

Universidad Pública de El Alto - Bolivia
Carrera de Ingeniería de Sistemas



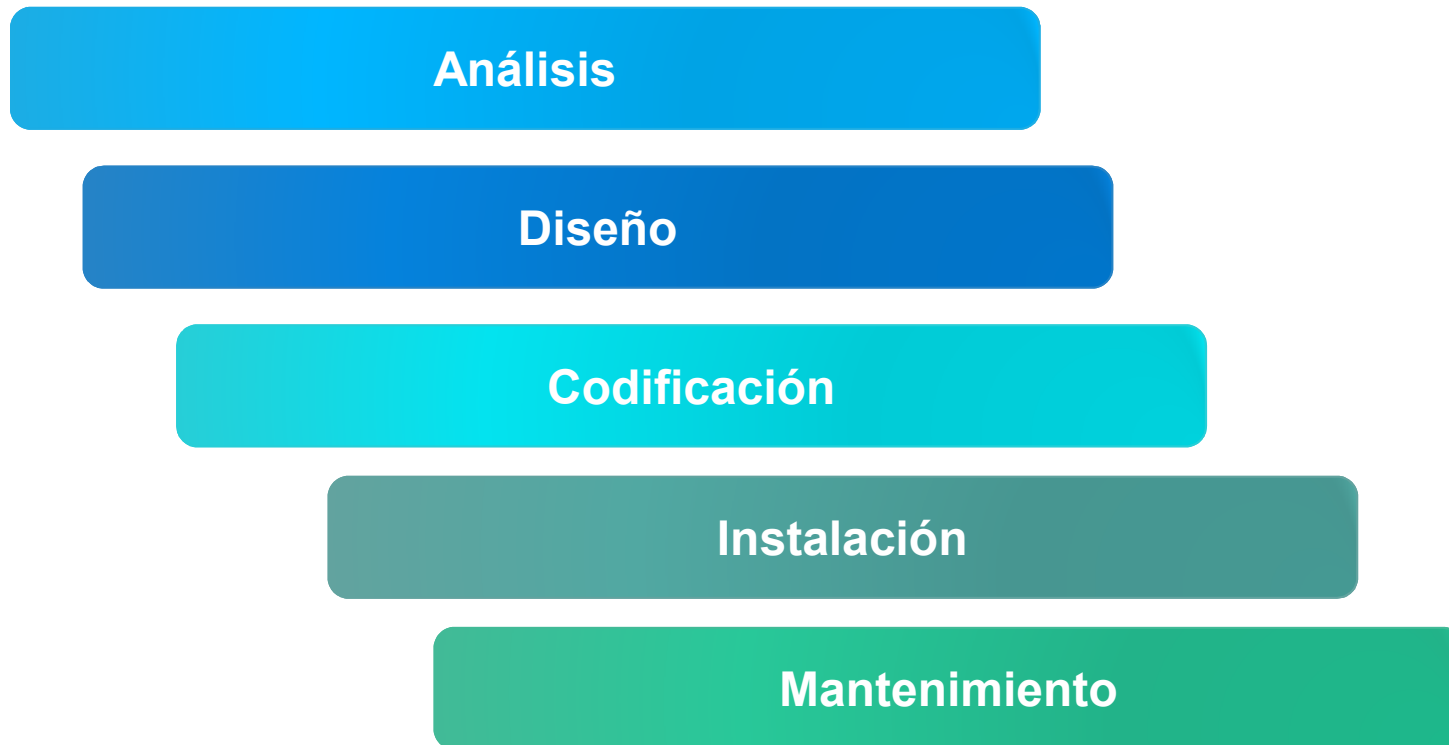
FUNDAMENTOS DE PROGRAMACION ORIENTADA A OBJETO

