pri/Poo sous Linux.

[username] est à remplacer par ton nom d'utilisateur.

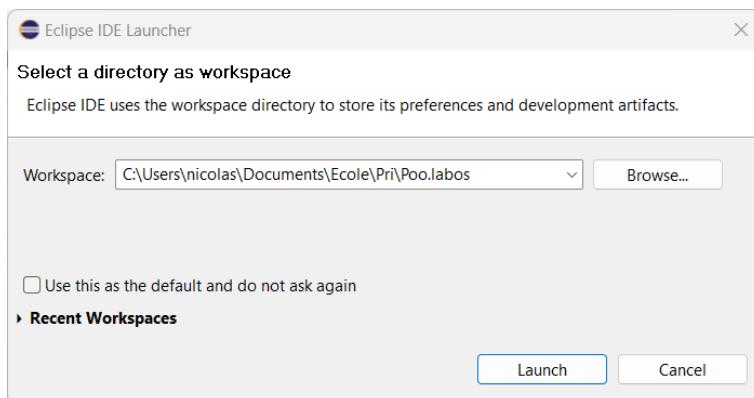


Figure 1 Lanceur Eclipse

### 💣 Quelques pièges à éviter

Choisis un chemin sans espaces, ni accents : certains EDI n'aiment pas cela.

Évite aussi les dossiers synchronisés comme un dossier OneDrive ou un dossier Dropbox qui peuvent causer des conflits avec les fichiers compilés.

Si le lanceur Eclipse n'apparaît pas au lancement, tu peux créer un espace de travail depuis le menu *File>Switch Workspace>Other...* (Figure 2)

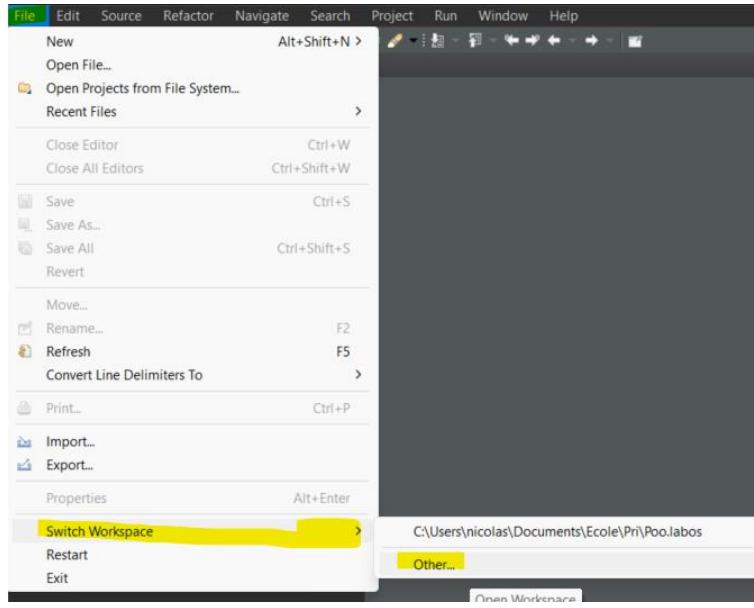


Figure 2 Changement d'espace de travail

Une fois l'espace de travail défini, tu pourras y importer tes projets.

### Cibler Java 21

Nous coderas avec la version 21 du JDK. Pour éviter tout problème de compatibilité, indique à Eclipse que ton espace de travail cible le JDK 21.

Pour ce faire, exécute l'item *Window > Preferences*. Une boîte de dialogue s'affiche. Ouvre l'onglet *Java* et clique sur *Compiler*. Dans la boîte de choix *Compiler compliance level*, choisis 21.



Applique les changements et ferme la boîte de dialogue.

### Importer le projet

Tu dois à présent importer le projet dans ton espace de travail. Commence par télécharger le stub disponible sur HELMo-Learn. Les figures suivantes détaillent la procédure d'importation.

