Rapport de Projet : **Shadow Adventurers**

PAR : KAMLI Amira et ATTIGUI Fatima Zohra

4décembre 2024

**Table des matières**

1. [Introduction](#_bookmark0) 2
2. [Structure des Fichiers et Commandes](#_bookmark1) 2
   1. [Structure des dossiers](#_bookmark2) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
   2. [Commandes pour compiler et exécuter](#_bookmark3) . . . . . . . . . . . . . . . . . . . . . 2
3. [Description des Fichiers et Améliorations](#_bookmark4) 2
   1. [Attaquable.java](#_bookmark5) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
   2. [Personnage.java](#_bookmark6) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
   3. [Guerrier.java](#_bookmark7) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
   4. [Mage.java](#_bookmark8) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
   5. [Voleur.java](#_bookmark9) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
   6. [Cleric.java](#_bookmark10) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
4. [Tests et Validation](#_bookmark11) 5
   1. [Example de Test : Simulation de Combat](#_bookmark12) . . . . . . . . . . . . . . . . . . . . 5
   2. [Analyse des Calculs - Tour 1 et Tour 7](#_bookmark13) . . . . . . . . . . . . . . . . . . . . . 6
      1. [Tour 1](#_bookmark14) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
      2. [Tour 7](#_bookmark15) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
      3. [Récapitulatif des Calculs](#_bookmark16) . . . . . . . . . . . . . . . . . . . . . . . . . 8
      4. [Conclusion](#_bookmark17) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
5. [Conclusion](#_bookmark18) 9

# Introduction

Le projet **Shadow Adventurers** est un jeu de combat au tour par tour où les joueurs incarnent un des personnages légendaires : Guerrier, Mage ou Voleur. Chaque classe offre des compétences uniques, des styles de combat variés et des forces/faiblesses spécifiques.

Ce rapport présente :

* La structure du projet et les commandes pour l’exécuter.
* Les améliorations et extensions apportées aux fichiers existants.
* La nouvelle classe ajoutée : Cleric.
* Les tests effectués pour valider le jeu.

# Structure des Fichiers et Commandes

Le projet est organisé comme suit :

## Structure des dossiers

* **src/** : Contient les fichiers source .java.
  + **Game/** : Tous les fichiers .java nécessaires au jeu.
* **bin/** : Contient les fichiers compilés .class.
  + **Game/** : Tous les fichiers compilés nécessaires au jeu.

## Commandes pour compiler et exécuter

1

# Compiler tous les fichiers . java dans le dossier bin javac -d bin src/ Game */\*. java*

*# Exécuter le jeu avec le fichier Play02 java -cp bin Game. Play02*

2

3

4

5

# Description des Fichiers et Améliorations

### Attaquable.java

**Description** : Interface définissant deux méthodes principales : attaquer et utiliserCompetence.

Ces méthodes sont implémentées par les classes dérivées de Personnage.

*// Interface definissant les comportements d’ attaque et de competence*

*.*

public interface Attaquable {

void attaquer( Personnage cible );

void utiliserCompetence ( Personnage cible );

}

1

2

3

4

5

### Personnage.java

**Description** : Classe abstraite servant de base pour tous les personnages. Elle contient :

* Attributs communs : nom, pointsDeVie, degats, niveau, experience.
* Méthodes pour infliger des dégâts (recevoirDegats) et gérer l’expérience (gagnerExperience).

**Améliorations apportées** :

* **Ajout de l’attribut** energie : Chaque personnage consomme de l’énergie pour uti- liser ses compétences spéciales.
* **Ajout des coups critiques** : Tous les personnages ont une probabilité de doubler leurs dégâts.
* Méthode utiliserEnergie : Vérifie si un personnage peut consommer de l’énergie.

*// EXTENSION: Ajout de l’ attribut energie et de la gestion des coups*

*critiques.*

protected int energie ;

public boolean estCritique () {

return Math . random () < 0.1; *// 10% de chances de coup critique*

}

public boolean utiliserEnergie ( int

if ( energie >= cout) { energie -= cout; return true ;

} else {

System . out. println ( nom + " return false ;

}

}

cout) {

n’a pas assez d’ energie !");

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

### Guerrier.java

**Description** : Classe dérivée représentant un guerrier avec des attaques physiques puis- santes.

**Améliorations apportées** :

* Ajout de la gestion des coups critiques dans les attaques.
* La compétence spéciale Coup de rage consomme de l’énergie.

