PHP OOP Concepts Cheat Sheet

1. Class & Object

Class is a blueprint for creating objects. An object is an instance of a class.

Example:

```
class Car {
    public $color;
    public function drive() {
        echo "Driving";
    }
}
$car = new Car();
$car->drive();
```

2. Constructor & Destructor

Constructor (__construct) is called automatically when an object is created. Destructor (__destruct) runs when the object is destroyed.

Example:

```
function __construct() { echo "Starting"; }
function __destruct() { echo "Ending"; }
```

3. Encapsulation

Hiding internal object details and only showing necessary parts using access modifiers (private, protected, public).

4. Inheritance

Allows a class to inherit properties and methods from another class using `extends`.

```
class A {}
class B extends A {}
```

5. Polymorphism

PHP OOP Concepts Cheat Sheet

Same method name behaves differently in different classes.

Useful for method overriding.

6. Abstraction

Hides implementation details and shows only essentials.

Use abstract classes and abstract methods.

7. Interface

Defines method names only. A class must implement all methods. Used for contracts between unrelated classes.

8. Traits

PHP's way to reuse code across multiple classes (PHP doesnt support multiple inheritance).

Use `use TraitName;` inside class.

9. Magic Methods

Start with __ (double underscore). Examples:

__construct(), __destruct(), __call(), __get(), __set(), __toString(), __invoke()

10. Overloading

Dynamic creation of properties/methods using __get, __set, __call. PHP doesn't support method overloading like Java.

11. Overriding

Child class redefines a method from the parent class.

12. Static vs Non-Static

Static: Access using ClassName:: without object.

PHP OOP Concepts Cheat Sheet

Non-static: Requires object to access methods/properties.

13. Final Keyword

Prevents class or method from being extended/overridden.

final class A {}, final public function myFunc() {}

14. Namespaces

Avoids name conflicts by organizing code. Use `namespace MyApp;` at top of file.

15. Autoloading & PSR-4

Autoloading automatically includes class files.

PSR-4 maps namespace to folder path using Composer.