Tell us about an abstract class and how it differs from an interface?

In PHP, an abstract class is a class that cannot be instantiated on its own but is meant to serve as a base class for other classes. An abstract class can contain abstract methods (methods without implementation) that must be implemented in child classes.

Here's an example of an abstract class in PHP:

```
abstract class Shape {
    abstract public function calculateArea();
}

class Circle extends Shape {
    private $radius;

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

    public function calculateArea() {
        return pi() * $this->radius * $this->radius;
    }
}

class Square extends Shape {
    private $side;

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

    public function calculateArea() {
        return $this->side * $this->side;
    }
}
```

An interface in PHP defines a contract that a class must fulfill. An interface can only include method signatures without implementations. A class implementing an interface must provide implementations for all methods defined in the interface.

Here's an example of using an interface in PHP:

```
interface Shape {
    public function calculateArea();
}

class Circle implements Shape {
    private $radius;

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

    public function calculateArea() {
        return pi() * $this->radius * $this->radius;
    }
}

class Square implements Shape {
    private $side;

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

    public function calculateArea() {
        return $this->side * $this->side;
    }
}
```

In summary, the main difference between an abstract class and an interface in PHP is that an abstract class can have implemented methods and properties and can be used for object instantiation, while an interface can only have method signatures and cannot be used for creating objects on its own.