Test unitaires

Il faut commencer par installer phpunit à l'aide de composer.

```
composer --dev require phpunit/phpunit
```

Il faut ensuite créer un dossier pour les classes et un dossier pour les tests. Pour lancer un test sur une classe, on utilise vendor/bin/phpunit dossier-test/ dans la console. Exemple d'une classe de test :

```
class MathTest extends PHPUnit\Framework\TestCase
{
    public function testDouble() {
        $this->assertEquals(5, \Grafikart\Math::double(2));
    }

    public function testDoubleIfZero() {
        $this->assertEquals(0, \Grafikart\Math::double(0));
    }
}
```

On peut aussi configurer phpunit à l'aide d'un fichier XML nommé phpunit.xml à la racine du projet. Exemple :

```
<?xml version="1.0" encoding="utf-8" ?>

<phpunit colors="true">
    <testsuite name="mes super tests">
        <directory>./tests</directory>
    </testsuite>
</phpunit>
```

Le fichier ci-dessus permet d'afficher les couleurs pour les tests unitaires et définit un dossier par défaut. On peut ainsi lancer nos tests unitaires avec un simple vendor/bin/phpunit

Concept de TDD : "Test Driven Development" : plutôt de créer une classe et de la tester, on va d'abord écrire les tests puis écrire notre classe.

La base des tests unitaires sont les **assertions**. On va simuler l'exécution de fonctions, puis voir si telle assertion est vérifiée ou non.

architecture :

> src
> SvgRenderer
> tests
> SvgRendererTest
> phpunit

Syntaxe : Les noms des classes doivent finir par "Test.php" et les noms des méthodes doivent commencer par "test"

Une méthode de test peut dépendre d'une autre classe de test :

```
/**
     * @depends testAddAttribute
     * @param HTMLBuilder $b
     */
```

Fournisseur de données : Une méthode peut être déclarée comme fournissant des données à une méthode de test.

1. on créée une source de données
2. on lie la source à la méthode de test
3. Les données sont utilisées comme paramètres de test
4.

Fixtures pour générer des fausses données

mock = faux objet

```
$o = createMock(A::class);
$f = function()...
$f->bindTo($o, $o);
```