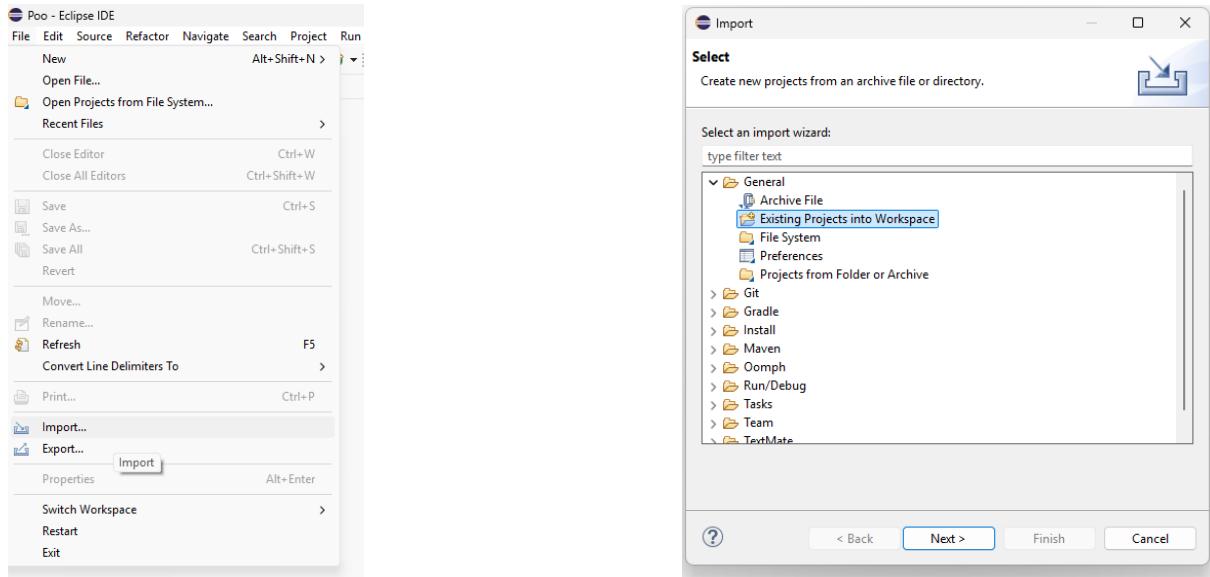


Figure 3 étape 2 choisir l'item General > Existing ...

Figure 4 étape 1 exécuter l'item File > Import...

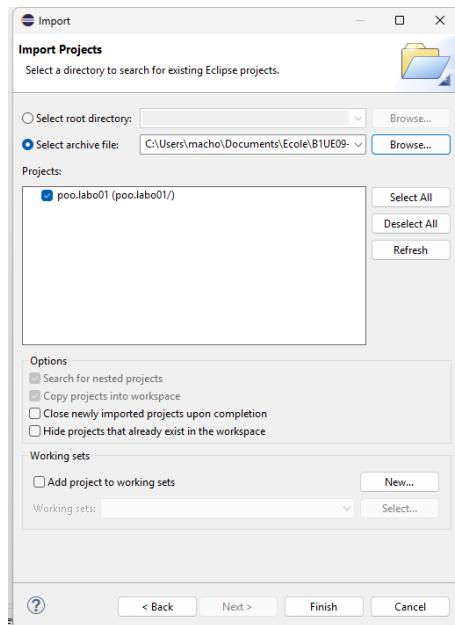


Figure 5 étape 3 choisir une archive, naviguer vers le fichier zip et cliquer sur Finish

