Repaso de UML

Diseño de Sistemas Software
Curso 2017/2018

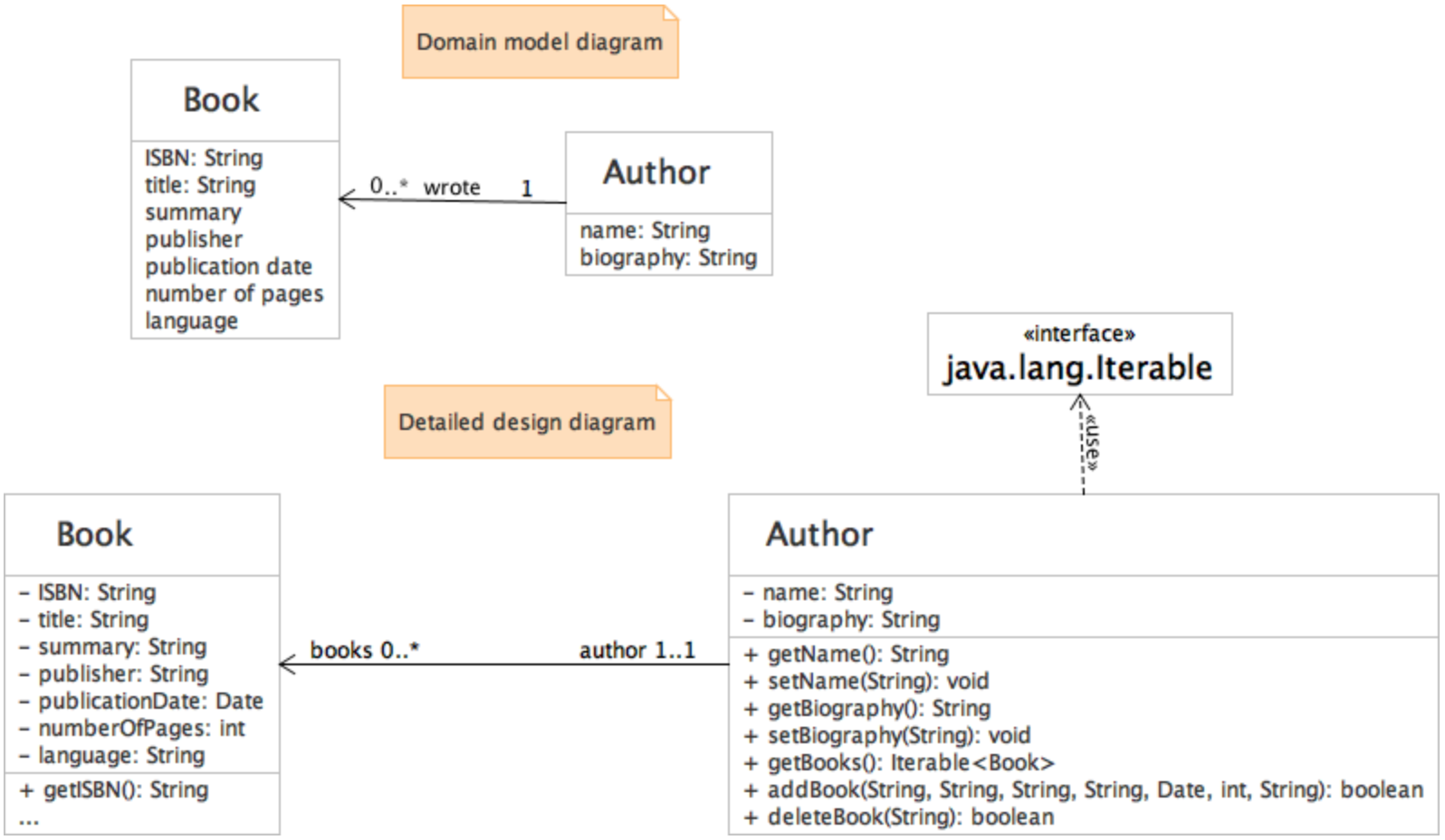
Carlos Pérez Sancho

Universitat d'Alacant
Universidad de Alicante
Departament de Llenguatges i Sistemes Informàtics
Departamento de Lenguajes y Sistemas Informáticos

