PHP Classes and Objects

Chapter 13

Randy Connolly and Ricardo Hoar

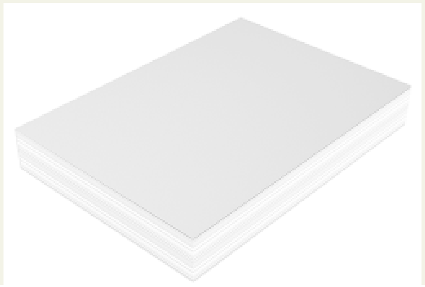
Fundamentals of Web Development

Chapter 13

Chapter 13

Object-Oriented Design

High level



Book class

Defines properties such as:
title, author, and number of pages

Objects (or instances of the Book class)

Each instance has its own title, author, and number of pages property values

Object-Oriented Design

Terminology

The notion of programming with **objects** allows the developer to think about an item with particular

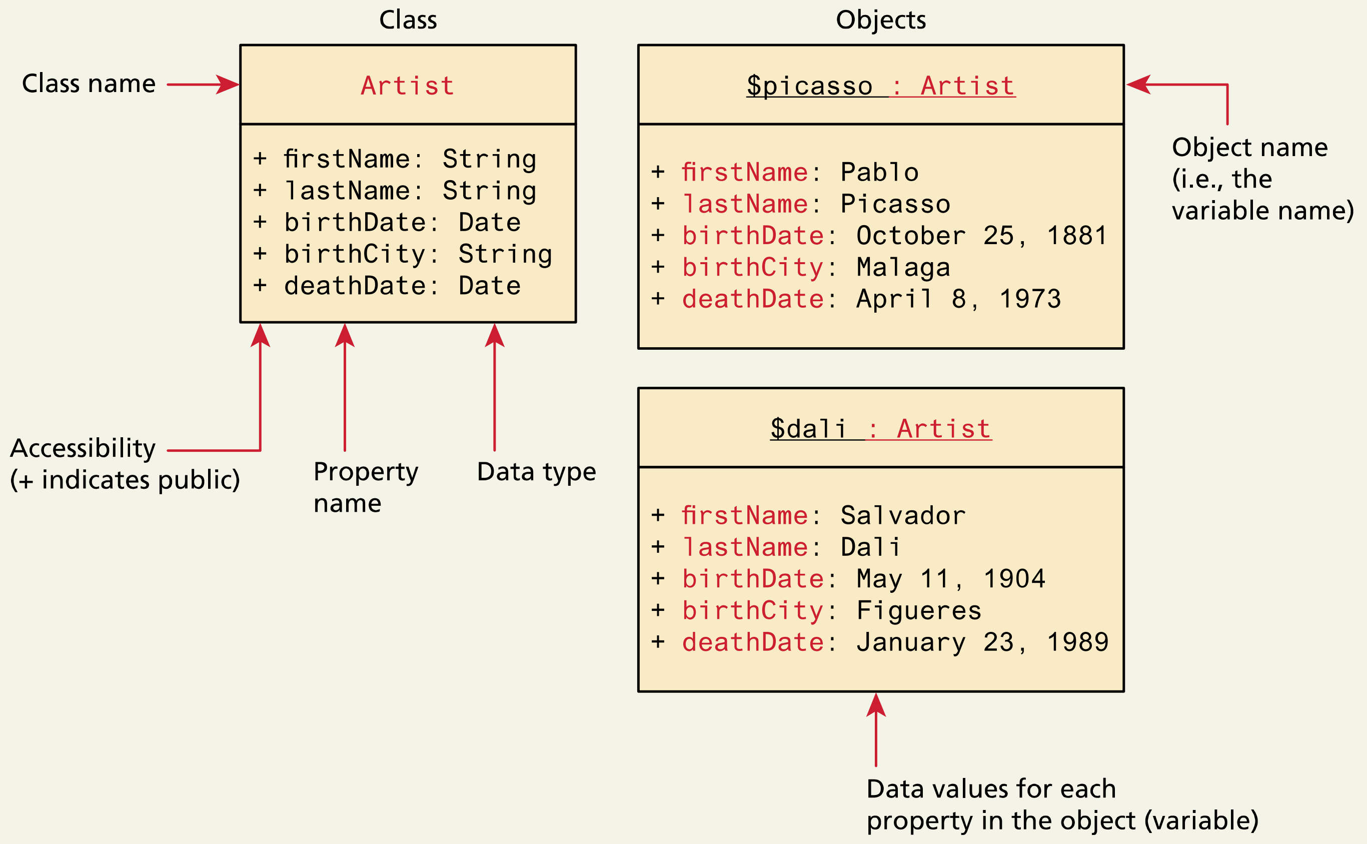
- **properties** (also called **attributes** or data members) and
- **methods** (**functions**).

The structure of these object is defined by **classes** , which outline the properties and methods like a blueprint.