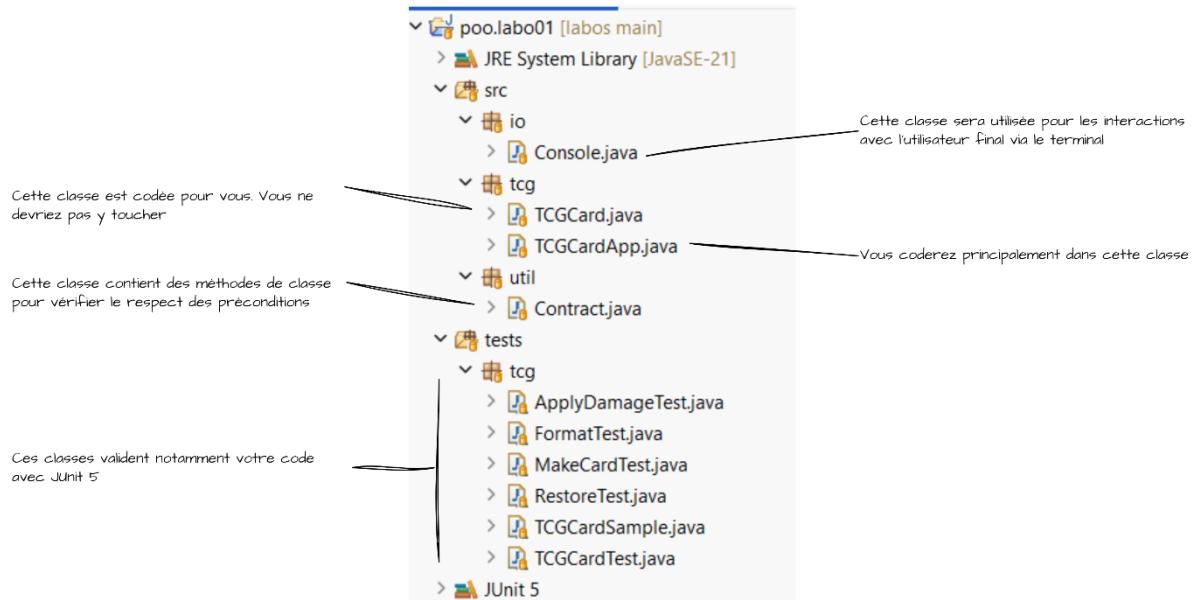


Figure 6 Structure du projet

## Exercices

Le laboratoire propose des exercices de difficulté croissante. Après chaque exercice, le responsable propose une solution et répond aux questions.

⚠️ Les temps indiqués dans les exercices sont indicatifs. En outre, ils ne tiennent compte ni des explications de l'énoncé, ni du temps pris pour discuter des solutions.

Nous voulons un programme pour un deck de cartes à collectionner (« *TCG Card* »). Le tableau suivant liste quelques exemples de cartes à collectionner et leurs données.

Nom	Texte	PV (HP)	Mana (MP)	Attaque (ATK)	Défense (DEF)
<b>Dragon</b>	Une créature légendaire qui crache du feu et domine le champ de bataille.	150	20	9	7
<b>Golem de Pierre</b>	Une créature solide qui absorbe les dégâts pour ses alliés.	200	10	5	10
<b>Chevalier Royal</b>	Un guerrier loyal qui protège ses alliés avec honneur.	120	18	7	6
<b>Loup-Garou</b>	Attaque rapidement et se régénère pendant la nuit.	90	14	8	3
<b>Mage des Ombres</b>	Utilise la magie noire pour infliger des dégâts indirects.	80	25	6	2
<b>Phénix</b>	Ressuscite avec la moitié de ses points de vie après avoir été vaincu.	180	30	10	5

Nous t'avons fourni une classe qui définit des objets `TCGCard`. D'autres classes utilitaires comme `Contract` et `Console` t'aideront dans ta tâche.

## Exercice 1 : mise en forme

### ⌚ Objectifs :

- Appeler des méthodes d'objet
- Écrire des tests unitaires

### ⌚ Durée : 20 min

La méthode de classe `TCGCardApp.format(TCGCard)` retourne un `String` représentant la carte reçue en argument. Elle appelle des méthodes de l'objet à cette fin. Nous voulons améliorer sa présentation avec des barres de vie et de mana, comme illustré ci-dessous.

