

INHERITANCE

PHP CLASSES

```
class Elephant
{
    private $name;

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

`$this->` refererar till det skapade objektets egenskaper

ACCESS

Public: default, öppet för alla

Private: kan enbart användas av klassen

Protected: kan användas av klassen och subklasser

```
class Elephant
{

    public static speak()
    {
        echo 'RUUUEEH!';
    }
}
```

```
Elephant::speak();
```

```
class Elephant
{
    public static $name = "Hathi"
    public static speak()
    {
        echo Self::$name; //Self is ref to class
    }
}
```

```
Elephant::speak();
```

INHERITANCE

Is a thing

Arv är när en klass utgår från en annan. Om den ena klassen är ett specialfall av den andra, så är det naturligt att låta dem ärva från varandra.

elephant (specifik) → mammal (mer allmän)
→ animal (mest allmän)

elephant (specifik) → mammal (mer allmän)
→ animal (mest allmän)

elephant ärver av mammal som ärver av animal

Allt som mammal kan kan även elephant

```
class Animal
{
    public function speak()
    {
        echo "I change shapes just to hide in this place"
    }
}
class Elephant extends Animal{}
```

Nyckelord: **extends**

OVERRIDE

```
class Animal {  
    public function makeNoise() { echo '...'; }  
}  
class Pig extends Animal {  
    public function makeNoise() { echo 'oink'; }  
}  
class Cat extends Animal {  
    public function makeNoise() { echo 'meow'; }  
}
```

Vi skriver över **basklassens** metoder

```
class Vehicle {  
    public $numberOfWheels;  
}  
class Car extends Vehicle {  
    public function __construct() {  
        $this->numberOfWheels = 4;  
    }  
}
```

Vehicle har redan numberOfWheels men denna variabel är olika beroende på vilken typ av Vehicle

Klassen Car ärver från Vehicle. Man säger ibland att Car **utökar** Vehicle.

Klassen Vehicle kallas för en **basklass** till Car.

Klassen Car kallas för en **subklass** till Vehicle.

Engelska: inherit, extend, base class, subclass.

OVERRIDE PARENT

```
class Vehicle {  
    public function goTo($destination) {}  
}  
class Car extends Vehicle {  
    public function goTo($destination) {  
        echo "Åker bil till $destination <br>";  
        parent::goTo($destination);  
    }  
}
```

parent:: kallar på basklassen

PROTECTED

Vi kan även använda `protected`

Egenskapen/metoden kan kommas åt av alla subklasser

Men inte av andra klasser eller utanför klasserna

```
class A {  
    public $p1;  
    protected $p2;  
    private $p3;  
}  
class B extends A {  
    public function whatWillHappen() {  
        echo $this->p1;    // 1  
        echo $this->p2;    // 2  
        echo $this->p3;    // 3  
    }  
}
```



```
class A {  
    public $p1;  
    protected $p2;  
    private $p3;  
}  
class B extends A {  
    public function whatWillHappen() {  
        echo $this->p1;    // public! open for everyone!  
        echo $this->p2;    // protected! Still open for B  
        echo $this->p3;    // private! FATAL ERROR  
    }  
}
```

ABSTRACT

/

INTERFACE

ABSTRACT

Ibland är ingenting logiskt

Klassen `Animal` behöver ju egentligen aldrig implementeras

Det finns inget random `Animal`

Ibland vill man bara ha klasser som referens eller som ett kontrakt:

`Abstract`

```
abstract class Animal {  
    abstract public function makeNoise();  
}  
class Cat extends Animal {  
    public function makeNoise() { echo 'meow'; }  
}  
class Dog extends Animal { } //FATAL ERROR!
```

Det är upp till subklassen att implementera funktionen

INTERFACE

Liknande med **interface**: ett kontrakt

En klass behöver inte vara en subklass men ska ändå uppnå vissa kriterer och ärva vissa drag

Vi kan då implementera ett **Interface**

```
interface iCanJump {  
    public function jump();  
}
```

```
class Cat implements iCanJump {  
    public function jump() { /**/ }    // ok  
}
```

```
class Human implements iCanJump {  
    public function jump() { /**/ }    // ok  
}
```

ABSTRACT / INTERFACE

Det jobbiga är att vi skriver inte någon faktiskt kod

Vi säger bara att vissa klasser **MÅSTE** ha viss kod

Vi underlättar för andra och för oss själva: genom att skapa interfaces och abstrakta klasser definierar vi vad en viss klass kan göra.

NU: MER ÖVNINGAR MED CLASSES

IMORGON: SQL INTRO