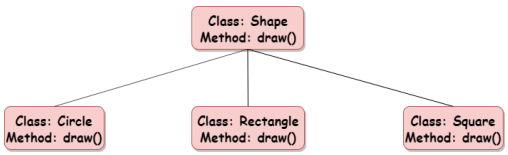
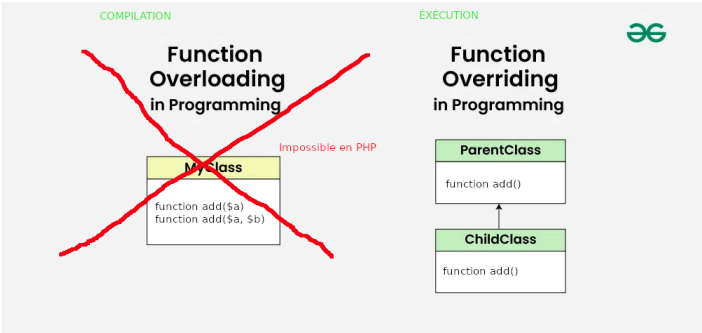
Polymorphisme en PHP



Le polymorphisme en PHP est un concept clé de la programmation orientée objet (POO) qui permet à une seule interface ou méthode d'être utilisée pour différents types d'objets. En d'autres termes, le polymorphisme permet d'utiliser une méthode ou une classe de manière générique, tout en conservant la possibilité pour des sous-classes ou des implémentations différentes de personnaliser ou de redéfinir son comportement.

Types de Polymorphisme en PHP



Contrairement à Java, PHP ne permet pas la surcharge de méthode:

```
function add($a);
function add($a, $b);
```

Avec les méthodes magiques ___call et ___callStatic, vous pouvez simuler une forme de surcharge dynamique de méthodes en PHP. Cependant, cette approche est évaluée à l'exécution et n'est pas vérifiée au stade de la compilation.

1. **Polymorphisme par héritage (ou polymorphisme de sous-classes)** :
   - Lorsqu'une classe enfant hérite d'une classe parent, elle peut redéfinir (override - remplacer) les méthodes de la classe parent. Les objets des classes enfant peuvent alors être traités comme

des objets de la classe parent, mais leur comportement peut être différent en fonction de la classe enfant spécifique.

- Exemple :

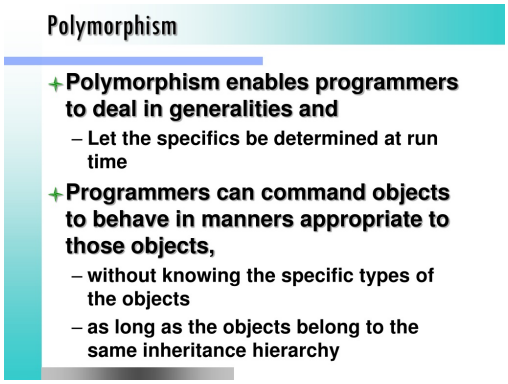
```
class Animal {
    public function faireDuBruit() {
        echo "L'animal fait un bruit";
    }
}

class Chien extends Animal {
    public function faireDuBruit() {
        echo "Le chien aboie";
    }
}

class Chat extends Animal {
    public function faireDuBruit() {
        echo "Le chat miaule";
    }
}

function faireBruit(Animal $animal) {
    $animal->faireDuBruit();
}

$chien = new Chien();
$chat = new Chat();

faireBruit($chien); // Le chien aboie
faireBruit($chat);  // Le chat miaule
```

2. **Polymorphisme par interface** :
   - Les interfaces en PHP permettent de définir un ensemble de méthodes que des classes peuvent implémenter. Une fois que différentes classes implémentent la même interface, elles peuvent être utilisées de manière interchangeable tout en ayant des implémentations différentes des méthodes définies par l'interface.
   - Exemple :

```
interface Outil {
```

```
    public function utiliser();
}

class Marteau implements Outil {
    public function utiliser() {
        echo "Utilisation du marteau pour enfoncer un clou";
    }
}

class Tournevis implements Outil {
    public function utiliser() {
        echo "Utilisation du tournevis pour visser";
    }
}

function utiliserOutil(Outil $outil) {
    $outil->utiliser();
}

$marteau = new Marteau();
$tournevis = new Tournevis();

utiliserOutil($marteau);    // Utilisation du marteau pour enfoncer un clou
utiliserOutil($tournevis);  // Utilisation du tournevis pour visser
```

Avantages du Polymorphisme



- **Flexibilité** : Le polymorphisme permet de créer du code plus flexible et plus réutilisable. Vous pouvez passer différents types d'objets à la

même méthode sans avoir à écrire du code spécifique pour chaque type d'objet.

- **Maintenance** : Le polymorphisme facilite la maintenance et l'extension du code. Si vous ajoutez un nouveau type d'objet, il suffit de créer une nouvelle classe ou implémentation sans toucher au code existant.
- **Abstraction** : Il permet de masquer les détails d'implémentation derrière une interface commune, ce qui simplifie la compréhension du code.

Conclusion

En résumé, le polymorphisme est un principe fondamental de la POO en PHP qui permet de traiter différents objets de manière uniforme tout en laissant à chaque objet le soin de définir son comportement spécifique.