*Le résultat de la fonction avant tes modifications*

```
Carte : Chevalier Royal  
Un guerrier loyal qui protège ses alliés avec  
honneur.  
PV : 60/120  
PM : 8/18  
ATK : 7 - DEF : 6
```

*Le résultat de la fonction après tes modifications*

```
Carte : Chevalier Royal  
Un guerrier loyal qui protège ses alliés avec  
honneur.  
PV : ████ ███████ 60/120  
PM : ███ □□□ 8/18  
ATK : 7 - DEF : 6
```

Pour construire ces barres, tu dois calculer les rapports HP/HP Max et MP/MP Max. En les multipliant respectivement par 10 et par 5, tu obtiens les nombres de « █ » à afficher. Les « □ » complètent les premiers rectangles pour en compter respectivement 10 et 5.

Exemple : reprenons la carte « Chevalier Royal »

$$\#\blacksquare_{hp} = 10 \times card_{hp} \div card_{hpmax} = 10 \times 60 \div 120 = 5, \#\square_{hp} = 10 - \#\blacksquare_{hp} = 5$$

$$\#\blacksquare_{mp} = 5 \times card_{mp} \div card_{mpmax} = 5 \times 8 \div 18 = 2, \#\square_{mp} = 5 - \#\blacksquare_{mp} = 3$$

### 💡 Suggestion

La méthode d'objet `String.repeat(int count)` retourne le string qui reçoit l'appel répété `count` fois. Appelle cette fonction pour générer les barres de vie et de mana et éviter des boucles.

Exemples

```
"a".repeat(3) retourne "aaa"
```

```
"be".repeat(2) retourne "bebe"
```

Valide tes adaptations avec le test « `FormatTest1 should return a string with card data` ». Ajoute un second test, inspiré du test précédent, où tu crées une carte de ton choix, réduis ses points de vie et de mana et vérifies que le résultat de la fonction correspond à tes attentes avec des assertions.

Vérifie enfin que le paramètre ne vaut pas `null`. Appelle à cette fin la méthode `Contract.requireNonNull(boolean, String)`, étudiée dans l'atelier 2. Tu peux valider cette vérification avec le test « `FormatTest should reject null arg` ».

### ❓ Question

Où placeras-tu cette vérification, sachant qu'elle correspond à une obligation de la méthode appelante (précondition) ?

<sup>1</sup> Tu trouveras cette classe dans le répertoire de sources « `tests` ».

## Exercice 2 : créer une nouvelle carte

### ⌚ Objectifs :

- Appeler des méthodes d'objet
- Déclarer des fonctions utilitaires

### ⌚ Durée : 20 min

La méthode de classe `TCGCardApp.makeCard()` interagit avec l'utilisateur avec la console pour obtenir les données d'une carte, instancier la classe `TCGCard` et retourner l'objet créé.

Attention, les arguments d'une nouvelle carte sont soumis aux contraintes ci-dessous. L'utilisateur doit encoder des valeurs correctes sous peine d'exceptions lancées à la création de l'objet.

Propriété	Contrainte 1	Contrainte 2
<b>Nom</b>	Pas null	Non blanc
<b>Texte</b>	Pas null	Non blanc
<b>HP et HP max</b>	$\geq 10$	$\leq 999$
<b>MP et MP max</b>	$\geq 5$	$\leq 99$
<b>ATK et DEF</b>	$\geq 1$	$\leq 10$

Pour les éviter, tu dois valider les données correspondantes avant d'instancier la classe `TCGCard`. Nous souhaitons que la méthode répète la demande d'encodage dès qu'une donnée n'est pas valide, comme avec l'exemple suivant.

```
Nouvelle carte
-----
Nom :Dragon
Texte :Une créature légendaire qui crache du feu et domine le champ de bataille.
HP [10; 999] :1000
HP [10; 999] :150
MP [5; 99] :-1
MP [5; 99] :20
ATK [1; 10] :9
DEF [1; 10] :7
```

Valide tes adaptations avec les tests définis dans « `MakeCardTest` ». Pour que ces derniers fonctionnent, utilise la classe `Console` du projet.

