Prise en main : Ouvrez un fichier `tp5.js` avec votre éditeur de texte favori. Ouvrez un fichier `tp5.html`, et recopiez-y le code suivant :

```
<!DOCTYPE HTML>
<html>
    <body>
        <script src="tp5.js"></script>
    </body>
</html>
```

Pour exécuter le code que vous y écrirez, il faudra ouvrir le fichier `tp5.html` dans un navigateur.

(a) La délicieuse **Morille**, reconnaissable entre toutes à son chapeau alvéolaire.

(b) La **Golmotte**, ou **Amanite Rougissante**, qu'on reconnaît à ses tâches roses ou vineuses, et qui est un excellent comestible.

(c) L'**Amanite Panthère**, qu'on évitera de confondre avec la golmotte. Celle-ci cause des indigestions graves, pouvant (dans de rares cas) être mortelles.

FIGURE 1 – Les trois variétés de champignons qui nous occupent.

Question 1 (classes et objets)

En évitant au maximum toute duplication de code, écrivez les classes `Champignon`, `Morille`, `Amanite`, `Golmotte` et `Panthere`, sachant que

- les golmottes (aussi appelées *amanites rougissantes*) et les amanites panthères font partie de la famille des amanites,
- les amanites et les morilles sont des champignons,
- chaque champignon possède un `nom`, un `poids` (en grammes) et un drapeau booléen `comestible` (dans notre sélection, seules les amanites panthères sont toxiques),
- de plus, une des manières de différencier les golmottes et les panthères est de s'intéresser à la présence de stries sur le bord du chapeau. On dote donc les amanites d'un attribut booléen supplémentaire, `stries`, qui sera `true` chez les amanites panthères, et `false` chez les golmottes.

On fera attention à ne pas demander des informations inutiles dans les constructeurs ; par exemple, le constructeur de `Morille` n'attendra que le poids de la morille.

Réimplémentez la méthode `toString()` de `Champignon`, puis celle d'`Amanite` (sans repartir de zéro).

Question 2 (tableaux, boucle *for* et `filter`)

Écrivez, selon chacune des trois façons suivantes :

1. d'abord à l'aide d'une boucle *for* classique,
2. ensuite, en utilisant un *for..of*,
3. enfin, en une ligne, à l'aide d'un `filter()` et d'une lambda-expression,

une fonction `selection()` qui attend en argument un panier (i.e. un tableau de `Champignons`), et renvoie un nouveau panier, où tous les champignons toxiques ont été écartés.

Question 3 (`reduce`)

Si la consommation de golmotte n'est pas dangereuse à dose modérée (à condition de bien la cuire), certains cas d'intoxication ont été reportés en cas d'ingestion de tros grosses doses. Pour simplifier, on va considérer qu'il y a danger à consommer plus d'un kilogramme de golmotte en un repas.

Écrivez, selon chacune des deux façons suivantes :

1. d'abord à l'aide d'un *for..of*,
2. ensuite, en utilisant un `reduce()`,

une fonction `danger_golmottes()` qui attend en argument un panier, et renvoie un booléen indiquant si on risque de tomber malade en consommant les golmottes contenues dans le panier - autrement dit, s'il en contient plus d'un kilo.

Question 4 (prototypes) Reprendre la Question 1, mais en gérant l'héritage directement via les prototypes - le mot-clef `class` est interdit. On utilisera des fonctions-usines pour créer les objets. Par exemple, chaque appel à la fonction `usine_morille()` créera une `Morille`, dont les attribus seront passés en argument de la fonction-usine.