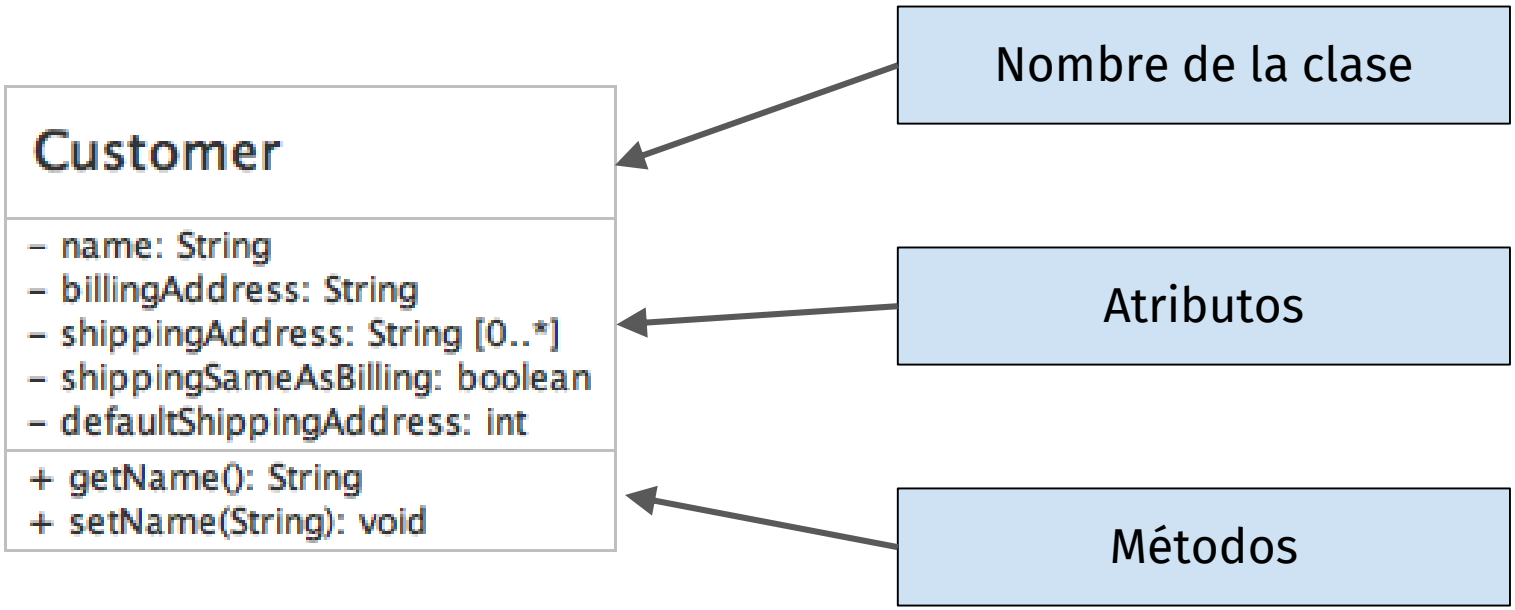
Diagramas de clases

Conceptos básicos

Modelo de dominio vs. Diseño detallado



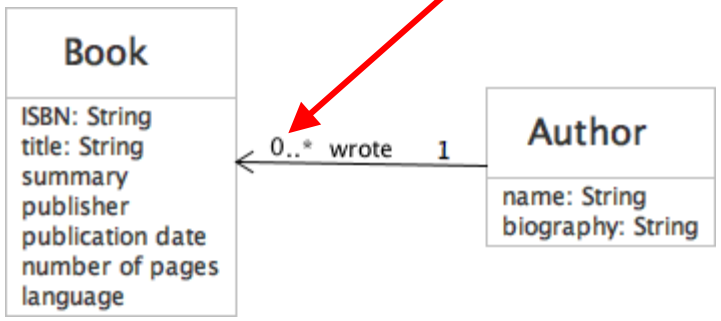
Clases


Customer
- name: String
- billingAddress: String
- shippingAddress: String [0..*]
- shippingSameAsBilling: boolean
- defaultShippingAddress: int
+ getName(): String
+ setName(String): void
Nombre de la clase
Atributos
Métodos