### 💡 Info

Un string est blanc quand il est vide ou composé uniquement de caractères d'espacement. La méthode d'objet `isBlank()` retourne true si le string qui reçoit l'appel est blanc.

Exemples

```
" ".isBlank(); // true
"\t\n".isBlank(); // true
```

### 💡 Suggestion

Déclare une fonction utilitaire `readNonBlank(String message)` qui répète la demande d'un nouveau String jusqu'à ce que ce dernier soit non blanc.

Déclare une fonction utilitaire `readInt(String message, int min, int max)` qui répète la demande d'un nouvel entier jusqu'à ce que ce dernier appartienne à l'intervalle `[min ; max]`.

Ces fonctions t'éviteront de répéter des instructions semblables et de complexifier la fonction `makeCard()`.

### Exercice 3 : endommager une carte

#### ⌚ Objectifs :

- Appeler des méthodes sur le bon objet
- Écrire des tests unitaires

#### ☒ Durée : 45 minutes.

Déclare une méthode de classe `TCGCardApp.applyDamage(TCGCard attacker, TCGCard target)` dont le type de retour est `int` et dont le corps implémente les opérations suivantes :

- Vérifier que les deux paramètres soient définis avec la fonction `Contract.require(boolean, String)`.
- Calculer les points de mana consommés par l'attaquant avec la formule suivante  
$$\text{mana consommés} = \max(1, \frac{\text{attacker}_{\text{attack}} + \text{attacker}_{\text{defense}}}{4})$$
- Vérifier que l'attaquant dispose d'assez de mana pour causer des dégâts.  
Pour y arriver, appelle la méthode `getManaPoints()`. Si l'attaquant ne possède pas assez de points de mana, la fonction arrête son exécution et retourne -1.
- Calculer les points de dégâts que l'attaquant cause à la cible avec la formule suivante  
$$\text{dégâts causés} = 4 \times \text{attacker}_{\text{attack}} - 2 \times \text{target}_{\text{defense}}$$
- Retirer les points de dégâts des points de vie de la cible  
Appelle la méthode `setHealthPoints(int newHp)` en veillant au respect des éléments suivants :
  - Cette méthode écrase les points de vie précédents. Tu dois donc transmettre comme argument les points de vie de la cible auxquels tu as retiré les points de dégâts.
  - Cette méthode lance une exception si l'argument est négatif ou dépasse les points de vie maximum. Adapte les points de dégâts en conséquence.
- Retirer les points de mana consommés de l'attaquant  
Pour ce faire, appelle la méthode `setManaPoints(int newMp)` en veillant au respect des éléments suivants :
  - Cette méthode écrase les points de mana précédents. Tu dois donc transmettre comme argument les points de mana de l'attaquant auxquels tu as soustrait les points de mana à retirer.
- Retourner les points de dégâts effectivement causés.

Le test « `ApplyDamageTest should pass the happy path` » doit réussir. Ajoute des tests pour vérifier, entre autres choses, que :

- Ta fonction produit les effets escomptés pour un second cas normal (« *happy path* »).
- Ta fonction ne mute pas les cartes si l'attaquant n'a pas assez de points de mana.
- Ta fonction « tue » la carte cible quand les points de dégâts dépassent les points de vie restants.

- Ta fonction ne cause aucun point de dégâts quand la carte cible a une défense forte et que l'attaque possède une attaque faible.

### Suggestion

La fonction [Math.clamp\(int val, int min, int max\)](#) peut t'aider à ramener val entre min et max.

Exercice 4 : restaurer les cartes

### Objectifs :

- Appeler des méthodes sur le bon objet
- Travailler avec des tableaux, notamment des tableaux d'objet
- Écrire des tests unitaires

 Durée : 45 minutes.