*// EXTENSION: Ajout de la gestion de l’ energie pour Coup de rage.*

@ Override

public void utiliserCompetence ( Personnage cible ) { if ( utiliserEnergie (20)) {

System . out. println ( nom + " utilise Coup de rage !"); cible . recevoirDegats( degats \* 2);

}

}

1

2

3

4

5

6

7

8

### Mage.java

**Description** : Classe dérivée représentant un mage utilisant des sorts magiques.

**Améliorations apportées** :

* Régénération automatique de l’énergie chaque tour.
* Gestion des coups critiques.
* La compétence spéciale Tempête magique consomme de l’énergie.

*// EXTENSION: Regeneration automatique de l’ energie par tour.*

public void regenererEnergie () {

energie += 5; *// Ajout automatique de 5 points d’ energie.*

System . out. println ( nom + " regenere 5 points d’ energie .");

}

1

2

3

4

5

### Voleur.java

**Description** : Classe dérivée représentant un voleur spécialisé dans les attaques furtives.

**Améliorations apportées** :

* Augmentation de la probabilité de coup critique à 20%.
* Gestion de l’énergie pour utiliser Attaque rapide.

*// EXTENSION: Gestion de l’ energie et augmentation des chances de critique.*

@ Override

public void attaquer( Personnage cible ) { if ( Math . random () < 0.2) {

System . out. println ( nom + " inflige un coup critique !"); cible . recevoirDegats( degats \* 2);

} else {

cible . recevoirDegats( degats);

}

}

1

2

3

4

5

6

7

8

9

10

### Cleric.java

**Description** : Nouvelle classe représentant un soigneur capable de restaurer ses points de vie ou ceux d’un allié.

**Code** :

*// EXTENSION: Classe Cleric specialisee dans le soin.*

public void utiliserCompetence ( Personnage cible ) { if ( utiliserEnergie (15)) {

System . out. println ( nom + " utilise sa competence speciale : Soin !");

pointsDe Vie += 20;

}

}

1

2

3

4

5

6

7

# Tests et Validation

Les tests suivants ont été réalisés pour valider le projet *Shadow Adventurers* et garantir le bon fonctionnement des nouvelles fonctionnalités et améliorations apportées :

* **Création des personnages** : Vérification des attributs initiaux (*Points de Vie*, *Dé- gâts*, et *Énergie*) pour chaque classe.
* **Attaques et compétences spéciales** : Validation des dégâts infligés et de la consom- mation d’énergie lors de l’utilisation des compétences spéciales.
* **Gestion de l’énergie** : Tests sur la consommation et la régénération d’énergie pour les classes correspondantes (*Mage*, *Cleric*).
* **Coup critique** : Vérification des chances de coup critique pour chaque classe (10% pour la plupart, 20% pour le *Voleur* ).
* **Action Défendre** : Validation de la mécanique de défense, réduisant les dégâts reçus au tour suivant.
* **Robustesse** : Tests de gestion des entrées utilisateur incorrectes dans Play02.java.

## Example de Test : Simulation de Combat

Le test suivant simule un combat complet entre les quatre personnages (*Guerrier*, *Mage*, *Voleur*, *Cleric*) afin de valider la gestion des attaques, des compétences spéciales, de l’éner- gie et du calcul des dégâts. Le combat se termine lorsque l’un des personnages est déclaré vainqueur.

Listing 1 – TestCombatSimulation.java

|  |  |  |  |
| --- | --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | package Game ;  public class TestCombatSimulation {  public static void main ( String [] args) {  *// Initialize characters*  Personnage guerrier = new Guerrier(" Thor"); Personnage mage = new Mage (" Gandalf"); Personnage voleur = new Voleur(" Loki"); Personnage cleric = new Cleric(" Eowyn ");  *// Combat simulation loop*  int turn = 1;  while ( isAny CharacterAlive ( guerrier , mage , voleur , cleric)) { System . out. println ("--- Turn " + turn + " ---"); | | |
| 15 |  | if | ( guerrier. estVivant ()) guerrier. attaquer( mage ); |
| 16 |  | if | ( mage . estVivant ()) mage . utiliserCompetence ( guerrier); |
| 17 |  | if | ( voleur. estVivant ()) voleur. utiliserCompetence ( cleric) |
|  |  |  | ; |
| 18 |  | if | ( cleric. estVivant ()) cleric. utiliserCompetence ( cleric) |
|  |  |  | ; |
| 19 |  | tur | n ++; |
| 20 | } |  |  |
| 21 |  |  |  |

