Les classes Event.php et Lagon.php ne nécessitaient pas de tests car elles ne contenaient qu'un constructeur et des fonctions de type getter et setter sans conditions particulières.

Tests sur la classe Anemones.php :

class AnemoneTest {

testNbJuvenile() **:** fait des assertions sur plusieurs objets de type Anemone pour vérifier que leur attribut actualCapacity diminue bien en fonction des bonnes conditions.

testRecruit() : fait des assertions sur plusieurs objets de type Anemone pour vérifier que l'attribut actualCapacity s'incrémente si l'anémone n'est pas morte.

testKill() : fait des assertions sur plusieurs objets de type Anemone pour vérifier que leur attribut actualCapacity diminue bien en fonction des bonnes conditions et que leur attribut isDead se mettent à jour si besoin (c'est à dire l'anémone est morte ou non).

Tous les tests sont passés avec succès pour cette classe.
}

Tests sur la classe Utils.php :

class UtilsTest {

testExplodeline() : fait des assertions pour vérifier la création d'un tableau de valeurs à partir d'une chaîne de caractères dans laquelle les valeurs sont séparés par des espaces

testGetArrayFromAnemoneIni() : fait des assertions pour vérifier que le tableau se crée bien à partir du fichier et avec les bonnes valeurs

testGetArrayFromLagonIni() : idem mais avec un fichier différent

testRandomFloat(): vérifie que le nombre aléatoire retourné est bien compris entre ses deux bornes.

testYearsToSeconds() : vérifie que les années sont convertis en secondes

testTireExp() : vérifie les bornes du résultat de la formule renvoyé

(La fonction fileFinding() de la classe Utils permet de récupérer une valeur précise dans le tableau de valeur $arrayOfLines en paramètre en fonction du paramètre $needle)

testFileFiding : vérifie donc que fileFinding() récupère la bonne valeur en fonction de $needle

Ce test ne passe pas lorsque :

- $arrayOfLines est un tableau vide : pour corriger cela, nous avons modifié le code de la fonction
- la fonction dépasse la limite du tableau $arrayOfLines : pas de solution pour cette erreur pour ce moment.
  Cette erreur entraîne parfois l'apparition de warnings sur la page.

pas de tests effectués sur sortingResults pour le moment mais on peut remarquer que si son paramètre $tmax vaut 0 alors la fonction va effectuer une division par zéro et donc lever une exception

pas de tests effectués sur les fonctions writeInFile(), writeInFileSorted() et appendInFile() qui modifient un fichier, nous n'avons pas trouvé comment écrire ceux-ci

lagonPopSorted_array() retourne un fichier sous la forme d'un tableau de tableau.
Dans cette fonction, on a :

- la variable $values un tableau où sont stockés les valeurs d'une ligne de fichier
- la variable $results un tableau de $values : c'est ce tableau qui est retourné par la fonction

Pour commencer, on peut remarquer grâce à un print_r que dans cette fonction, ils essayent de "nettoyer" le tableau de ses cases vides restantes mais ils se sont trompés d'une case, et cela enlève donc une valeur du tableau au lieu de retirer la case vide (unset($values[9]) au lieu de unset($values[10]))

Pour tester cette fonction, il aurait fallu faire testLagonPopSorted_array() qui vérifierait que le tableau se crée avec les bonnes valeurs (de la même façon que les fonctions GetArrayFromAnemoneIni() et testGetArrayFromLagonIni())

On peut donc se poser la question : n'aurait-il pas été possible et plus simple d'utiliser une fonction générale pour créer un tableau à partir de n'importe quel fichier ?

Mise à part les tests sur fileFinding(), les autres tests sont passés avec succès.
}

Pour les classes Horloge.php et Algo.php nous avons rencontré plusieurs problèmes qui nous ont empêché de tester correctement celles-ci.

Pour commencer, la classe Algo se base sur une simulation avec des temps donnés et des variables aléatoires, cela nous empêche donc de vérifier concrètement les résultats.
Ensuite, leur classes couplent le modèle et la vue donc le format MVC n'est pas très bien respecté : cela rend le code très long et compliqué à analyser puisque les calculs et l'affichage se font dans la même classe.
De plus,certaines fonctions publiques que nous avons testé sur le fichier dev/dev erroné sont devenus privées : les tests sont donc à refaire.
Pour résumer, pour pouvoir tester réellement ces classes sans les modifier, il aurait fallu vérifier le rendu de l'affichage, ce qui est quasi impossible.
La solution à notre problème aurait été de découpler certaines fonctions sur une autre classe pour pouvoir les tester individuellement, c'est ce que nous avons tenté de faire avec la classe Initalizer et PopulationCounter mais sans succès.

Malgré tout, la classe Anemone et Utils sont testables en grande partie.