CLASSE ABSTRAITE

```php
<?php

abstract class Car {
    private $model;
    private $color;
    private $maxSpeed;

    abstract public function gaz();
    abstract public function brake();

  //  Dans le monde réel, il n'y a pas de « voitures abstraites ».
  //  Il y a des camions, des voitures de course, des berlines..etc…


    public function getModel(): string {
            return $this->model;
    }

    public function getColor(): string {
        return $this->color;
    }
    public function getMaxSpeed(): string {
        return $this->maxSpeed;
    }

    public function setColor($color): self {
        return $this->color = $color ;
        return $this;
    }

    public function setModel($model): self {
        return $this->model = $model;
        return $this;
    }

    public function setMaxSpeed($speed): self {
        return $this->maxSpeed = $speed ;
        return $this;
    }
}

?>
```

Cette classe est déclarée abstraite car elle a 2 fonctions abstraites qui ne sont pas définies 'déclarées mais non définies.

Dans le monde réel, il n'y a pas de « voitures abstraites ». Il y a des camions, des voitures de course, des berlines..etc…

```php
<?php
class Berline extends Car {
    public function gaz() {
        echo "la Berline accelere ";
    }
    public function brake() {
        echo "la Berline freine ";
    }
}
?>
```

C'est très semblable à l'héritage. Mais les classes dont on hérite ne sont pas abstraites. dans ces leçons, les méthodes ne sont pas abstraites. Cependant, cette solution a un certain nombre d'inconvénients que les classes abstraites résolvent.

Pour commencer, on **ne peux pas créer une instance d'une classe abstraite**