Lic. Mario Torrez

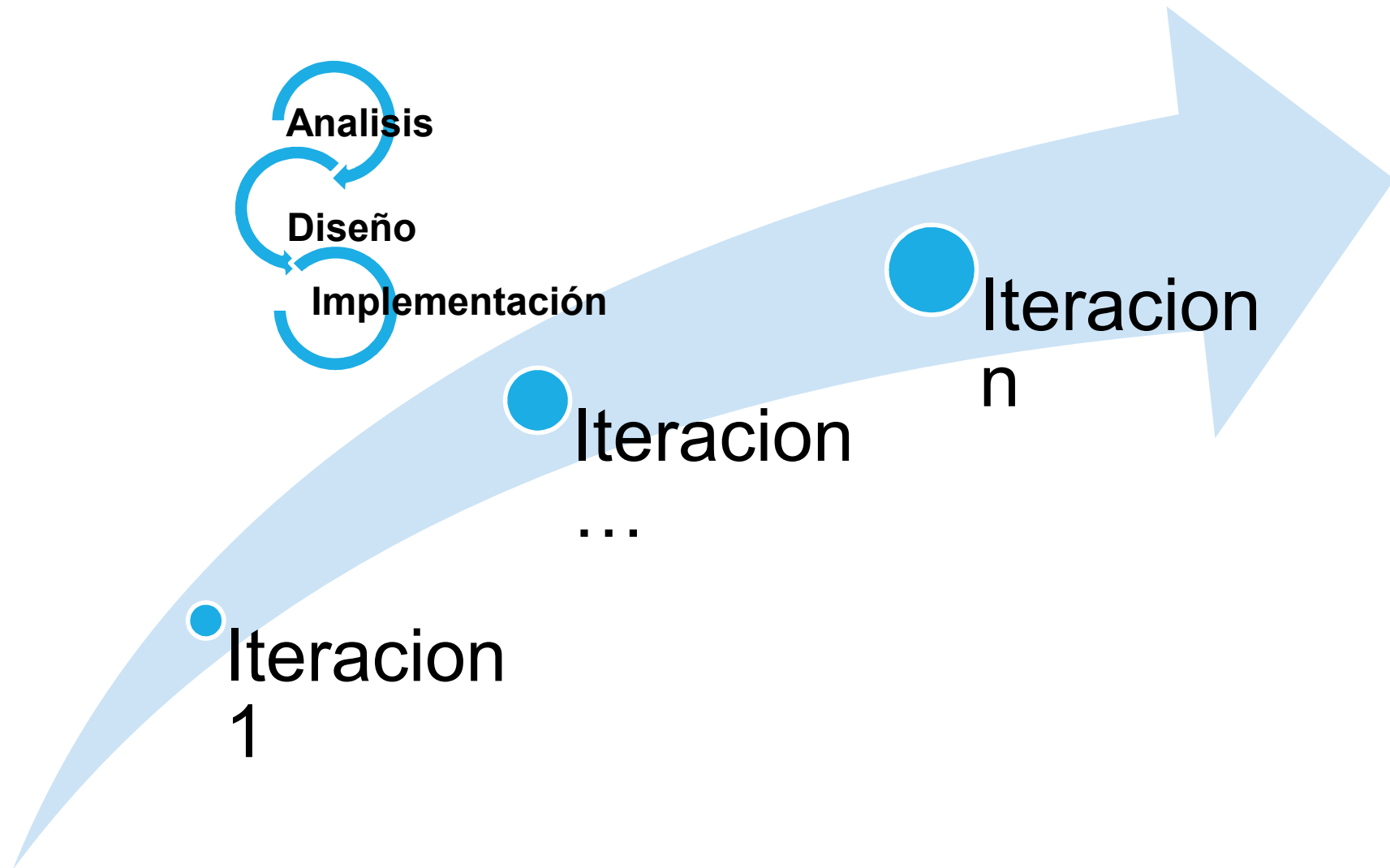
Metodologías de desarrollo de software

- SCRUM
- Programación extrema (XP)
- SSADM
- Proceso Unificado de Rational
- Agile Unified Process
- Adaptive Software Development
- Crystal clear
- Rapid Application Development