22

*// Determine winner*

if ( guerrier. estVivant ()) System . out. println (" The winner is Thor!");

}

private static boolean isAny CharacterAlive ( Personnage ... personnages) {

for ( Personnage p : personnages) if (p. estVivant ()) return true ;

return false ;

}

}

23

24

25

26

27

28

29

30

**Résultats du Test**

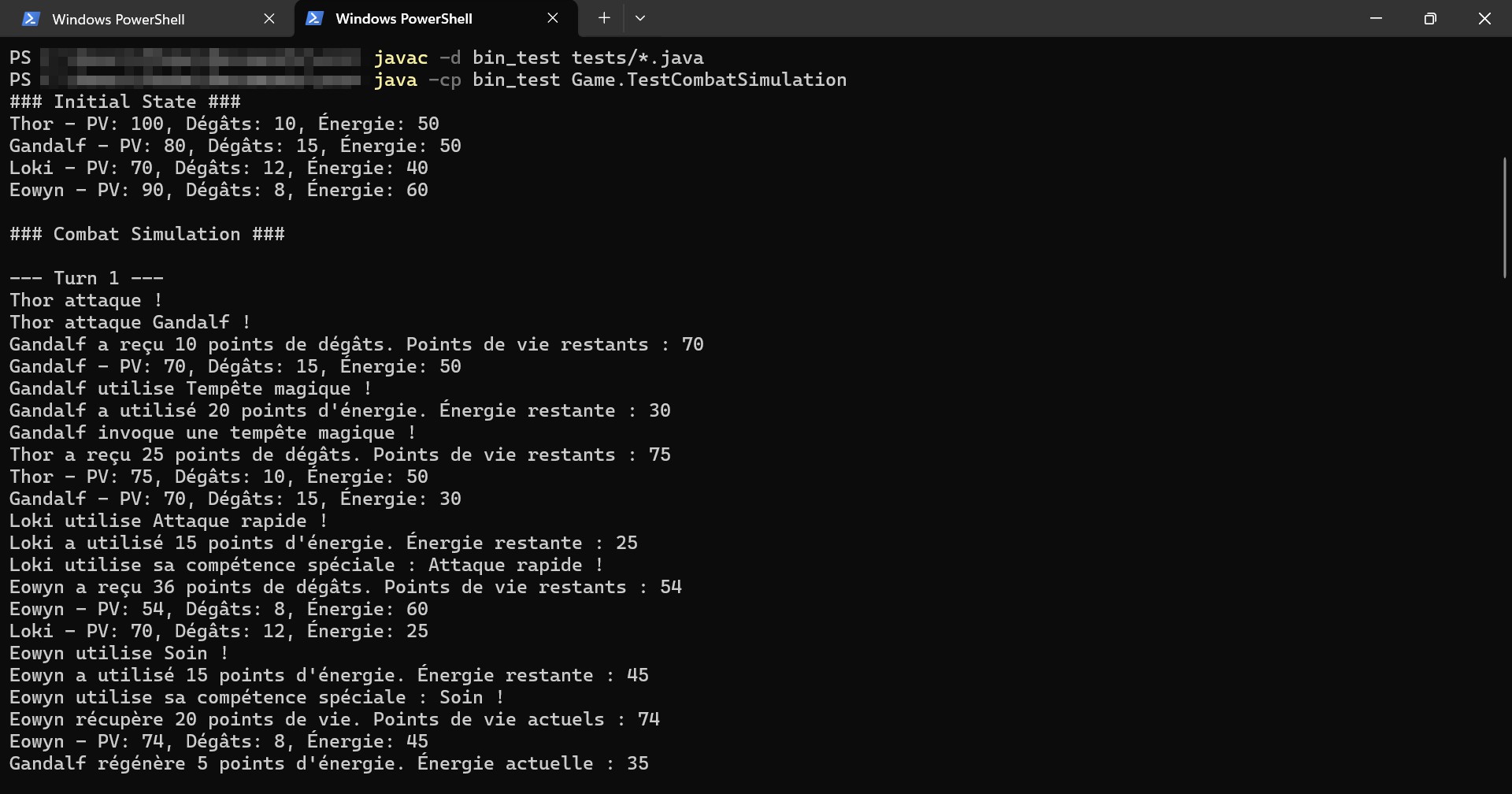
****

Figure 1 – Résultat du test de simulation de combat dans le terminal (Turn 1).