Each variable created from a class is called an **object** or **instance**

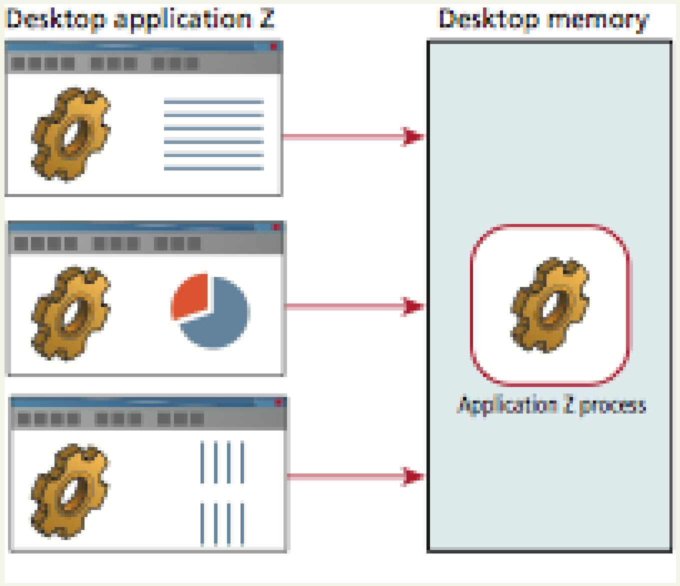
Object-Oriented Design

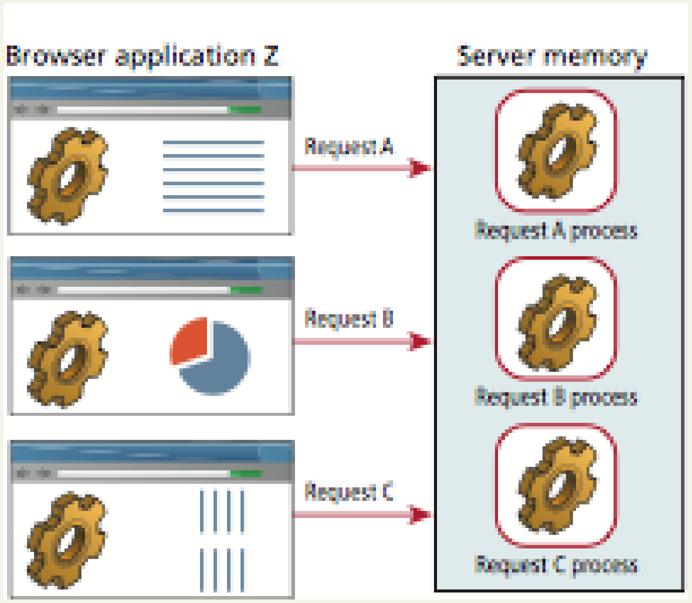
The Unified Modeling Language



Object-Oriented Design

Differences between Server and Desktop Objects





Chapter 13

Classes and Objects in PHP

Defining Classes

```
class Artist {
        public $firstName;
        public $lastName;
        public $birthDate;
        public $birthCity;
        public $deathDate;
}
```

Classes and Objects in PHP

Instantiating Objects

Use the new keyword

```
$picasso = new Artist();

$dali = new Artist();
```

Classes and Objects in PHP

Properties

Once you have instances of an object, you can access and modify the properties using the object's variable name and an arrow (->)

```
$picasso = new Artist();
$dali = new Artist();
$picasso->firstName = "Pablo";
$picasso->lastName = "Picasso";
$picasso->birthCity = "Malaga";
$picasso->birthDate = "October 25 1881";
$picasso->deathDate = "April 8 1973";
```

Classes and Objects in PHP

Constructors

```
class Artist {
  // variables from previous listing still go here
  function __construct($firstName, $lastName, $city, $birth,$death=null) {
          $this->firstName = $firstName;
          $this->lastName = $lastName;
          $this->birthCity = $city;
          $this->birthDate = $birth;
          $this->deathDate = $death;
  }
}
```

Classes and Objects in PHP

Instantiating using constructors

```
$picasso = new Artist("Pablo","Picasso","Malaga","Oct 25,1881","Apr 8,1973");

$dali = new Artist("Salvador","Dali","Figures","May 11 1904", "Jan 23 1989");
```

Classes and Objects in PHP

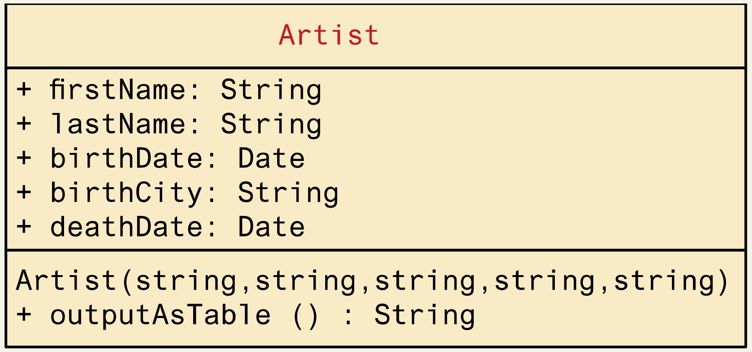
Method

Methods define the tasks each instance of a class can perform and are useful since they associate behavior with objects