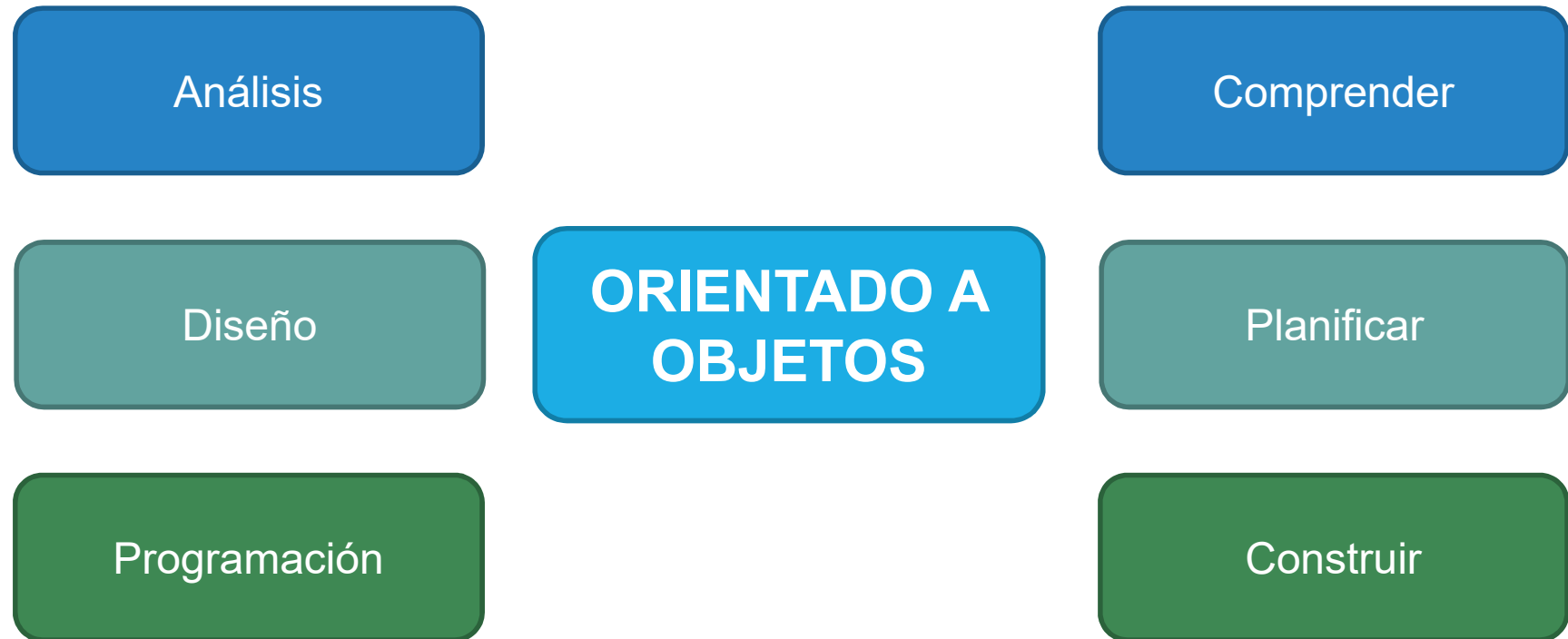
Modelo en cascada




Enfoque ágil o iterativo



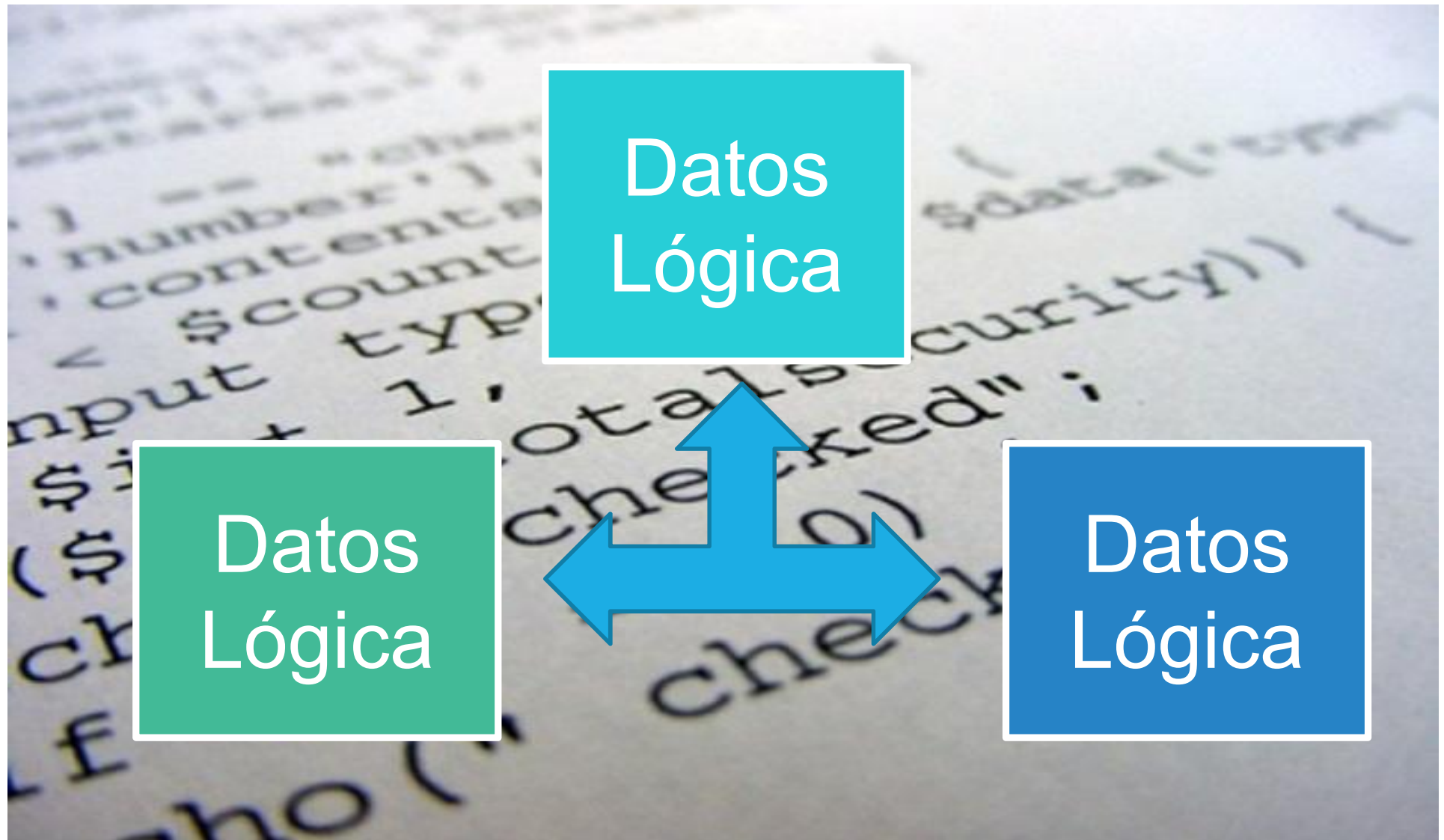
Paradigma OO





Abstracción
Polimorfismo
Herencia
Encapsulacion
Composición
Asociación
Agregamiento
Constructores
Destruyores
Cardinalidad
Singleton