Atributos y métodos

- Visibilidad:
  - + pública
  - - privada
- Multiplicidad
  - 1 obligatorio (multiplicidad por defecto si no se indica)
  - 0..1 opcional
  - 0..* opcional, puede tener múltiples valores
  - 1..* obligatorio, puede tener múltiples valores

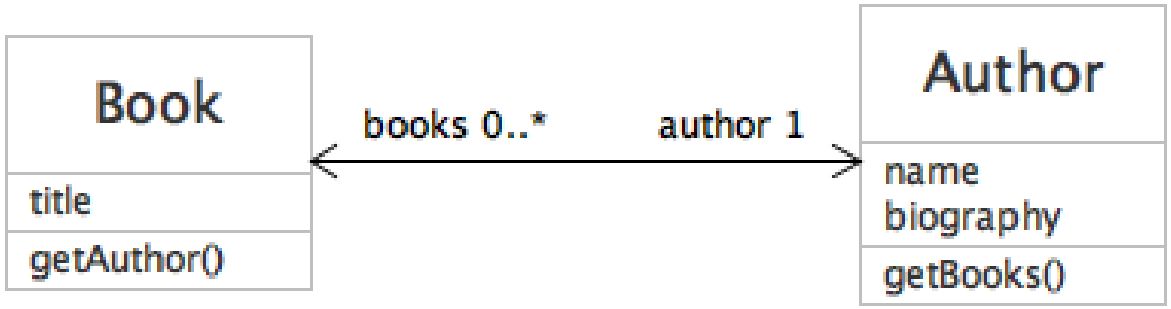
Propiedades



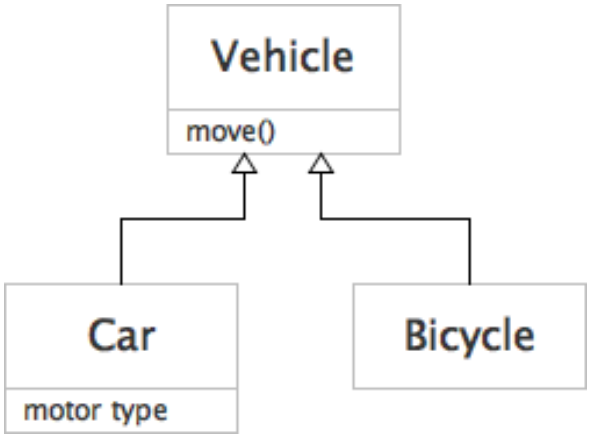

```php
class Author {
    private $name;
    private $biography;
    private $books = array();

    function __construct($name) {
        $this->name = $name;
    }
    public function addBook($book) {
        $this->books[] = $book;
    }
}


class Book {
    private $title;
        // ...

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

$author = new Author("Martin Fowler");
$book = new Book("UML Distilled");
$author->addBook($book);
```

