# Correction du Q.C.M.

<= Retour au cours

-

**Question**

Laquelle de ces définitions correspond le plus au mot **trait** ?

- **Bonne réponse**

  Un trait est un regroupement de méthodes pouvant être importées dans une classe

- **Explications**

  Un trait est un regroupement de méthodes pouvant être importées dans une classe. Relisez a toute première partie pour vous en convaincre !

-

**Question**

Laquelle de ces affirmations est vraie ?

- **Bonne réponse**

  Une méthode définie dans une classe mère est écrasée par la méthode possédant le même nom dans le trait que la classe fille utilise

- **Explications**

  Retenez cette phrase : « *La classe est plus forte que le trait, le trait est plus fort que la mère* ». Ainsi, si une classe mère définie une méthode, celle-ci est écrasée par le trait utilisé par la classe fille si ce trait définie

cette même méthode.

- **Question**

  Un trait peut-il définir des attributs statiques ?

  - **Bonne réponse**

    Non

- **Explications**

  La réponse est non : un trait ne peut pas définir d'attribut statique.

- **Question**

  Ce code ne lève aucune erreur :

  **Code : PHP - Sélectionner**

```php
<?php
    trait MonTrait
    {
        protected $attr = 'Hello !';

        public function sayHello()
        {
            echo $this->attr;
        }
    }

    class MaClasse
    {
        use MonTrait;

        protected $attr = 'Hello !';
    }

    $objet = new MaClasse;
    $objet->sayHello();
```

  - **Bonne réponse**

    Faux : une erreur stricte est levée

- **Explications**

  Une erreur de niveau **E_STRICT** est ici levée car la classe déclare un attribut déjà existant dans le trait qu'elle utilise. L'erreur n'est pas fatale car l'attribut déclaré dans la classe a la même visibilité et la même valeur par défaut que l'attribut déclaré dans la classe.

**Question**

Ce code lève une erreur :

**Code : PHP - Sélectionner**

```php
<?php
    trait A
    {
        public function saySomething()
        {
            echo 'Je suis le trait A !';
        }
    }

    class MaClasse
    {
        use A
        {
            saySomething as sayWhoYouAre;
        }
    }

    $o = new MaClasse;
    $o->saySomething();
```

- **Bonne réponse**

  Non

- **Explications**

  Non, ce code ne lève pas d'erreur. L'alias créé **copie** la méthode sous un autre nom. La méthode est donc toujours accessible sous son ancien nom !

- **Question**

  Si je déclare une méthode abstraite dans un trait, toute classe utilisant ce trait sera obligée d'implémenter cette méthode, qu'elle soit abstraite ou non.

- **Bonne réponse**

  Faux

- **Explications**

  Ceci est faux. Si la classe utilisant le trait est abstraite, elle n'est pas obligée d'implémenter la méthode. Par contre, ses classes filles, oui.


Note : 20 / 20

<= Retour au cours