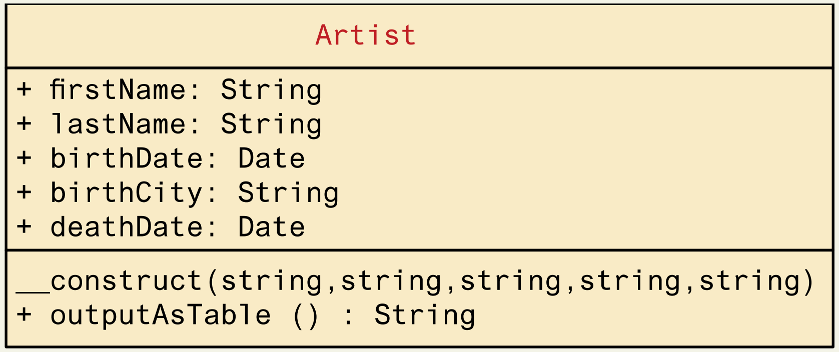
```
class Artist {
        //. . .
        public function outputAsTable() {
        //. . .
        }
}
```

Classes and Objects in PHP

Updated UML (two ways to show constructor)



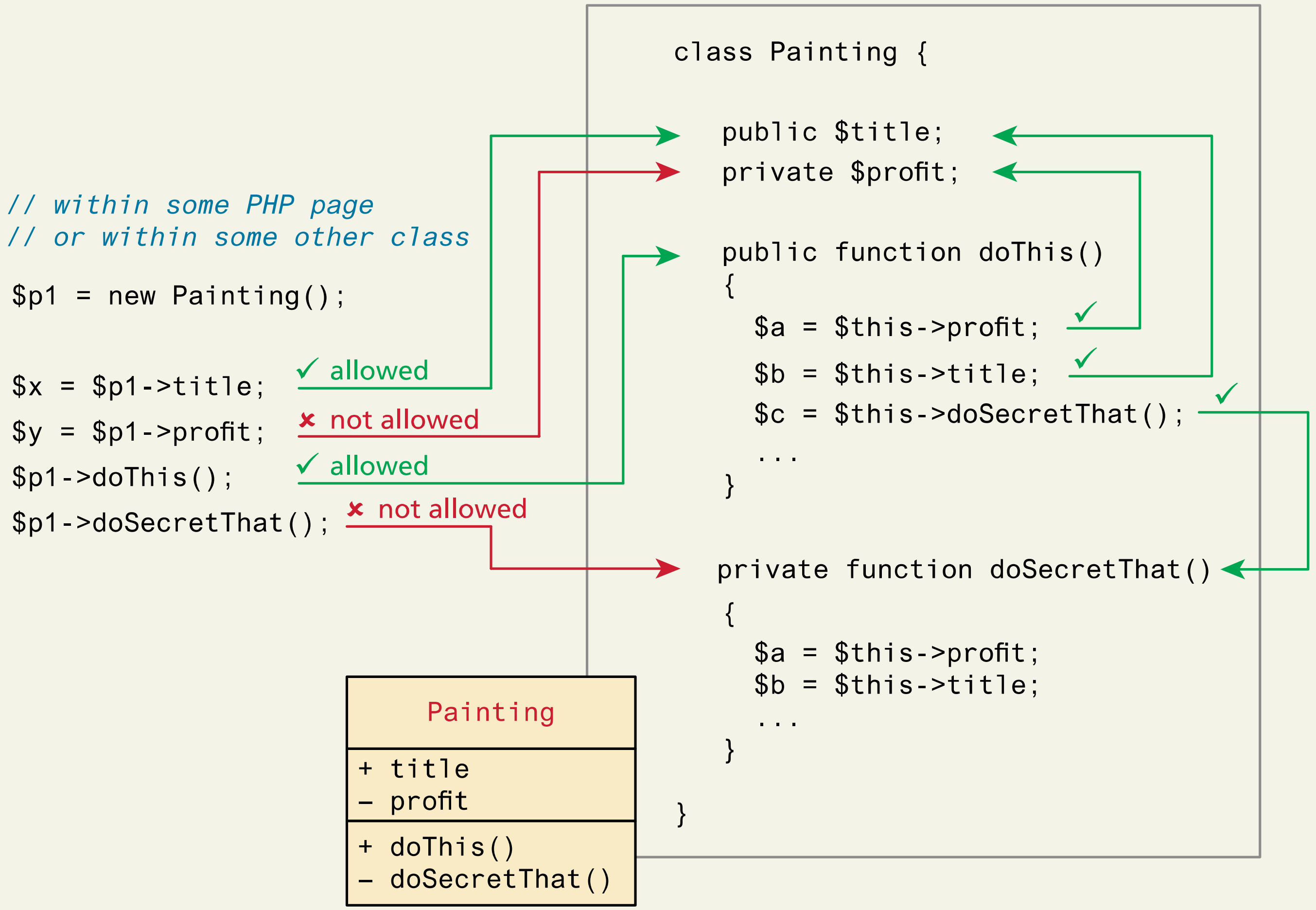
```
Artist
+ firstName: String
+ lastName: String
+ birthDate: Date
+ birthCity: String
+ deathDate: Date
Artist(string,string,string,string,string)
+ outputAsTable () : String
```



```
Artist
+ firstName: String
+ lastName: String
+ birthDate: Date
+ birthCity: String
+ deathDate: Date
__construct(string,string,string,string,string)
+ outputAsTable () : String
```