Asociaciones bidireccionales

- En una asociación bidireccional se puede navegar en los dos sentidos

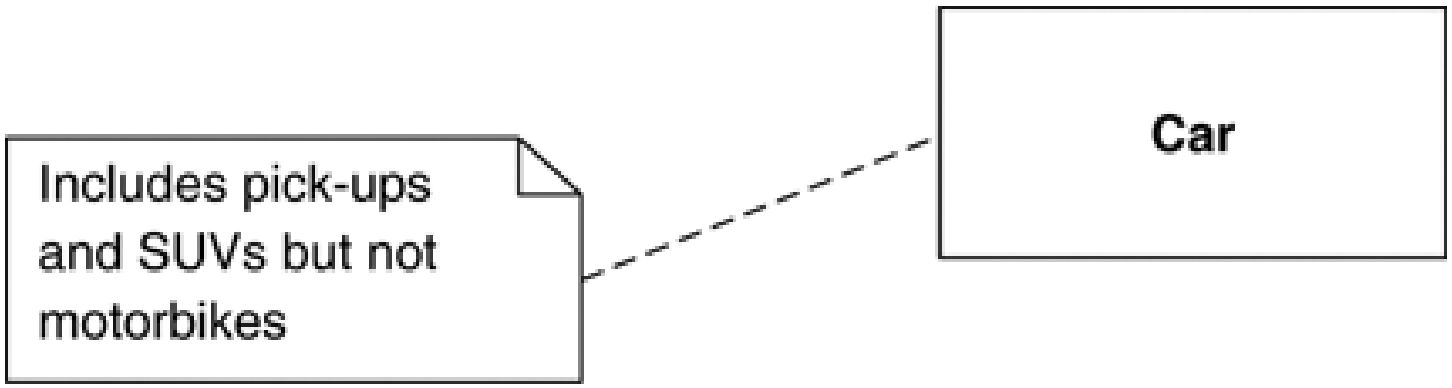


Herencia

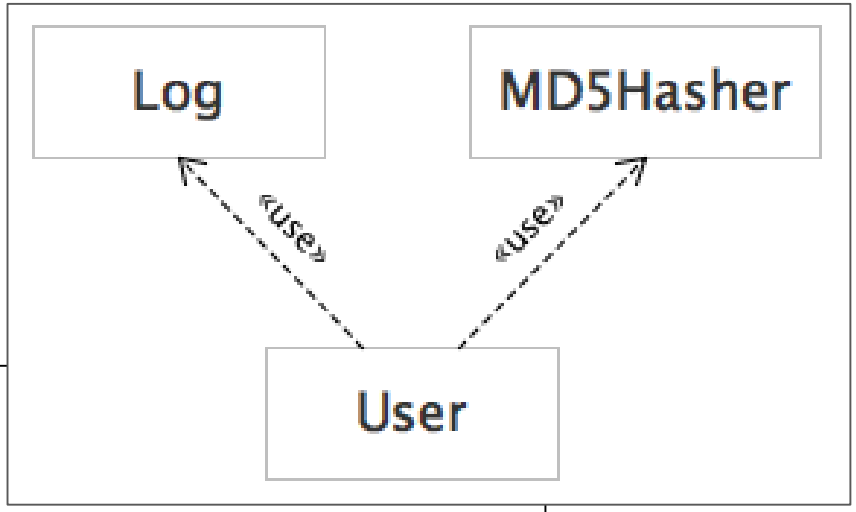



```php
class Vehicle {
    public function move() {
        echo "I just moved!";
    }
}

class Car extends Vehicle {
    private $motorType;
}

class Bicycle extends Vehicle {

}

$car = new Car();
$car->move();
$bike = new Bicycle();
$bike->move();
```

Notas



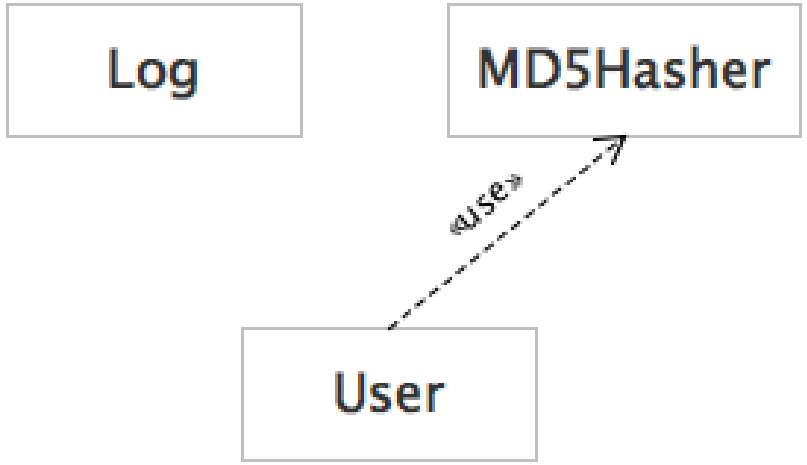
Dependencias




```php
use Illuminate\Support\Facades\Log;

class User {
    public function login($name, $pass) {
        Log::debug("Checking $name and $pass");
        $hasher = new MD5Hasher();
        $hash = $hasher->hash($pass);
        // …
    }
}

$user = new User();
$user->login("username", "strongpassword");
```

Dependencias

- No se deben mostrar todas las dependencias en los diagramas, hay demasiadas y cambian frecuentemente
- Sólo se deben mostrar las que sean relevantes para el problema