Déclare une méthode de classe TCGCardApp.restore(TCGCard[] deck, int cardsCount) dont le type de retour est int[] et dont le corps implémente les opérations suivantes :

- Vérifier que deck est défini et que count est compris dans l'intervalle [0 ; deck.length].
- Pour i allant de 0 à cardsCount - 1,
  - Si la carte i du deck est morte, lui remplir ses points de vie.
  - Si la carte i du deck n'a plus assez de points de mana, lui remplir ses points de mana.
 

 *Les points de mana d'une carte sont insuffisants s'ils sont plus petits que ceux consommés quand elle attaque (voir Exercice 3 : endommager une carte).*
- Retourner deux entiers : le premier correspond au nombre de fois que les points de vie ont été restaurés et le second, au nombre de fois que les points de mana ont été restaurés.

Valide ta méthode en écrivant au moins trois tests. Par exemple, quel est le résultat retourné pour un tableau de cinq cartes dans lequel aucune n'a besoin d'être restaurée ? De même, que vaut le résultat pour un tableau de trois cartes dans lequel toutes doivent être restaurées en points de vie et de mana ?

### Suggestion

Le calcul des points de mana consommés quand une carte attaque peut faire l'objet d'une fonction `getConsumedMana(TCGCard)`, ce qui t'évitera de dupliquer ce code.

Exercice 5 : rassembler les morceaux

### Objectif : intégrer des objets dans un programme

 Durée : 60 minutes.

Dans la fonction principale de TradingCardApp, déclare un tableau pour cinq cartes à collectionner.

Affiche ensuite un menu composé des items suivants :

1. Ajouter une carte s'il reste des slots disponibles  
*Appelle la méthode makeCard et ajoute la carte à la position suivante du tableau.*
2. Afficher le deck  
*Affiche le résultat des appels à format(TCGCard) sur chaque carte créée*

3. Attaquer  
Demande à l'utilisateur deux choisir deux cartes. La première carte choisie attaque la seconde.
4. Restaurer le deck  
*Appelle la méthode restore(TCGCard[]) et affiche son résultat*
5. Terminer l'exécution

Appelle les méthodes créées aux endroits que tu juges adéquats. Essaie de reproduire l'exécution suivante.

Exemple d'exécution

```

POO - Labo 1
=====
1. Ajouter une carte [5 slots disponibles]
2. Afficher le deck
3. Attaquer
4. Restaurer le deck
5. Terminer l'exécution

Votre choix ?1

Nouvelle carte
-----
Nom :Dragon
Texte :Une créature légendaire qui crache du feu et domine le champ de bataille.
HP [10; 999] :1000
HP [10; 999] :150
MP [5; 99] :-1
MP [5; 99] :20
ATK [1; 10] :9
DEF [1; 10] :7

Carte Dragon ajoutée

1. Ajouter une carte [4 slots disponibles]
2. Afficher le deck
3. Attaquer
4. Restaurer le deck
5. Terminer l'exécution

Votre choix ?1

Nouvelle carte
-----
Nom :Golem de Pierre
Texte :Une créature solide qui absorbe les dégâts pour ses alliés.
HP [10; 999] :200
MP [5; 99] :10
ATK [1; 10] :5
DEF [1; 10] :10

Carte Golem de Pierre ajoutée

1. Ajouter une carte [3 slots disponibles]
2. Afficher le deck
3. Attaquer
4. Restaurer le deck

```

## 5. Terminer l'exécution

Votre choix ?1

Nouvelle carte

-----

Nom :Loup-Garou

Texte :Attaque rapidement et se régénère pendant la nuit.

