

Pertemuan 4: Inheritance (Pewarisan)

Tujuan Pembelajaran

Setelah mengikuti pertemuan ini, mahasiswa diharapkan dapat:

1. Memahami konsep inheritance dalam OOP
2. Mengimplementasikan inheritance dengan keyword `extends`
3. Memahami konsep parent class dan child class
4. Menggunakan keyword `parent::` untuk mengakses parent methods
5. Memahami method overriding dan method overloading
6. Menerapkan multilevel inheritance
7. Memahami prinsip "is-a" relationship

Konsep Inheritance

Definisi Inheritance

Inheritance (pewarisan) adalah mekanisme dalam OOP yang memungkinkan sebuah class (child class) untuk mewarisi properti dan method dari class lain (parent class). Ini memungkinkan kode reuse dan membuat hierarki class yang terorganisir.

Keuntungan Inheritance

1. **Code Reusability** - Menghindari duplikasi kode
2. **Maintainability** - Perubahan di parent class otomatis berlaku untuk child class
3. **Extensibility** - Mudah menambahkan fitur baru di child class
4. **Polymorphism** - Child class dapat diperlakukan sebagai parent class
5. **Organization** - Membuat struktur class yang hierarkis dan logis

Sintaks Dasar

```
class ParentClass {  
    // Properties dan methods parent  
}  
  
class ChildClass extends ParentClass {  
    // Properties dan methods tambahan  
    // Dapat override parent methods  
}
```

Jenis-jenis Inheritance

1. Single Inheritance

PHP hanya mendukung single inheritance - sebuah class hanya dapat mewarisi dari satu parent class.

```
class Animal {  
    public $name;  
}  
  
class Dog extends Animal {  
    public $breed;  
}
```

2. Multilevel Inheritance

Child class dapat menjadi parent untuk class lainnya.

```
class Animal {  
    // Base class  
}  
  
class Mammal extends Animal {  
    // Child of Animal  
}  
  
class Dog extends Mammal {  
    // Child of Mammal, grandchild of Animal  
}
```

3. Hierarchical Inheritance

Beberapa child class mewarisi dari parent class yang sama.

```
class Animal {  
    // Base class  
}  
  
class Dog extends Animal {  
    // Child 1  
}  
  
class Cat extends Animal {  
    // Child 2  
}
```

Access Modifiers dalam Inheritance

Public

- Dapat diakses dari mana saja
- Diwariskan ke child class

Protected

- Dapat diakses dari class itu sendiri dan child class
- Tidak dapat diakses dari luar class
- Ideal untuk inheritance

Private

- Hanya dapat diakses dari class itu sendiri
- Tidak diwariskan ke child class

```
class Parent {
    public $publicProp = "Public";
    protected $protectedProp = "Protected";
    private $privateProp = "Private";
}

class Child extends Parent {
    public function testAccess() {
        echo $this->publicProp;    // ✓ Bisa diakses
        echo $this->protectedProp; // ✓ Bisa diakses
        echo $this->privateProp;   // ✗ Error!
    }
}
```

Method Overriding

Definisi

Method overriding adalah proses mendefinisikan ulang method parent class di child class dengan implementasi yang berbeda.

Aturan Method Overriding

1. Method name harus sama persis
2. Parameter harus compatible
3. Visibility tidak boleh lebih restrictive
4. Return type harus compatible (PHP 7.4+)

```
class Animal {
    public function makeSound() {
        return "Some sound";
    }
}

class Dog extends Animal {
    public function makeSound() {
        return "Woof!"; // Override parent method
    }
}
```

```
}  
}
```

Keyword `parent::`

Mengakses Parent Methods

```
class Child extends Parent {  
    public function someMethod() {  
        parent::someMethod(); // Panggil parent method  
        // Kode tambahan  
    }  
}
```

Mengakses Parent Constructor

```
class Child extends Parent {  
    public function __construct($param1, $param2) {  
        parent::__construct($param1); // Panggil parent constructor  
        // Inisialisasi child properties  
    }  
}
```

Abstract Classes dan Methods

Abstract Class

```
abstract class Animal {  
    abstract public function makeSound();  
  
    public function sleep() {  
        return "Sleeping...";  
    }  
}  
  
class Dog extends Animal {  
    public function makeSound() {  
        return "Woof!";  
    }  
}
```

Final Classes dan Methods

Final Class

```
final class Math {  
    // Class ini tidak bisa diwariskan  
}
```

Final Method

```
class Parent {  
    final public function importantMethod() {  
        // Method ini tidak bisa di-override  
    }  
}
```

Instanceof Operator

Untuk mengecek apakah object adalah instance dari class tertentu:

```
$dog = new Dog();  
if ($dog instanceof Animal) {  
    echo "Dog is an Animal";  
}
```

Best Practices

1. Favor Composition Over Inheritance

```
// Instead of inheritance  
class FlyingCar extends Car {  
    // Multiple inheritance problem  
}  
  
// Use composition  
class Car {  
    private $flyingCapability;  
  
    public function __construct(FlyingCapability $flying = null) {  
        $this->flyingCapability = $flying;  
    }  
}
```

2. Use Protected for Inheritance

```
class Parent {  
    protected $inheritableProperty; // Good for inheritance
```

```
    private $internalProperty;    // Keep internal
}
```

3. Document Inheritance Relationships

```
/**
 * Base class for all vehicles
 */
abstract class Vehicle {
    // ...
}

/**
 * Represents a car vehicle
 * @extends Vehicle
 */
class Car extends Vehicle {
    // ...
}
```

Liskov Substitution Principle (LSP)

Child class harus dapat menggantikan parent class tanpa merusak fungsionalitas program.

```
class Bird {
    public function fly() {
        return "Flying";
    }
}

class Penguin extends Bird {
    public function fly() {
        throw new Exception("Penguins can't fly!"); // LSP violation
    }
}
```

Method Resolution Order

PHP mencari method dengan urutan:

1. Current class
2. Parent class
3. Grandparent class
4. Dan seterusnya ke atas

Contoh Implementasi

Lihat file `example.php` untuk berbagai contoh implementasi inheritance di PHP.

Latihan

1. Buat hierarki class untuk sistem perpustakaan:
 - Base class: `Item` (judul, pengarang, tahun)
 - Child classes: `Book`, `Magazine`, `DVD`
2. Implementasikan method overriding untuk method `getInfo()`
3. Gunakan protected properties yang dapat diakses child classes
4. Buat method yang menggunakan `parent::` untuk memanggil parent method

Tugas Rumah

Buat sistem manajemen karyawan dengan inheritance:

- Base class: `Employee` (nama, id, gaji)
- Child classes: `Manager`, `Developer`, `Designer`
- Setiap child class memiliki method khusus dan override method `calculateBonus()`
- Implementasikan multilevel inheritance: `SeniorDeveloper extends Developer`
- Gunakan abstract methods untuk memaksa implementasi di child classes