Ejercicio

Código ↔ Diseño

- Para el ejemplo de código de la siguiente página:
- Dibuja el diagrama de clases correspondiente al código
- Modifica el diseño realizando los siguientes cambios
  - Crea una clase `BubbleSort`, con un método `sort($array)`
  - Crea una clase `Util`, con un método `swap(&$a, &$b)`
  - Elimina los métodos `sort` y `swap` de la clase `Professor`
  - Añade las relaciones necesarias
- Escribe el código correspondiente al nuevo diseño

```php
class Professor {
    private $subject; // Gives access to student's marks

    private function swap(&$a, &$b) {
        $c = $a; $a = $b; $b = $c;
    }

    private function bubbleSort($array) {
        // does stuff… calls swap()
        return $sortedArray;
    }

    private function getTopStudents($marks) {
        // $students is an associative array [ '00000001A' => 8.7 ]
        $sorted = $this->bubbleSort($marks);
        $top = array_slice($sorted, 0, 10);
        return array_keys($top); // returns top-10 students
    }

    public function givePrize() {
        $top = $getTopStudents($this->subject->getStudentMarks());
        foreach ($top as $dni => $mark) {
            $this->subject->setMark($dni, $mark + 1.0);
        }
    }
}
```

Conceptos avanzados

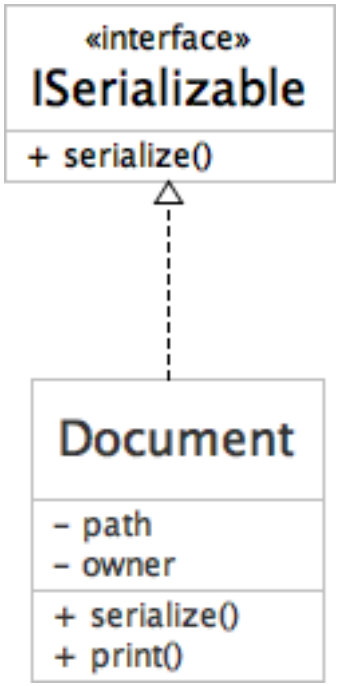
Diagramas de clases

Estereotipos e interfaces

Los estereotipos permiten añadir información a una clase

Notación: <<nombre_estereotipo>>

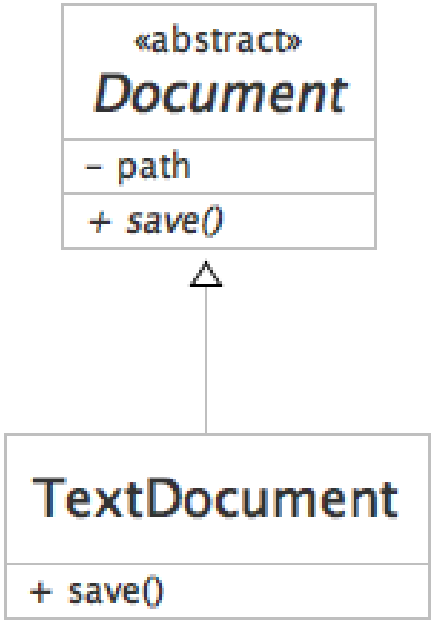
Un ejemplo de estereotipo son los **interfaces**



Clases abstractas

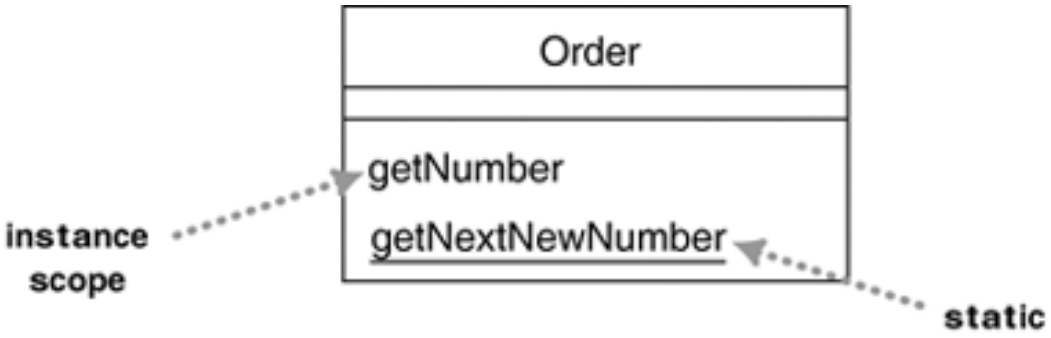
Las clases abstractas contienen métodos abstractos, no se pueden instanciar

