La joute de Monseigneur Node

Le tournoi de combattants

version 1.3.1





1. Introduction

Monseigneur Node est un honorable prêtre JavaScript de Villeville qui a créé sa propre branche: NodeJS. Ayant voué sa vie à répandre sa vision du JavaScript au travers de ses nombreux écrits. Sa réputation est désormais connue des contrées de l’Islande jusqu’aux confins de la Bulgarie. Respecté et honoré, Monseigneur Node vécut heureux, jusqu’à ce jour fatidique où il découvrit un cheveu gris.

Il comprit alors que le temps de passer le flambeau était arrivé et donc qu’il devrait trouver un successeur. Alors que vous faisiez le tour des stands du marché, vous entendez alors : “C’EST MON DISCIPLE !”. En effet, c’est vous qu’il désignait.

Après vous avoir amené jusqu’à son lieu de culte où il vous a initié et formé. Maintenant qu’il sait que vous détenez les connaissances suffisantes, il souhaite les tester !

Aujourd’hui, vous devrez réaliser un projet reprenant différentes notions existantes dans le JavaScript telles que les classes et objets, les fonctions traditionnelles, anonymes et fléchées, la portée et types des variables ainsi que les boucles et les conditions.

Sur ce, bonne chance !

1. Consignes

* Pour ce projet-là, il vous sera demandé de créer un repository avec le nom : cc\_joute\_de\_monseigneur\_node.
* N’oubliez pas de push régulièrement !
* En cas de question, pensez à demander de l’aide à votre voisin de droite. Puis de gauche. Ou inversement. Puis demandez enfin à un Cobra si vous êtes toujours bloqué(e).
* Pensez à faire valider chaque partie que vous réaliserez à un Cobra lorsque vous l’aurez terminée.
* N’hésitez pas à faire des bonus et à ajouter des fonctionnalités lorsque votre projet sera terminé et validé.
* Vous avez tout à fait le droit d’utiliser internet pour trouver des réponses ou pour vous renseigner.

1. Sujet

Monseigneur Node veut que vous organisiez un tournoi. Plus précisément, il souhaite que vous organisiez un tournoi de combattants, des formes d’intelligences artificielles (**IA**). Les combats seront sous la forme d’un contre un.

Durant un match, lorsqu’un combattant attaquera, il infligera des dégâts égaux à sa force moins les points d’armure de son adversaire. Chaque match ne pourra durer plus de 20 tours. Dans le cas où la vie d’aucun combattant n’a atteint 0, celui avec le meilleur score (total de toutes les propriétés qui sont des nombres) sera déclaré vainqueur.

À la fin de chaque combat, le gagnant se verra remettre ses points de vie au maximum et gagnera une valeur aléatoire de points d’expérience entre 3 et 7 qui lui permettront de passer au niveau supérieur lorsque ceux-ci seront au nombre minimum de 10.

En passant de niveau, le combattant gagnera 10 points de vie, 5 points d’armure et 3 points de force !

Quand un combattant perd un combat, il est éliminé du tournoi et peut donc être effacé de la liste des participants.

Les combattants devront être issus d’une classe nommée **Personnage**.

Ils devront disposer d’un nom, de points de vie, de points d’armure et de points de force. Mais aussi de points d’expérience qui seront à 0 et d’un niveau qui sera défini à 1 au début du tournoi.

Parmi tous les combattants, l’un d’eux devra être issu du prototype d’un autre et portera le même nom, à la différence que le nom de celui-ci commencera par **“**Copie de **”**.

Les combattants seront tous stockés dans un tableau qui devra être stocké dans une classe **Tournoi**.

Dans cette classe **Tournoi**, devra se trouver une méthode **start()** qui permettra de lancer le tournoi et fera office de présentateur du tournoi. De ce fait, devra se comporter comme tout présentateur en présentant l’évènement ainsi que les différents combattants avec leurs caractéristiques.

N’hésitez pas à ajouter des lignes de texte durant le combat qui pourront décrire les actions réalisées, vainqueur de chaque match ou même décrire les différentes actions !

1. Indices

Indice n°1 : définition de la classe Personnage

La première étape va être de définir tout ce dont vous aurez besoin dans notre classe. Dans le sujet, il est question de **points de vie**, **points d’armure**, de **points de force**. Mais aussi de **points d’expérience**, de **niveau** et sans oublier d’**un nom** ! Il est aussi intéressant de lui attribuer une méthode **description** qui reprendrait les différents attributs du combattant.

Indice n°2 : définition de la classe Tournoi

La première chose quand on fait une classe, c’est de déclarer son constructeur. D’après le sujet, il prend uniquement la liste des combattants.

Dans la consigne, il est mentionné une fonction **start()**. C’est cette méthode que se déroulera le programme.

Il vous faudra aussi une méthode qui permettra de faire les combats. Celle-ci devrait prendre deux arguments, le premier et le second combattant. Elle n’aura pas besoin de renvoyer quoi que ce soit puisque les éléments que vous lui enverrez en paramètre seront modifiés à l’origine.

Une méthode qui vous permettra de vérifier si un combattant peut augmenter de niveau et le faire si possible, serait utile. Elle prendrait en paramètre un combattant et ne renverrait rien.

