Programmation web sur client

TD2 - Programmation orientée objet

Matthieu Nicolas

Exercice 1 - Point

Dans le fichier `/geometry/point.js`

1. Écrire la classe `Point`. Un `Point` possède deux propriétés, `x` et `y` qui représentent ses coordonnées. Le constructeur de la classe `Point` prend ces 2 paramètres pour créer une instance.
2. Écrire la méthode `translate` de la classe `Point`. Cette méthode prend 2 paramètres, `xShift` et `yShift`, et déplace les coordonnées du `Point` en conséquence.
3. Écrire la méthode `clone` de la classe `Point`. Cette méthode retourne une nouvelle instance de `Point` qui possède les mêmes coordonnées que l'instance actuelle.

Exercice 2 - Rectangle

Dans le fichier `/geometry/rectangle.js`

1. Écrire la classe `Rectangle`. Un `Rectangle` possède deux propriétés, `pointBL` et `pointTR` qui représentent respectivement le point en bas à gauche et le point en haut à droite du rectangle. Le constructeur de la classe `Rectangle` prend ces 2 paramètres pour créer une instance.
2. Écrire les getters `width` et `height` de la classe `Rectangle`. Ces getters calculent et retournent respectivement la largeur et la hauteur de l'instance de `Rectangle`.
3. Écrire les getters `perimeter` et `area` de la classe `Rectangle`. Ces getters calculent et retournent respectivement le périmètre et l'aire de l'instance de `Rectangle`.
4. Écrire la méthode `translate` de la classe `Rectangle`. Cette méthode prend 2 paramètres, `xShift` et `yShift`, et déplace les coordonnées du `Rectangle` en conséquence.
5. Écrire la méthode `clone` de la classe `Rectangle`. Cette méthode retourne une nouvelle instance de `Rectangle` qui possède les mêmes coordonnées que l'instance actuelle.

Exercice 3 - Character

Dans le fichier `/characters/character.js`

1. Écrire la classe `Character`. Un `Character` possède trois propriétés, `name`, `hp` et `hpMax` qui représentent respectivement le nom du personnage, son nombre de points de vie courant et son nombre de points de vie maximal. Le constructeur de la classe `Character` prend 2 paramètres pour créer une instance, `name` et `hp`. Le paramètre `hp` représente la valeur initiale pour les propriétés `hp` et `hpMax`.

2. Écrire la méthode `doDamage` de la classe `Character`. Cette méthode ne prend aucun paramètre et se contente de retourner le nombre `10`.

3. Écrire la méthode `takeDamage` de la classe `Character`. Cette méthode prend 1 paramètre, `damage` et retire le montant indiqué au nombre de points de vie courant du personnage.

Exercice 4 - Warrior

Dans le fichier `/characters/warrior.js`

1. Écrire la classe `Warrior`. Cette classe hérite de `Character`. Un `Warrior` possède une propriété additionelle, `rage`. Le constructeur de la classe `Warrior` prend les mêmes paramètres que celui de `Character` et initialise en plus la propriété `rage` à 0.

2. Écrire la méthode `doDamage` de la classe `Warrior`. De base, cette méthode se comporte comme celle de la classe `Character`. Cependant, si le `Warrior` a plus de 20 de `rage`, elle consomme ce montant pour augmenter les dégâts effectués de 30%.

3. Écrire la méthode `takeDamage` de la classe `Warrior`. De base, cette méthode se comporte comme celle de la classe `Character`. En plus, elle incrémente la `rage` du `Warrior` du montant de dégâts subi.

Exercice 5 - Mage

Dans le fichier `/characters/mage.js`

1. Écrire la classe `Mage`. Cette classe hérite de `Character`. Un `Mage` possède plusieurs propriétés supplémentaires: `mp`, `mpMax` et `threshold`. Le constructeur de la classe `Mage` prend les paramètres `mp` et `threshold` en plus des paramètres du constructeur de `Character`. Le paramètre `mp` représente la valeur initiale pour les propriétés `mp` et `mpMax`.

2. Écrire la méthode `doDamage` de la classe `Mage`. Si le `Mage` a suffisamment de points de magie, il consomme `20 mp` pour infliger `15` points de dégâts. Sinon, son attaque par défaut inflige `80%` du montant de l'attaque de `Character`, et lui restaure `5` points de magie.

3. Écrire la méthode `takeDamage` de la classe `Mage`. De base, cette méthode se comporte comme celle de la classe `Character`. En plus, si les points de vie du `Mage` passent sous la valeur de `threshold`, il dépense `20` points de magie pour se soigner `15` points de vie, si possible.

Exercice 6 - typeof et instanceof

1. Exécuter les morceaux de codes contenus dans le dossier `/typeof/`.
   Observer les différents résultats obtenus.
   Que pouvez-vous en conclure sur le comportement et limites de `typeof`?

2. Exécuter les morceaux de codes contenus dans le dossier `/instanceof/`.
   Observer les différents résultats obtenus.
   Que pouvez-vous en conclure sur le comportement de `instanceof`?