Se indican en letra *cursiva*



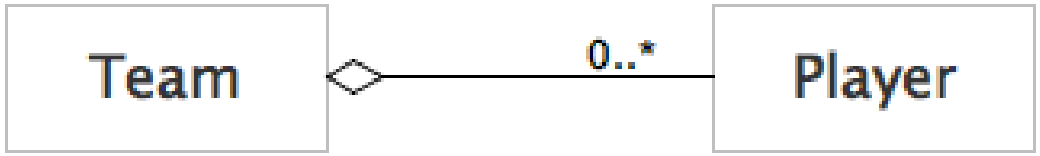
Métodos y atributos estáticos

Se indican con el nombre del método o atributo subrayado



Agregación

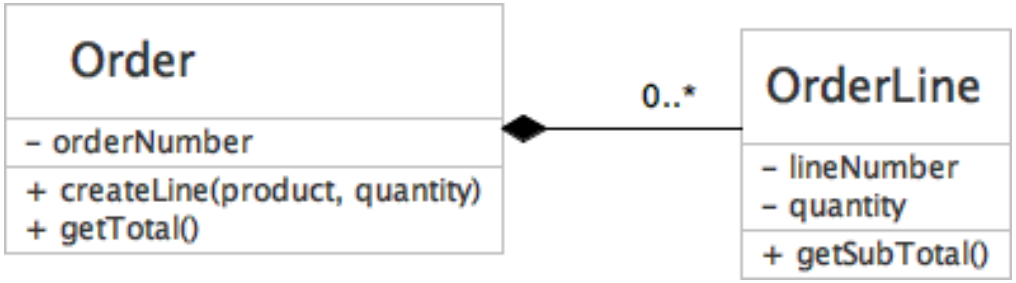
Agregación: las instancias de los objetos agregados pueden ser compartidas con otros objetos




```php
class Team {
        private $players = array();

        public function addPlayer($player) {
                $this->players[] = $player;
        }
}
```

Composición

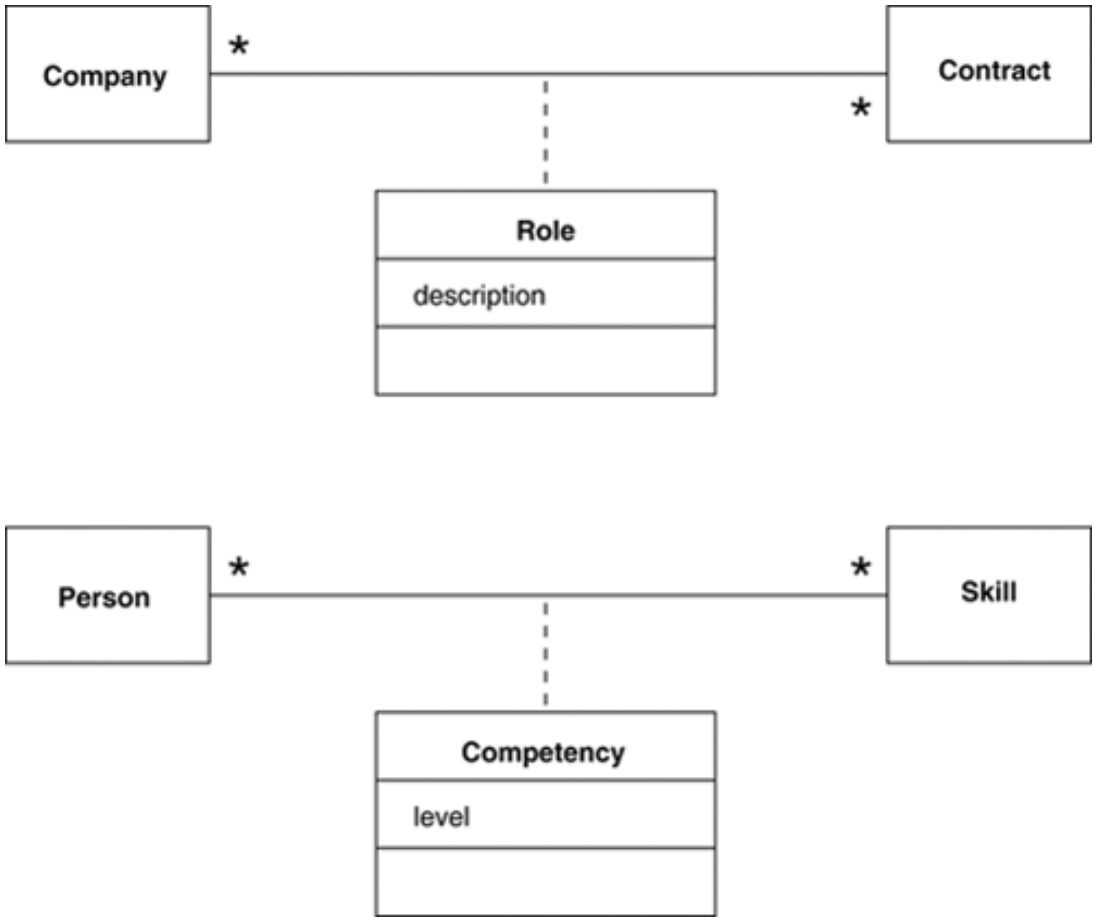
Composición: los componentes forman parte del objeto compuesto