Lenguajes orientado a objeto



Lenguajes de programación orientado a objetos

- C++, C#, Java, JavaScript, Perl, PHP, Python, Ruby





¿ Que es un objeto ?

Objetos de programación

nombre: Juan
edad: 25
genero: masc.

correr()
caminar()

Objeto: cuenta de banco

saldo: Bs. 1000
cuenta: 123456

depositar()
retirar()

Objeto: cuenta de banco

saldo: Bs. 1000
edad: 22
genero: fem.

caminar()
correr()

Objeto: persona



¿ Que es un clase ?

Clase o plano



Objeto / la casa



Clase y múltiples objetos



Creando una clase

nombre

Empleado, CuentaDeBanco, Jugador

atributos

ancho, alto, color, longitud

comportamiento

reproducir(), abrir(), buscar(),
guardar()

Clases y objetos

| CuentaBancaria |
|--|
| NumeroCuenta balance fechaApertura tipoCuenta |
| abrir() cerrar() deposito() retirar() |

clase

crear objetos = instanciación

| | | |
|--|--|---|
| A7652 \$500 5/3/2000 Comprobación abrir() cerrar() deposito() retirar() | B2311 -\$50 1/2/2012 Comprobación abrir() cerrar() deposito() retirar() | S2314 \$7500 1/2/1994 Ahorro abrir() cerrar() deposito() retirar() |
|--|--|---|

joseCuenta

aliciaCuenta

juanCuenta

objeto (instancia)

Librerías o frameworks

- Usualmente los lenguajes de POO tienen un conjunto de clases para cadenas, fechas, colecciones, comunicaciones de red, y mucho mas


