Étude des indices de masse corporelle

L'indice de Masse Corporelle (IMC) d'une personne de masse corporelle (poids) p kg et de taille t mètres est : $IMC = p \times t^2$. Il permet entre autre de savoir si une personne est en surpoids (IMC trop élevé) ou au contraire trop maigre (IMC trop faible). On veut faire des utilitaires qui permettent de manipuler des personnes et d'étudier leur IMC.

Vous exécuterez les programmes avec Node : ouvrez une console Node, rendez vous dans votre dossier de travail et tapez `node fichier.mjs` pour exécuter le code javascript écrit dans le module `fichier.mjs`. Vous utiliserez `console.log()` pour afficher à l'écran.

Vous essaierez d'être le plus concis possible (sans que ce soit au détriment de la clarté du code).

Personne

Écrivez le module `personne.mjs` qui exporte la classe `Personne` représentant une personne : une personne stocke son identifiant `id` (une chaîne), son poids `poids` (masse en kg) et sa taille `taille` (en mètres) ; elle est construite à partir de ces trois données ; elle a un accesseur `IMC` (et non une méthode `IMC()`) qui renvoie son indice de masse corporelle. **Écrivez** le module `affichagepersonne.mjs` qui exporte par défaut la fonction `affichagePersonne` qui renvoie une chaîne de caractères décrivant la personne passée en paramètre sous la forme de l'exemple suivant : `Personne P36EE41, 46.5 kg, 1.52 m : IMC 107.4336` (**utilisez** les chaînes interpolées).

Écrivez le module `testPersonne.mjs` qui contient la procédure `test1` qui crée la personne d'identifiant `TEST001` pesant 70,3 kg et de taille 1,74 m et l'affiche à l'écran, et qui exécute cette procédure. **Testez**. **Ajoutez** dans le module `testPersonne.mjs` la procédure `test2` qui crée un objet ayant pour propriétés `id` valant `TEST002`, `poids` valant 62,4 et `taille` valant 1,62, crée la personne correspondante et l'affiche à l'écran, et faites la exécuter par le module. **Testez**.

Utilisation d'objets et JSON

On risque de se tromper dans l'ordre des paramètre quand on crée une personne et on voudrait pourvoir sérialiser une personne en JSON et inversement créer une personne à partir de son JSON.

Modifiez le constructeur de `Personne` pour qu'il prenne explicitement en paramètre un objet ayant les propriétés `id`, `poids` et `taille` contenant les données de la personne et ajoutez une méthode de classe `depuisDonnes(id, poids, taille)` qui renvoie une personne à partir des données passées en paramètres.

Adaptez les procédures de test `test1` et `test2` pour qu'elles utilisent le constructeur modifié et ajoutez la procédure de test `test3` qui crée la personne `TEST003` pesant 55,3 kg et de taille 1,54 m et l'affiche à l'écran en utilisant `depuisDonnes`. **Testez**.

On veut qu'une personne soit représentée au format JSON sous la forme du JSON d'un objet ayant les propriété `id`, `poids` et `taille`. **Ajoutez** dans la classe `Personne` la méthode de classe `depuisJSON` qui prend en paramètre le JSON d'un tel objet et crée et renvoie la personne correspondante. **Ajoutez** la procédure de test `test4` qui contient la chaîne représentant le JSON de la personne `TEST004` pesant 56,3 kg et de taille 1,5 m, crée une personne à partir de ce JSON et l'affiche. **Testez**. **Ajoutez** la procédure `test5` qui crée la personne `TEST005` pesant 55,3 kg et de taille 1,54 m, puis crée une autre personne à partir de son JSON et l'affiche. **Testez**.

Tableaux de personnes

Écrivez le module `affichagespersonnes.mjs` qui exporte la fonction `afficherTableauPersonnes` qui affiche chaque personne du tableau de personnes passées en paramètre sous la forme `[`*id*`, `*poids*`, `*taille*`]`. **Écrivez** le module `testPersonnes.mjs` qui contient la procédure `test1` qui crée un tableau de deux personnes et l'affiche. **Testez**. Ajoutez dans le module `affichagespersonnes.mjs` et exportez la fonction `afficherPersonnes` qui prend un nombre variable de personnes en paramètre et les affiche. **Ajoutez** la procédure `test2` qui crée deux personnes et les affiche. **Testez**.

La méthode `forEach` des tableaux prend en paramètre une fonction `f` et applique `f` à chaque élément du tableau. **Modifiez** la fonction `afficherTableauPersonnes` pour qu'elle utilise `forEach` et une fonction anonyme. **Testez**.

À partir d'un tableau de personnes on veut extraire les données (les poids et les tailles) de différentes façons. **Écrivez** le module `donneespersonnes.mjs` qui exporte les fonctions `poids` et `tailles` qui prennent en paramètre un tableau de personnes et renvoie respectivement le tableau des poids et des tailles de ses personnes : vous utiliserez `Array.from` et des fonctions anonymes. **Écrivez** le module `testdonnees.mjs` qui contient la procédure de test `test1` qui crée un tableau de personnes et qui affiche le JSON du tableau des poids et du tableau des tailles de ces personnes. **Testez**.

On veut maintenant extraire les poids et les tailles. **Ajoutez** et **exportez** dans le

module `donneespersonnes.mjs` la fonction `donnees` qui prend en paramètre un tableau de personnes et renvoie un tableau d'objets ayant pour propriétés `poids` et `taille` contenant les poids et tailles de personnes. **Ajoutez** dans `test1` l'affichage du JSON du tableau des données. **Testez**.

Génération de page

Regardez la page `htmlpersonnes.html` qui affiche les données de personnes dans un tableau. En reprenant la structure de cette page, **écrivez** le module `htmlpersonnes.mjs` qui exporte la fonction `htmlPersonnes` qui prend en paramètre un tableau de personne et renvoie une chaîne de caractère contenant la page HTML correspondante (vous utiliserez les chaînes interpolées qui peuvent être imbriquées et les méthodes adéquates des tableaux pour n'utiliser ni boucle ni concaténation explicite de chaînes avec +). **Écrivez** le module `testhtml.mjs` qui contient la procédure de test `test1` qui crée un tableau de personnes et qui affiche la page HTML de ces personnes. **Testez**.