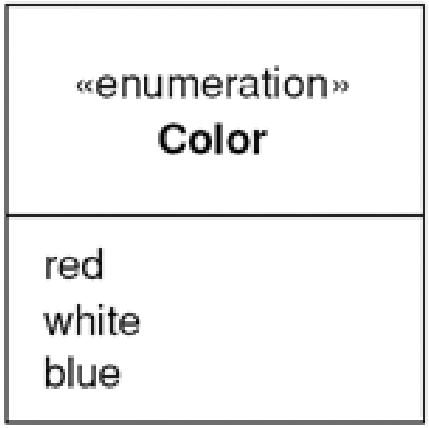

```php
class Order {
    private $lines = array();

    public function createLine($product, $quantity) {
        $this->lines[] = new OrderLine($product, $quantity);
    }
}
```

Clases de asociación

¿Cuál de las dos puede dar problemas?



Enumeraciones



Ejercicios

Diagramas de clases

Widgets 1.0

Supongamos que queremos implementar un paquete de componentes para aplicaciones de entorno gráfico.

Necesitamos los siguientes objetos para componer las pantallas:

- Etiqueta: para mostrar pequeños textos explicativos
- Botón: para que el usuario pueda iniciar acciones
- Checkbox: para que el usuario pueda activar/desactivar opciones

Todos los componentes tienen unas propiedades comunes, como la posición en pantalla, y métodos para mostrarlos y ocultarlos.

Dibuja un diagrama de clases para este problema

Widgets 1.1

Tenemos que añadir dos nuevos componentes para ampliar las funcionalidades de nuestro software:

- Cuadro de texto: permite al usuario escribir texto para luego procesarlo
- Visor HTML: muestra páginas web

Ya contamos con una implementación del visor HTML en una clase llamada `HTMLRenderer` desarrollada por una empresa externa, pero desgraciadamente la interfaz que ofrece no es compatible con nuestro sistema.

Widgets 1.2

Para hacer nuestro sistema un poco más atractivo, nos han pedido cuadros de texto con borde, para que el usuario distinga fácilmente los campos que son opcionales de los obligatorios en los formularios para introducir datos.

Widgets 1.3

Nuestra última actualización ha gustado mucho, así que ahora nos han pedido que todos los componentes puedan tener borde, igual que los cuadros de texto.

¿Preguntas?

Bibliografía

- Fowler, M. (2003). ***UML Distilled, 3rd Edition***. Addison-Wesley Professional. Leer en Safari Books Online