Classes and Objects in PHP

Visibility



```
// within some PHP page
// or within some other class

$p1 = new Painting();

$x = $p1->title;
$y = $p1->profit;
$p1->doThis();
$p1->doSecretThat();
```

```
class Painting {

    public $title;
    private $profit;

    public function doThis()
    {
      $a = $this->profit;
      $b = $this->title;
      $c = $this->doSecretThat();
      ...
    }

    private function doSecretThat()
    {
      $a = $this->profit;
      $b = $this->title;
      ...
    }

}
```

Painting
+ title – profit
+ doThis() – doSecretThat()

Classes and Objects in PHP

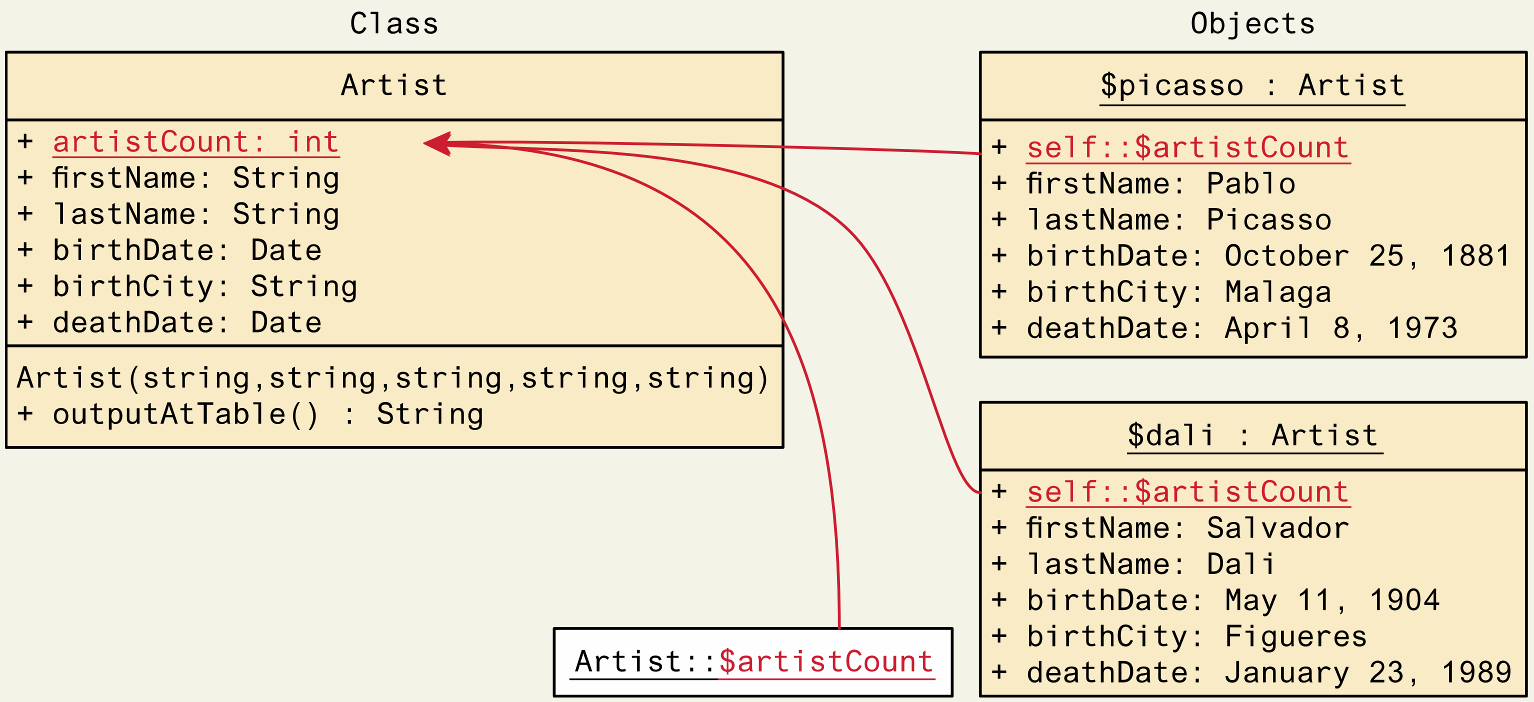
Static Members

A **static member** is a property or method that all instances of a class share.

Unlike an instance property, where each object gets its own value for that property, there is only one value for a class's static property, shared across all instances.