La première chose à vérifier est le nombre de combattants. Si vous en avez 0 ou 1, il ne peut simplement pas y avoir de tournoi.

Indice n°2 : méthode start()

La première chose à vérifier est le nombre de combattants. Si vous en avez 0 ou 1, il ne peut simplement pas y avoir de tournoi.

Ensuite, comme le demande le sujet, il serait pertinent de faire le commentateur avec une présentation des différents combattants, etc…

L’étape suivante consiste à boucler tant qu’il y a plus d’un combattant dans notre liste de combattants. Dans cette boucle, on vérifiera d’abord l’état des combattants afin d’enlever ceux qui ne peuvent plus combattre. Juste derrière, on vérifiera le nombre de combattants dans la liste. S’il n’en reste qu’un, on utilisera le mot clé **break** qui servira à casser la boucle et à en sortir.

La dernière étape de notre méthode consiste à vérifier s’il y a bien au moins un élément dans notre liste et si c’est le cas, féliciter ce gagnant tout en faisant un petit récapitulatif de ces stats. Sinon on dit juste qu’il n’y a pas de vainqueur pour cette session.

Indice n°3 : vérification de l’état des combattants

Comment vérifier l’état des combattants puis les retirer de la liste au besoin ?  
Dans un premier temps, il vous faudra boucler sur chaque élément du tableau.

Puis regarder si la variable **peutCombattre** de cet élément est à **false**. Dans l’optique où vous avez défini cette variable dans votre classe Personnage.

Si c’est le cas, vous utiliserez la méthode **splice()** pour retirer l’élément de la liste. Cette méthode prend deux paramètres. La manière dont vous vous en servirez est la suivante : **splice(index, 1)**. C’est-à-dire l’index auquel se trouve notre élément et la quantité d’éléments que vous voulez retirer, soit 1.

Ensuite, vous remettez notre variable représentant notre index à 0, de sorte à ce qu’on soit sûr de n’oublier personne.

Indice n°4 : level up

Pour faire augmenter de niveau les statistiques d’un combattant, regardez d’abord si ses points d’xp sont **inférieurs à 10**, dans ce cas quittez la méthode puisqu’on ne pourra rien pour lui.

Sinon, on va retirer 10 points d’xp (on ne va pas lui laisser, sinon il pourrait augmenter de niveau à l’infini). Puis on va incrémenter sa variable niveau de 1, sa variable santé de 10, sa variable armure de 5 et sa variable force de 3.

Indice n°5 : le combat

La première chose à faire, c’est de créer deux variables (qui seront donc locales à cette méthode) qui vous permettrez de garder en mémoire la vie maximale de chacun des deux combattants (qui sont en arguments de la méthode).

L’étape suivante, c’est de définir une variable qui vous servira à dire combien d’xp remportera le gagnant du combat. Il faut que ce soit entre 3 et 7.

Regardez comment utiliser **Math.random()**

Enfin déclarez une variable qui vous servira à savoir à quel tour vous êtes rendu de ce combat (variable initialisée à 1) car 20 tours maximum autorisés par combat.

Maintenant, il va falloir faire une boucle qui va tourner tant que la vie des deux est supérieur à 0 et que le nombre de tours est inférieur à 21 (pas 20 sinon vous n’auriez que 19 tours).

L’étape suivante consiste à faire des actions qui vont influer sur la vie des combattants. Pour cela créez à nouveau deux variables (dans la boucle) qui vont servir à stocker les dégâts que vont recevoir chaque combattant durant ce tour.

Puis enlevez à la santé de chaque combattant, les dégâts que vous avez calculés précédemment.

Attention à ne pas enlever un nombre négatif, car enlever un nombre négatif revient à ajouter de la vie.

Ensuite, il vous faudra vérifier si le nombre de tour est égal à 21 et si la vie des deux combattants est supérieure à 0. Dans le cas où notre condition serait validées, vous regarderez ensuite lequel des deux combattants à le meilleur score (en appelant leur méthode **score()**). Dans le cas où c’est combattant1, vous définirez la vie de combattant2 à -1 (on pourrait définir à 0, mais moi j’aime bien -1 ).

Pour finir, on va regarder lequel des deux a le plus de vie (donc le gagnant). Si c’est combattant1, on remettra sa vie à son maximum (qu’on a stocké au départ). Puis lui donner l’xp qu’il a remporté. On va aussi définir la variable **peutCombattre** du combattant2 à **false** du coup. Et pour finir on va appeler la méthode **levelUp()** de notre **classe Tournoi** en lui donnant combattant1 en paramètre pour s’assurer qu’il passe de niveau s’il peut ! Évidemment, il faudra faire un **else**, s’il s'agit du combattant2.

1. Conclusion

Voilà c’est la fin, Monseigneur Node est fier de votre travail et est heureux de voir la relève est assurée ! Cependant, Monseigneur Node est sûr qu’il reste encore plein d’autres aspects à découvrir ! N’hésitez pas à vous renseigner par vous-même.

Par ailleurs, s’il vous reste du temps, pourquoi n’essayeriez-vous pas de rajouter des éléments à votre travail ? Comme la possibilité d’utiliser des compétences, des objets qui pourraient être dans l’inventaire du combattant, faire de la coloration dans le terminal, etc...