HP [10; 999] :90

MP [5; 99] :14

ATK [1; 10] :0

ATK [1; 10] :8

DEF [1; 10] :42

DEF [1; 10] :3

Carte Loup-Garou ajoutée

1. Ajouter une carte [2 slots disponibles]

2. Afficher le deck

3. Attaquer

4. Restaurer le deck

5. Terminer l'exécution

Votre choix ?3

1. Dragon    2. Golem de Pierre 3. Loup-Garou

Carte qui attaque : 1

Carte qui subit : 3

Dragon attaque Loup-Garou

Loup-Garou a subi 30 points de dégâts

1. Ajouter une carte [2 slots disponibles]

2. Afficher le deck

3. Attaquer

4. Restaurer le deck

5. Terminer l'exécution

Votre choix ?2

-----

Carte : Dragon

Une créature légendaire qui crache du feu et domine le champ de bataille.

PV : ████ ████████ ████ 150/150

PM : ████ ████ □ 16/20

ATK : 9 - DEF : 7

-----

Carte : Golem de Pierre

Une créature solide qui absorbe les dégâts pour ses alliés.

PV : ████ ████████ ████ ████ 200/200

PM : ████ ████ ████ 10/10

ATK : 5 - DEF : 10

-----

Carte : Loup-Garou

Attaque rapidement et se régénère pendant la nuit.

PV : ████ ████ ████ ████ □ □ □ □ 60/90

PM : ■■■■■ 14/14  
ATK : 8 - DEF : 3

---

1. Ajouter une carte [2 slots disponibles]
2. Afficher le deck
3. Attaquer
4. Restaurer le deck
5. Terminer l'exécution

Votre choix ?3

1. Dragon    2. Golem de Pierre 3. Loup-Garou

Carte qui subit : 3

Dragon attaque Loup-Garou

Loup-Garou a subi 30 points de dégâts

1. Ajouter une carte [2 slots disponibles]
2. Afficher le deck
3. Attaquer
4. Restaurer le deck
5. Terminer l'exécution

Votre choix ?3

1. Dragon    2. Golem de Pierre 3. Loup-Garou

Carte qui attaque : 1

Carte qui subit : 3

Dragon attaque Loup-Garou

Loup-Garou a subi 30 points de dégâts

Loup-Garou a perdu la vie

1. Ajouter une carte [2 slots disponibles]
2. Afficher le deck
3. Attaquer
4. Restaurer le deck
5. Terminer l'exécution

Votre choix ?2

---

Carte : Dragon

Une créature légendaire qui crache du feu et domine le champ de bataille.

PV : ■■■■■■■■■■ 150/150

PM : ■■□□□ 8/20

ATK : 9 - DEF : 7

---

Carte : Golem de Pierre

Une créature solide qui absorbe les dégâts pour ses alliés.

PV : ■■■■■■■■■■ 200/200

PM : ■■■■■ 10/10

ATK : 5 - DEF : 10

---

Carte : Loup-Garou

Attaque rapidement et se régénère pendant la nuit.

PV : ████ 0/90

PM : ████ 14/14

ATK : 8 - DEF : 3

---

1. Ajouter une carte [2 slots disponibles]

2. Afficher le deck

3. Attaquer

4. Restaurer le deck

5. Terminer l'exécution

Votre choix ?4

1 carte avec leurs HP restaurés

0 carte avec leurs MP restaurés

---

1. Ajouter une carte [2 slots disponibles]

2. Afficher le deck

3. Attaquer

4. Restaurer le deck

5. Terminer l'exécution

Votre choix ?2

---

Carte : Dragon

Une créature légendaire qui crache du feu et domine le champ de bataille.

PV : ██████████ 150/150

PM : ████ 8/20

ATK : 9 - DEF : 7

---

Carte : Golem de Pierre

Une créature solide qui absorbe les dégâts pour ses alliés.

PV : ██████████ 200/200

PM : ████ 10/10

ATK : 5 - DEF : 10

---

Carte : Loup-Garou

Attaque rapidement et se régénère pendant la nuit.

PV : ██████████ 90/90

PM : ████ 14/14

ATK : 8 - DEF : 3

---

1. Ajouter une carte [2 slots disponibles]

2. Afficher le deck

3. Attaquer

4. Restaurer le deck

5. Terminer l'exécution

Votre choix ?5

Au revoir