Classes and Objects in PHP

Static Members



Classes and Objects in PHP

Class Constants

constant values can be stored more efficiently as class constants so long as they are not calculated or updated

const EARLIEST_DATE = 'January 1, 1200';

They can be accessed both inside and outside the class using self::EARLIEST_DATE in the class and classReference::EARLIEST_DATE

Chapter 13

Object-Oriented Design

Data Encapsulation

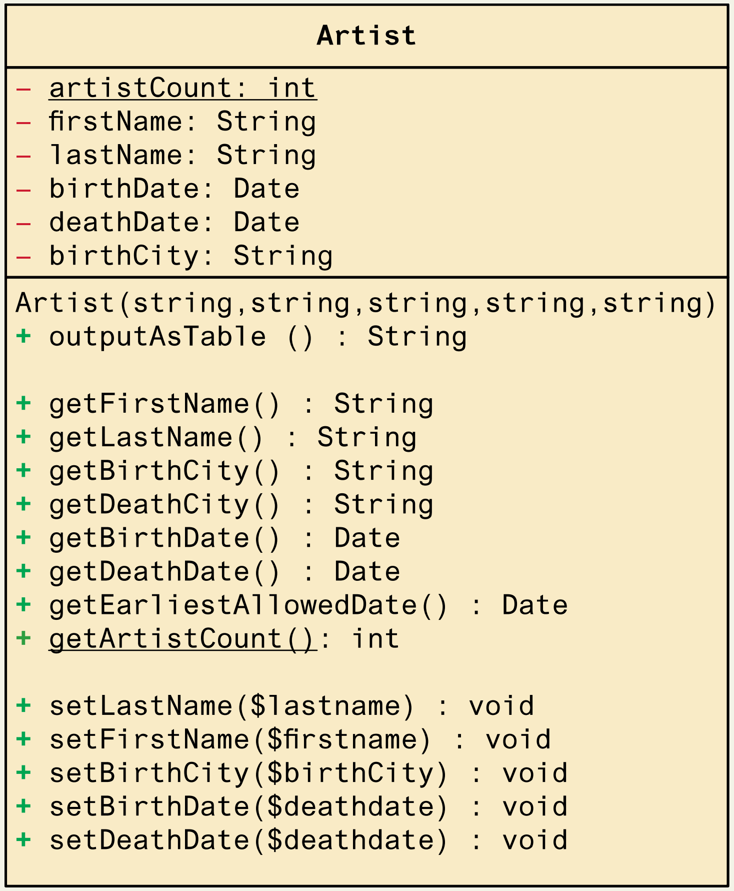
Encapsulation generally refers to restricting access to an object's internal components. Another way of understanding encapsulation is: it is the hiding of an object's implementation details.

A properly encapsulated class will define an interface to the world in the form of its public methods, and leave its data, that is, its properties, hidden (i.e., private). This allows the class to control exactly how its data will be used.

The typical approach is to write methods for accessing and modifying properties rather than allowing them to be accessed directly. These methods are commonly called **getters and setters** (or accessors and mutators).

Object-Oriented Design

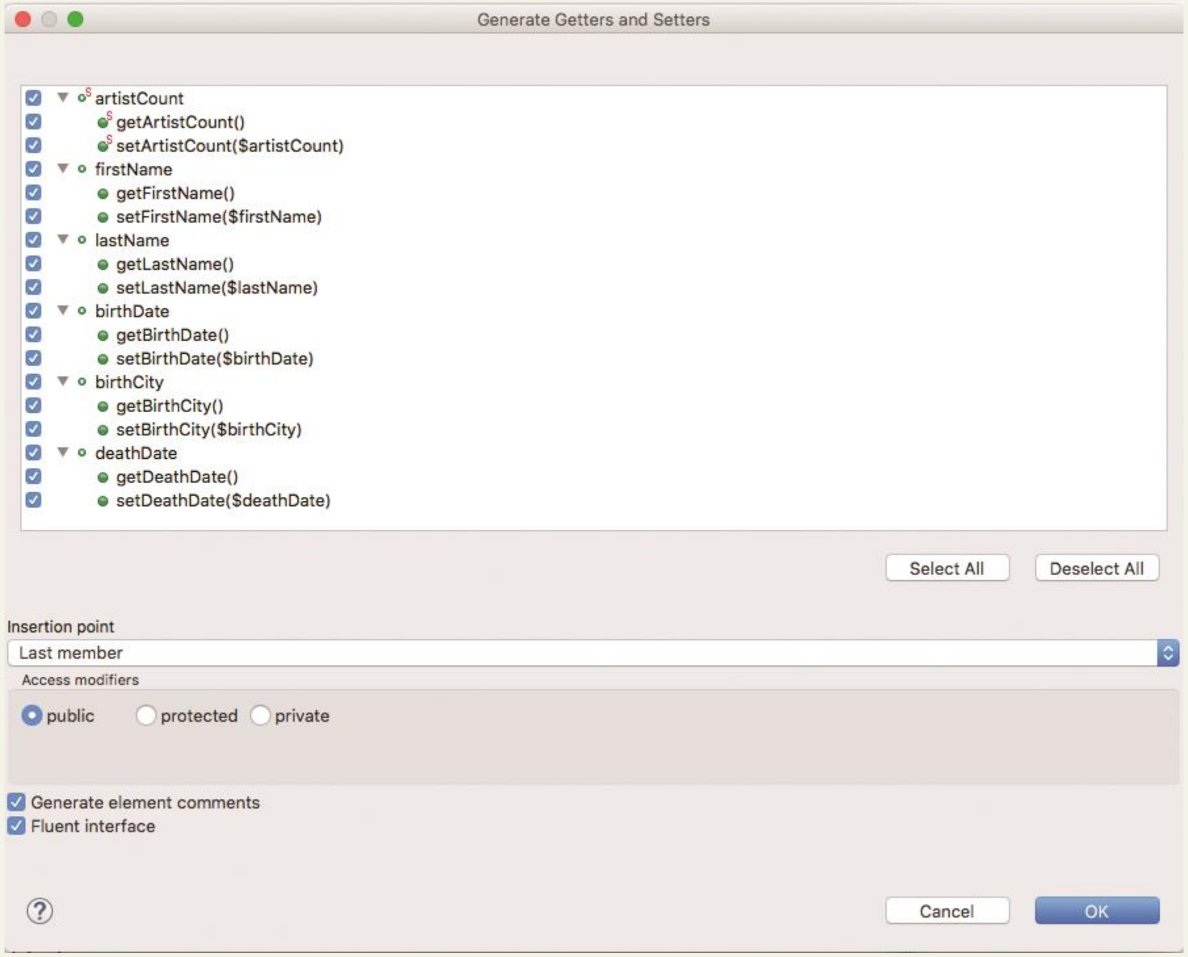
Data Encapsulation



```
Artist
- artistCount: int
- firstName: String
- lastName: String
- birthDate: Date
- deathDate: Date
- birthCity: String

Artist(string,string,string,string,string)
+ outputAsTable () : String

+ getFirstName() : String
+ getLastName() : String
+ getBirthCity() : String
+ getDeathCity() : String
+ getBirthDate() : Date
+ getDeathDate() : Date
+ getEarliestAllowedDate() : Date
+ getArtistCount(): int

+ setLastName($lastname) : void
+ setFirstName($firstname) : void
+ setBirthCity($birthCity) : void
+ setBirthDate($deathdate) : void
+ setDeathDate($deathdate) : void
```

