The difference between abstract and interface

Interface

- Define a contract with no implementation.
  - Support multiple inheritance (a class can implement multiple interfaces).
  - Cannot have properties, only constants

Use interfaces when you want to define a contract that various classes must follow, especially when those classes are not related through a common ancestor. Interfaces are great for defining capabilities or behaviors that can be shared across different classes

```php
// Interface example
interface Shape {
    public function area(); // All methods are abstract and public
}

class Circle implements Shape {
    private $radius;

    public function __construct($radius) {
        $this->radius = $radius;
    }

    public function area() {
        return pi() * $this->radius * $this->radius; // Implementing interface
method
    }
}
```

Abstract

- Can provide both abstract and concrete methods.
- Support single inheritance (a class can extend only one abstract class).
- Can have properties and define visibility

Use abstract classes when you want to provide a common base with some shared functionality, while still enforcing that certain methods must be implemented in derived classes. Abstract classes are useful when you have a common base class that should not be instantiated directly

```php
// Abstract class example
abstract class Animal {
    protected $name;

    public function __construct($name) {
        $this->name = $name;
    }

    abstract public function makeSound(); // Abstract method

    public function getName() { // Concrete method
        return $this->name;
    }
}
```