## Analyse des Calculs - Tour 1 et Tour 7

Cette section détaille les calculs des *Points de Vie (PV)*, *Dégâts*, et *Énergie* des per- sonnages pendant le **Tour 1** et le **Tour 7**. Les références au code associé sont également fournies.

#### Tour 1

* + - * **Thor attaque Gandalf :**
        + Code : guerrier.attaquer(mage);

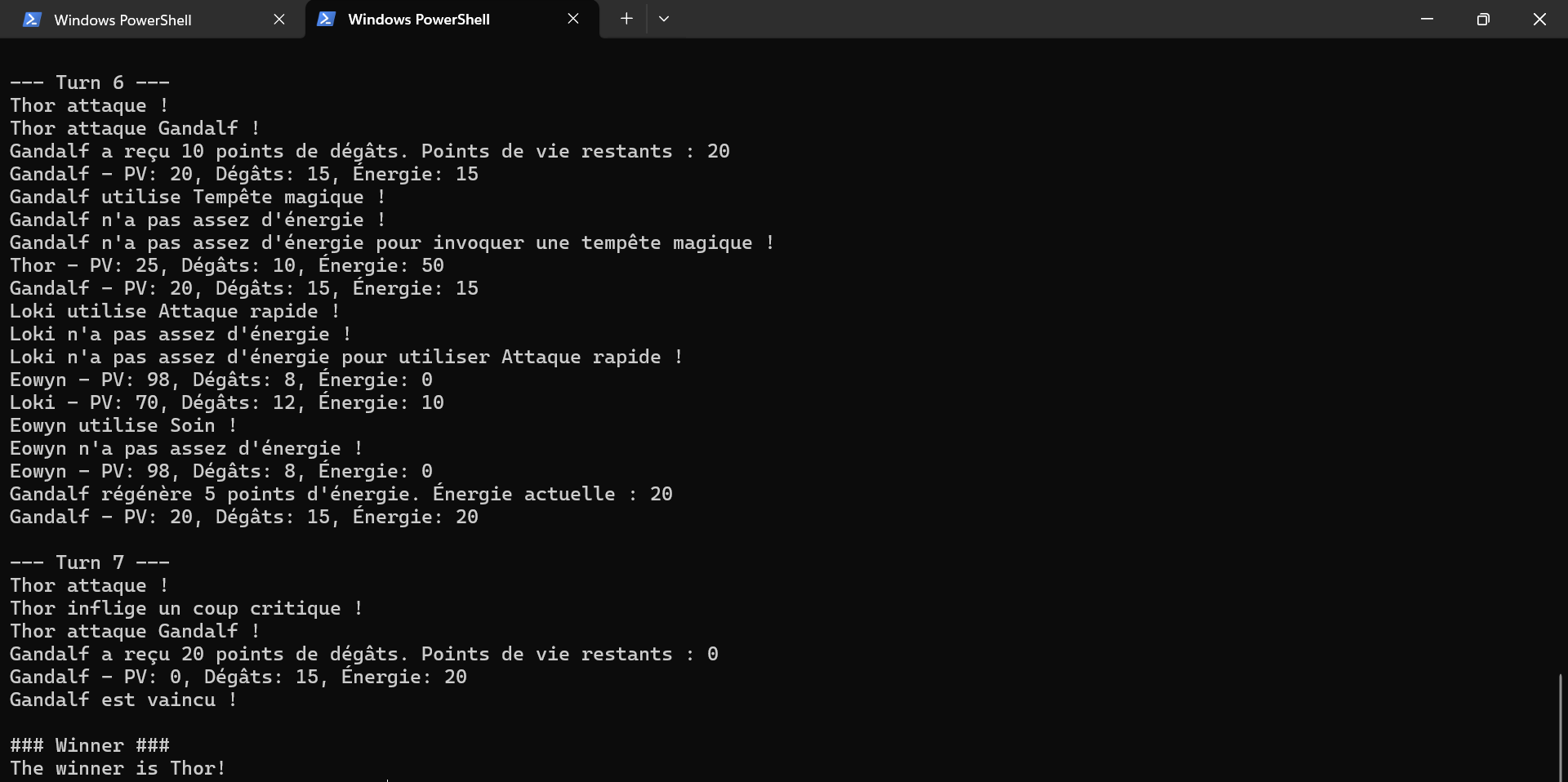


Figure 2 – Résultat du test de simulation de combat dans le terminal (Turn 7).