Object-Oriented Design

Generating Getters and Setters through an IDE



Generate Getters and Setters

```
☑ ▼ artistCount
☑     getArtistCount()
☑     setArtistCount($artistCount)
☑ ▼ firstName
☑     getFirstName()
☑     setFirstName($firstName)
☑ ▼ lastName
☑     getLastName()
☑     setLastName($lastName)
☑ ▼ birthDate
☑     getBirthDate()
☑     setBirthDate($birthDate)
☑ ▼ birthCity
☑     getBirthCity()
☑     setBirthCity($birthCity)
☑ ▼ deathDate
☑     getDeathDate()
☑     setDeathDate($deathDate)
```

Insertion point: Last member

Access modifiers: [✓ public] protected private

☑ Generate element comments

☑ Fluent interface

Object-Oriented Design

Inheritance

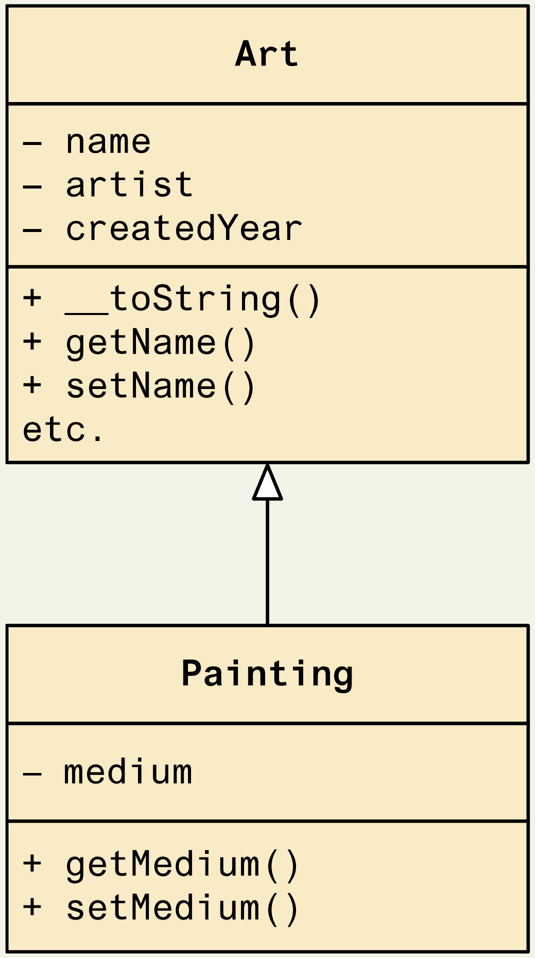
Inheritance enables you to create new PHP classes that reuse, extend, and modify the behavior that is defined in another PHP class.

A class that is inheriting from another class is said to be a **subclass** or a **derived class** . The class that is being inherited from is typically called a **superclass** or a **base class**.

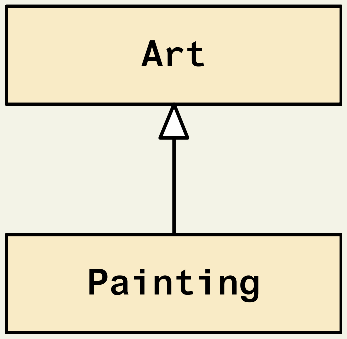
When a class inherits from another class, it inherits all of its public and protected methods and properties.