Java Class Library

NET Framework

C++ Estándar Library

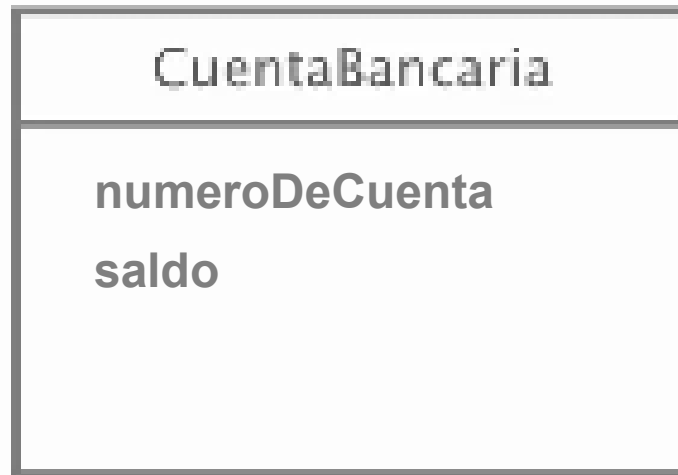
Ruby Estándar Library

Python Estándar Library



Abstracción
Polimorfismo
Herencia
Encapsulación

Abstracción



A7652
\$500
5/3/2000
Comprobación

La cuenta de Jose

B2311
-\$50
1/2/2012
Comprobación

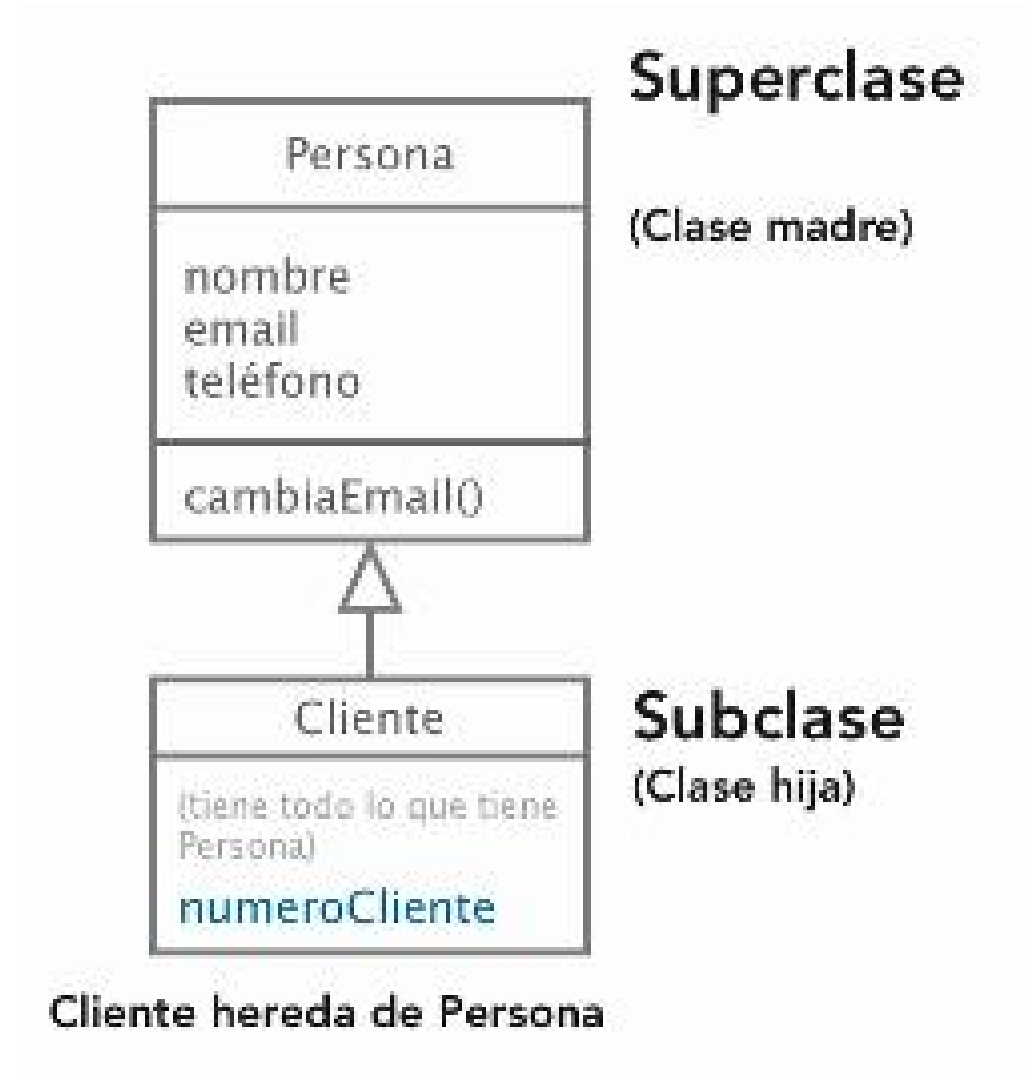
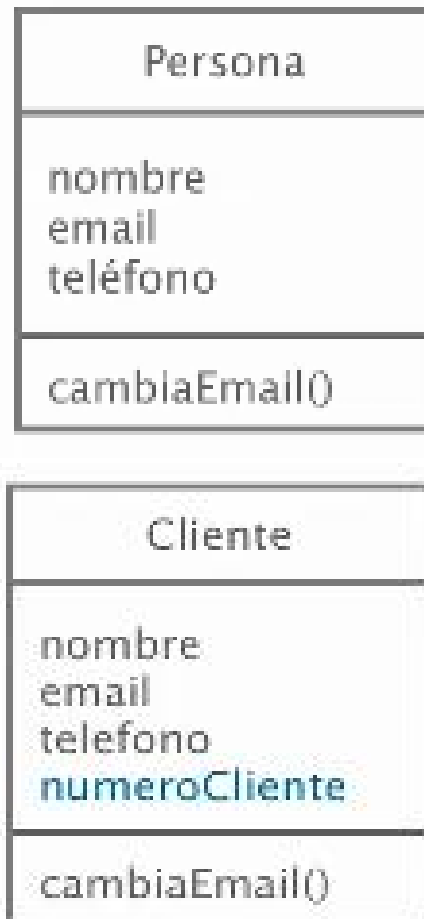
La cuenta de Alicia