* + - * + Méthode utilisée : Personnage::recevoirDegats
        + Calcul : Thor inflige degats = 10, donc les PV de Gandalf passent de 80 à :

*PVGandalf* = 80 *−* 10 = 70

* + - * **Gandalf utilise Tempête magique sur Thor :**
        + Code : mage.utiliserCompetence(guerrier);
        + Méthode utilisée : Mage::utiliserCompetence
        + Calcul des dégâts : degats + 10 = 15 + 10 = 25. Les PV de Thor passent de 100 à :

*PVThor* = 100 *−* 25 = 75

* + - * + Calcul de l’énergie : Gandalf consomme 20 énergie, donc :

*EnergieGandalf* = 50 *−* 20 = 30

* + - * **Loki utilise Attaque rapide sur Eowyn :**
        + Code : voleur.utiliserCompetence(cleric);
        + Méthode utilisée : Voleur::utiliserCompetence
        + Calcul des dégâts : degats \* 3 = 12 \* 3 = 36. Les PV d’Eowyn passent de 90 à :

*PVEowyn* = 90 *−* 36 = 54

* + - * + Calcul de l’énergie : Loki consomme 15 énergie, donc :

*EnergieLoki* = 40 *−* 15 = 25

* + - * **Eowyn utilise Soin :**
        + Code : cleric.utiliserCompetence(cleric);
        + Méthode utilisée : Cleric::utiliserCompetence
        + Calcul des soins : Eowyn récupère 20 PV. Ses PV passent de 54 à :

*PVEowyn* = 54 + 20 = 74

* + - * + Calcul de l’énergie : Eowyn consomme 15 énergie, donc :

*EnergieEowyn* = 60 *−* 15 = 45

#### Tour 7

* + - * **Thor attaque Gandalf :**
        + Code : guerrier.attaquer(mage);
        + Méthode utilisée : Personnage::recevoirDegats
        + Calcul : Thor inflige un **coup critique**, donc les dégâts sont doublés :

*degatscritique* = 10 *×* 2 = 20

Les PV de Gandalf passent de 20 à :

*PVGandalf* = 20 *−* 20 = 0

* + - * + Gandalf est vaincu.
      * **Autres personnages (Loki et Eowyn)** :
        + Code : voleur.utiliserCompetence(cleric) et cleric.utiliserCompetence(cleric);
        + Loki et Eowyn n’ont plus assez d’énergie pour utiliser leurs compétences spéciales.

Aucune action n’est effectuée.

#### Récapitulatif des Calculs

|  |  |  |  |
| --- | --- | --- | --- |
| **Personnage** | **Action** | **PV** | **Énergie** |
| Thor | Attaque Gandalf (coup critique) | 25 (inchangé) | 50 (inchangé) |
| Gandalf | Aucun effet (vaincu) | 20 *→* 0 | 20 (inchangé) |
| Loki | Aucun effet (manque d’énergie) | 70 (inchangé) | 10 (inchangé) |
| Eowyn | Aucun effet (manque d’énergie) | 98 (inchangé) | 0 (inchangé) |

Table 1 – Résumé des calculs pour le Tour 7.

#### Conclusion

Pendant le **Tour 7**, Thor inflige un coup critique à Gandalf (guerrier.attaquer(mage)), infligeant 20 points de dégâts et éliminant Gandalf. Les autres personnages, Loki et Eowyn, sont incapables de contribuer au combat en raison du manque d’énergie. **Thor** est donc déclaré vainqueur.

# Conclusion

Le projet **Shadow Adventurers** a été enrichi de nouvelles fonctionnalités et amélio- rations. Les ajouts (comme le Cleric, l’énergie, et les coups critiques) rendent le jeu plus interactif et stratégique. Tous les ajouts sont clairement marqués dans le code pour faciliter leur compréhension.