Object-Oriented Design

Inheritance

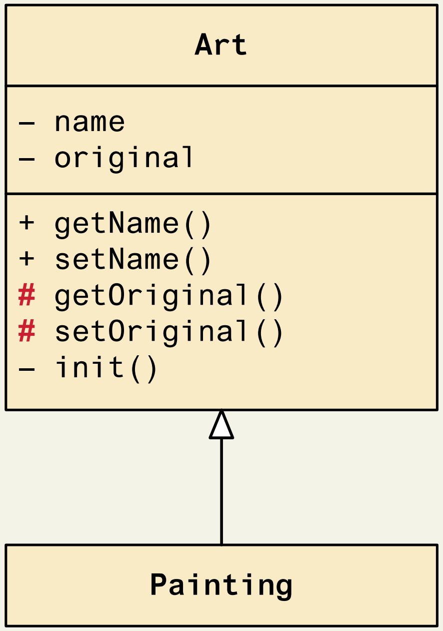


class Painting **extends** Art { . . . }



Object-Oriented Design

Inheritance



```
class Painting extends Art {
   …
   private function foo() {
     …
     // these are allowed
✓    $w = parent::getName();
✓    $x = parent::getOriginal();

     // this is not allowed
✗    $y = parent::init();
   }
}
```

```
  // in some page or other class
  $p = new Painting();
  $a = new Art();

  // neither of these references are allowed
✗ $w = $p->getOriginal();
✗ $y = $a->getOriginal();
```

Object-Oriented Design

Polymorphism

Polymorphism is the notion that an object can in fact be multiple things at the same time.

Conceptually, a sculpture is a work of art and a painting is a work of art.

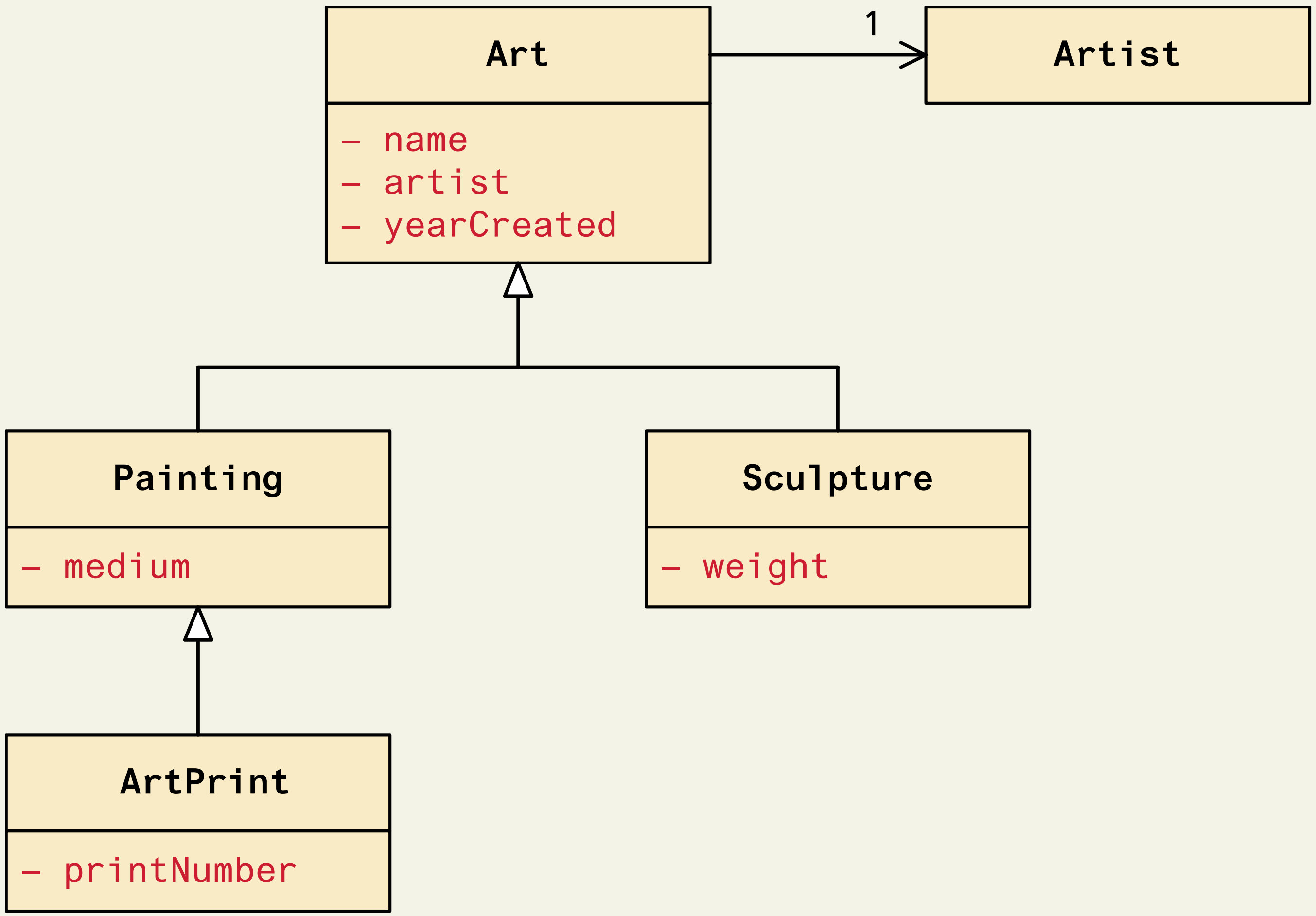
$guernica = new Painting("1937",$picasso,"Guernica","Oil on canvas");

The variable $guernica is both a Painting object and an Art object due to its inheritance.