Encapsulamiento

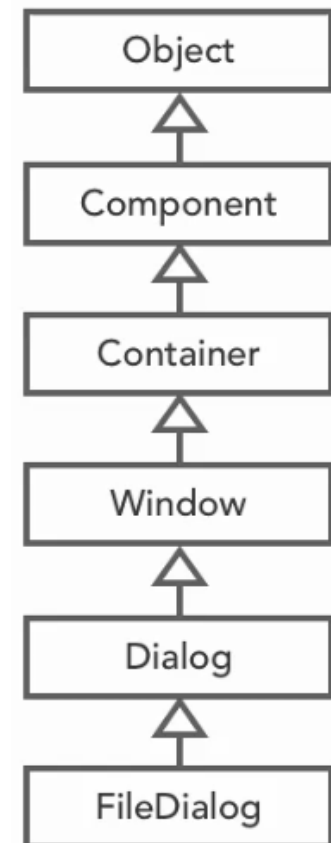
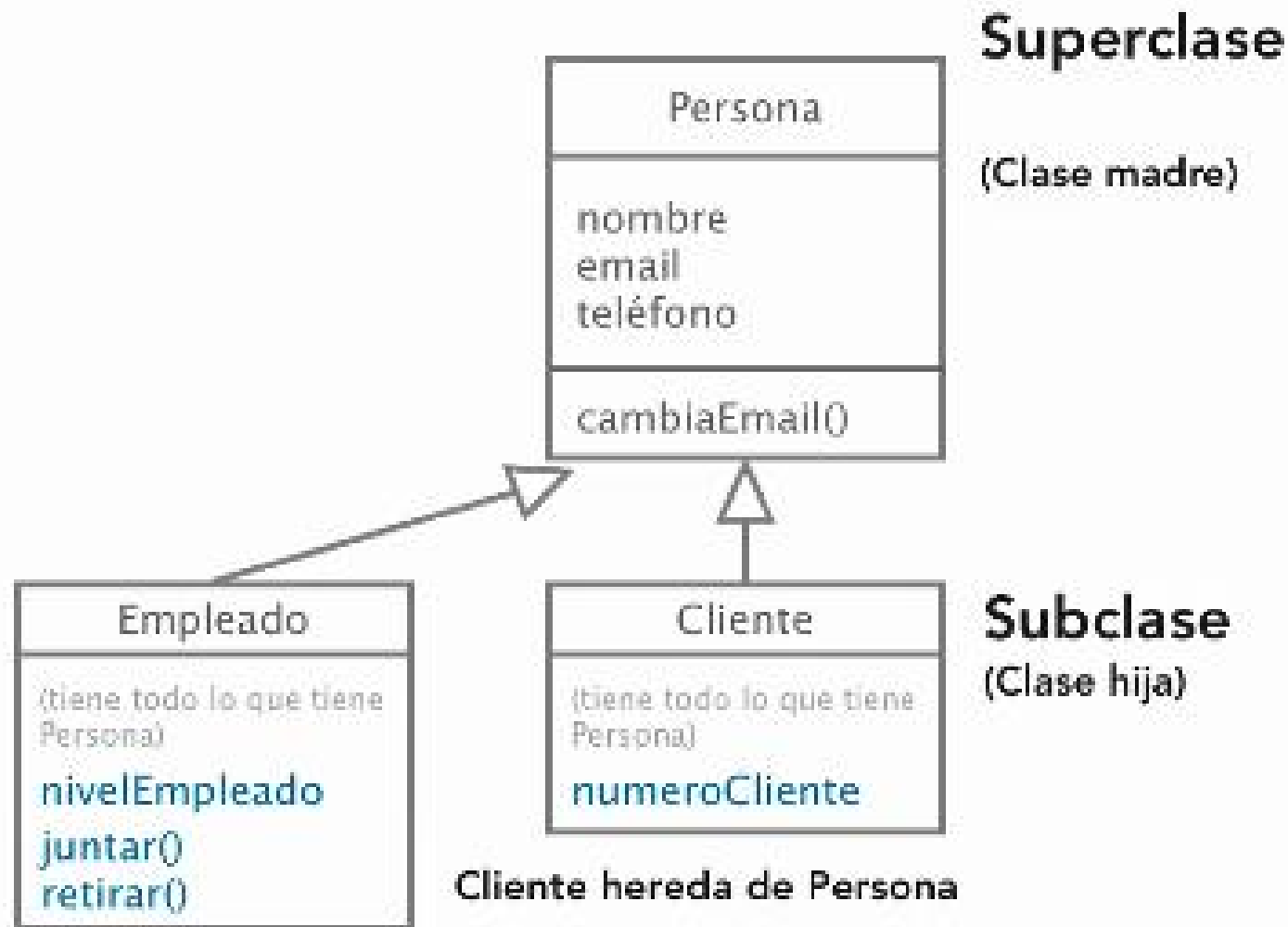


| CuentaBanco |
|--|
| numeroCuenta saldo fechaApertura tipoCuenta |
| abrir() cerrar() depositar() retirar() |

