L’héritage

L'héritage établit une relation de généralisation-spécialisation.

Possibilité d’une classe fille d’hériter des attributs et méthodes d’une classe mère. C’est une spécialisation de la classe fille par rapport à la classe mère. Une classe B hérite d'une classe A. La classe A est donc considérée comme la classe mère et la classe B est considérée comme la classe fille.

L’héritage

Lorsqu’une classe B hérite d’une classe A la classe B hérite de toutes les méthodes publiques et protégées et aussi des propriétés. Les méthodes privées d'une classe parente ne sont pas accessible à la classe enfant. En plus d’hériter des méthodes et attributs de la classe mère la classe enfant peut implémenter ses propres attributs et méthodes.

Une classe fille peut redéfinir les méthodes de sa classe mère afin de les modifier et les adapter à son comportement.

L’héritage

Pour procéder à un héritage (c'est-à-dire faire en sorte qu'une classe hérite des attributs et méthodes d'une autre classe), il suffit d'utiliser le mot-clé **extends**

```
class ClasseFille extends ClasseMere{
        //instructions
        ……
        …..
}
```

L’héritage

```
class ClassA{
        // déclaration des attribut de la classe
        private $attribut_1;
        // déclaration de méthodes de la classe
        public function nomMethode(){
                // instructions
        }
}
class ClassB extends ClassA{
}
```

ici ClassB hérite des attributs et méthodes de classA.

Redéfinition de méthode

```
class ClasseA{
        protected function calcul(){
                return 10;
        }
}
```

Redéfinition de méthode

```
class ClasseB extends ClasseA{
        public function calcul(){
                $retour = parent::calcul();
                if($retour > 100)
                        return "$retour est supérieur à 100";
                else
                        return "$retour est inférieur à 100";
        }
}
```

Le mot clé protected

Comme public et private le mot protected permet de définir la visibilité d’un attribut ou d’une méthode.

Quand un attribut ou une méthode est défini avec le mot clé protected il sera accessible dans toute classes filles de la classe qui la défini.

Classe abstraite

Une classe abstraite est une classe non instanciable. L'objectif est de représenter une factorisation, une généralisation. Dans de nombreux cas il est souhaitable de mettre certaines méthodes Par exemple les getters et les setters dans une classe abstraite.

syntaxe

```
abstract class ClassAbstraite {

        …

}
```

Méthode abstraite

Une méthode aussi peut-être abstraite avec la syntaxe suivante:

```
abstract protected function nomDeMethode();
```

Elle ne peut pas être implémentée dans la classe de base, elle fournit une signature, et devra être implémenté dans toutes les classes fille de la classe qui définit la méthode.

Une classe qui possède au moins une méthode abstraite doit être déclarée abstraite.

Classes et méthodes Final

Une classe final est une classe qui n'est pas héritable.

C'est la fin d'une chaîne d'héritage.

```
final class NomDeClasse {
        …
}
```

Une méthode final n'est pas surchargeable.

```
final public function NomDeFonction( … ) {
        …
}
```

Méthode statique, propriété statique

Le fait de déclarer des propriétés ou des méthodes comme statiques vous permet d'y accéder sans avoir besoin d'instancier la classe.

Comme les méthodes statiques peuvent être appelées sans qu'une instance d'objet n'ait été créée, la pseudo-variable $this n'est pas disponible dans les méthodes déclarées comme statiques.

pour déclarer une méthode ou une propriété statique on fait précéder le nom de la propriété ou méthode du mot clé static juste après le mot désignant sa portée.

Méthode statique, propriété statique

Les éléments statiques (static) sont des éléments des éléments de classe et non pas des éléments d'instances.

La syntaxe de déclaration d'une propriété statique est la suivante :

```
public static $nomDePropriete = "valeur_Initiale" ;
```

La syntaxe de déclaration d'une méthode statique est la suivante :

```
public static function méthode(){

    ...

}
```

Les constantes de classe

Comme pour les méthodes et propriété statiques les constantes de classe sont utilisables dans le script sans instanciation d'objet.

pour définir une constante de classe on utilise le mot clé const.

```
const NOM_DE_CONSTANTE = “valeur”;
```

L’opérateur ::

L'opérateur **::** , appelé opérateur de résolution de portée, permet d'accéder à des éléments statiques et aux constantes d’une classe.

```
NomClasse::NOMDECONSTANTE;

NomClasse::$nomDeVariableStatique;

NomDeClasse::nomDeMethodeStatique();
```

Le mot clé self

Le mot clé **self** et l'opérateur de résolution de portée permettent d'accéder aux éléments statiques et constantes de la classe depuis l'intérieur de celle ci.

```
self::NOMDECONSTANTE;

self::$nomDeVariableStatique;

self::nomDeMethodeStatique();
```

Le mot clé parent

Le mot clé **parent** et l'opérateur de résolution de portée permettent d'accéder aux attributs et méthodes statiques publique ou protégé d'une classe à partir de sa classe fille.

```
parent::NOMDECONSTANTE;

parent::$nomDeVariable;

parent::nomDeMethode();
```