We can manage a list of Art objects, and call the same (overridden) method on each

Object-Oriented Design

Polymorphism



Object-Oriented Design

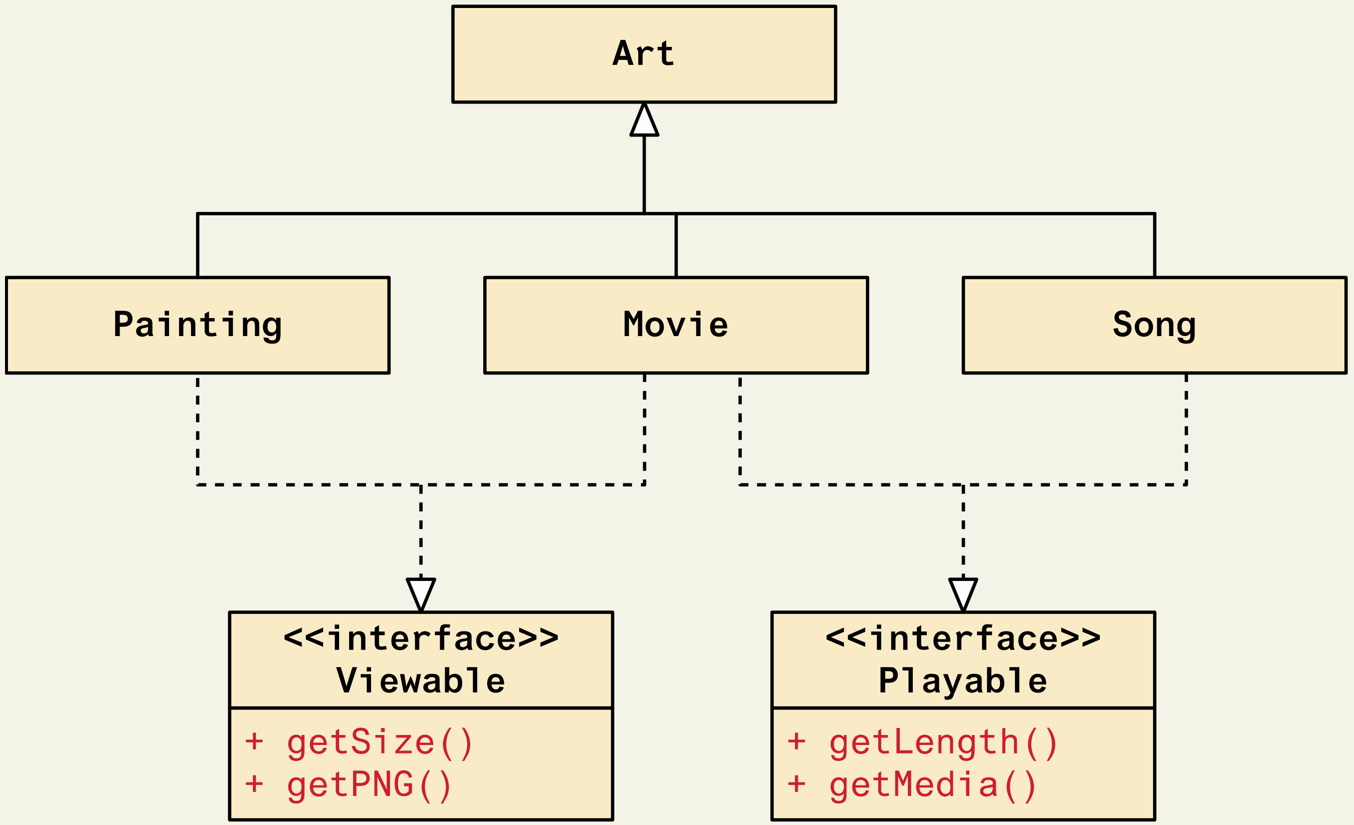
Object Interfaces

An object **interface** is a way of defining a formal list of methods that a class must implement without specifying their implementation.

```
interface Viewable {
        public function getSize();
        public function getPNG();
}

class Painting extends Art implements Viewable { . . . }
```

Object-Oriented Design

Interfaces in class diagram



Chapter 13

1 Object-Oriented Overview

2 Classes and Objects in PHP

3 Object-Oriented Design

4 Summary

Summary

Key Terms

base class
class
class member
constructor
data members
derived class
dynamic dispatching
encapsulation
getters and setters
inheritance
instance
instantiate
Integrated Development Environment (IDE)
interface
magic methods
methods
naming conventions
objects
polymorphism
properties
refactoring
skeleton
static
subclass
superclass
UML (Unified Modeling Language)
visibility

Summary

Questions?