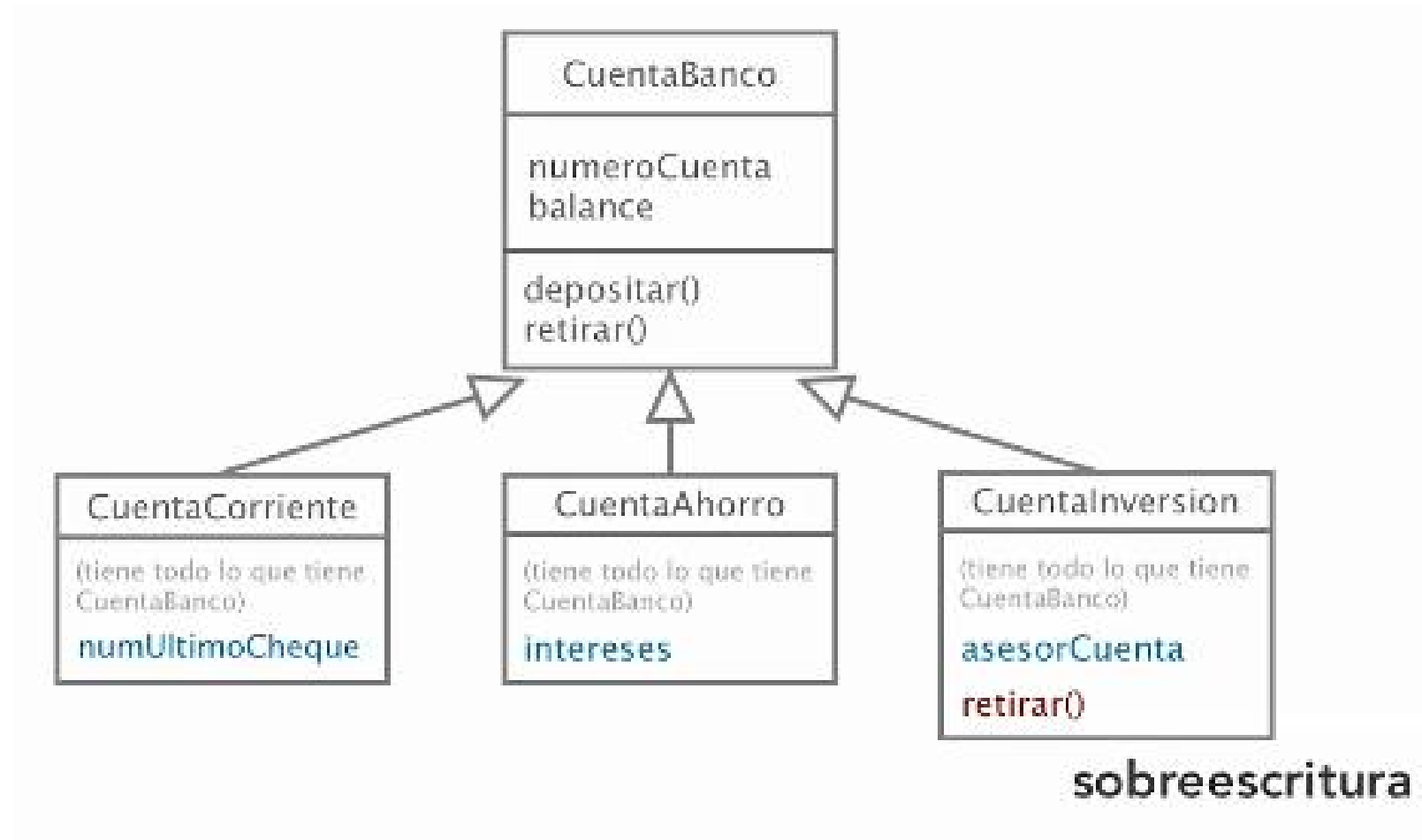
Herencia



Herencia



Polimorfismo



ABSTRACCION Y MODULARIZACION

Abstracción

- La abstracción es la habilidad de ignorar los detalles de las partes para centrar la atención en un nivel mas alto de un problema

Modularización

- La modularización es el proceso de dividir un todo en partes bien definidas que pueden ser construidas y examinadas separadamente

ANÁLISIS Y DISEÑO ORIENTADO A OBJETO

Pasos del proceso

1. Determinar de requerimientos
2. Describir la aplicación
3. Identificar los objetos principales
4. Describir las interacciones
5. Crear un diagrama de clases

DETERMINACION DE REQUERIMIENTOS

Determinación de requerimientos

- Requerimientos funcionales: Lo que hace
 - Características/capacidades
- Requerimientos no funcionales: otras cosas
 - Ayuda
 - Aspectos legales
 - Rendimiento
 - Soporte
 - Seguridad

Requerimientos NO funcionales

- El sistema debe responder a las búsquedas en 2 segundos
- La ayuda telefónica esta disponible de 8 a 20 horas
- Cumplir las restricciones legales
- El soporte debe estar disponible el 99% del tiempo en horario laboral

FURPS / FURPS+

- Functional requeriments
 - Usability requeriments
 - Reliability requeriments
 - Performance requeriments
 - Supportability requeriments
-
- + Requerimiento de forma, implementación, interfaz, físicos

DESCRIPCION DE LA APLICACION

Descripción de la aplicación

- En un lenguaje narrativo describir como funcionará la aplicación

Ejemplo:

- El cliente confirma los elementos en el carrito de compra. El cliente proporciona datos de envío y de pago para procesar la compra. El sistema valida el pago y proporciona un número de pedido que el cliente puede utilizar para realizar un seguimiento de su pedido. El sistema enviará al cliente una copia de su pedido vía email.

Casos de uso

| | |
|------------------|----------------------|
| Título | Cual es objetivo |
| Actor | Quien lo desea |
| Escenario | Como se llega a ello |

El título debe ser una frase corta:

- Registra miembro
- Transfiere fondos
- Compra elementos
- Crea nueva pagina
- Agrupa pagos atrasados



Casos de uso: Actor

- Usuario
- Cliente
- Miembro
- Administrador
- SistemaACME

Caso de uso: Escenario

Título: comprar elementos

Actor: cliente

Escenario: El cliente revisa los elementos en el carrito de compra. El cliente aporta información de pago y envío. El sistema valida la información de pago y responde con confirmación de pedido y provee un identificador de pedido que el cliente puede usar para comprobar el estado de su pedido. El sistema enviará al cliente confirmación de los detalles de su pedido y de un código de seguimiento vía email.

Caso de uso: Escenario

Título: comprar elementos

Actor: cliente

Escenario:

1. El cliente elige entrar al proceso de compra
2. Al cliente se le muestra una página de confirmación de su pedido, permitiéndole cambiar cantidades, quitar elementos o cancelar
3. El cliente introduce su dirección de envío
4. El sistema valida la dirección de envío
5. El cliente selecciona un método de pago
6. El sistema valida los detalles del pago
7. El sistema crea un número de pedido
8. El sistema muestra una pantalla de confirmación al cliente
9. Se envía un email al cliente con los detalles del pedido

Identificación de actores

- Sistemas u organizaciones externos
 - Fuentes de datos externos, servicios Web, otras aplicaciones, sistemas de respaldo
- Roles y grupos de seguridad
 - Visitante, miembro, administrador, propietario
- Trabajos y departamentos
 - Gerente, administradores de nominas, equipo ejecutivo, contabilidad

Identificación de escenarios

Título: Comprar elementos

Actor: Clientes

Escenario: El cliente revisa los artículos en el carrito de compra. El cliente provee información de pago y de envío. El sistema valida la información y responde con un email donde se incluye un código con el que el cliente puede realizar un seguimiento de su pedido. El sistema enviará al cliente dicho email con el código de seguimiento

Extensiones: Que no haya extensiones de uno o mas productos

- a) El cliente quita los elementos sin existencia y continua
- b) El cliente cancela el pedido entero

Diagramas de caso de uso

- Es un tipo de diagrama de UML
- Se muestran diferentes casos de uso
- Se muestran diferentes actores
- Perspectiva global de los miembros y sus interacciones
- No es un reemplazo para los casos de uso escritos

IDENTIFICACION DE OBJETOS (MODELO CONCEPTUAL)

Modelo conceptual del sistema

- Identificar los objetos
- Refinarlos
- Dibujarlos en un diagrama
- Identificar relaciones entre objetos

Escenario de caso de uso

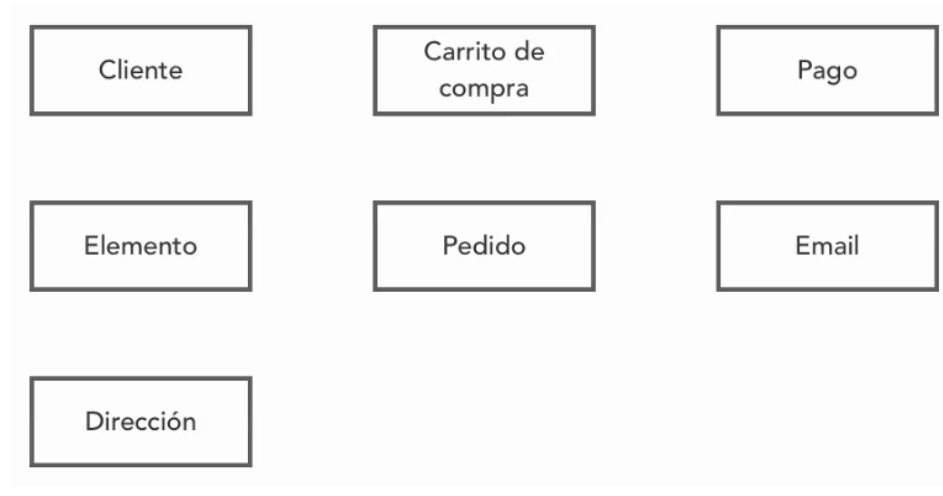
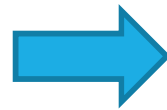
- El cliente confirma los elementos en el carrito de compra. El cliente proporciona datos de envío y de pago para procesar la compra. El sistema valida el pago y proporciona un número de pedido que el cliente puede utilizar para realizar un seguimiento de su pedido. El sistema enviará al cliente una copia de su pedido vía email.

Identificando objetos

- El cliente confirma los elementos en el carrito de compra. El cliente proporciona datos de envío y de pago para procesar la compra. El sistema valida el pago y proporciona un número de pedido que el cliente puede utilizar para realizar un seguimiento de su pedido. El sistema enviará al cliente una copia de su pedido vía email.

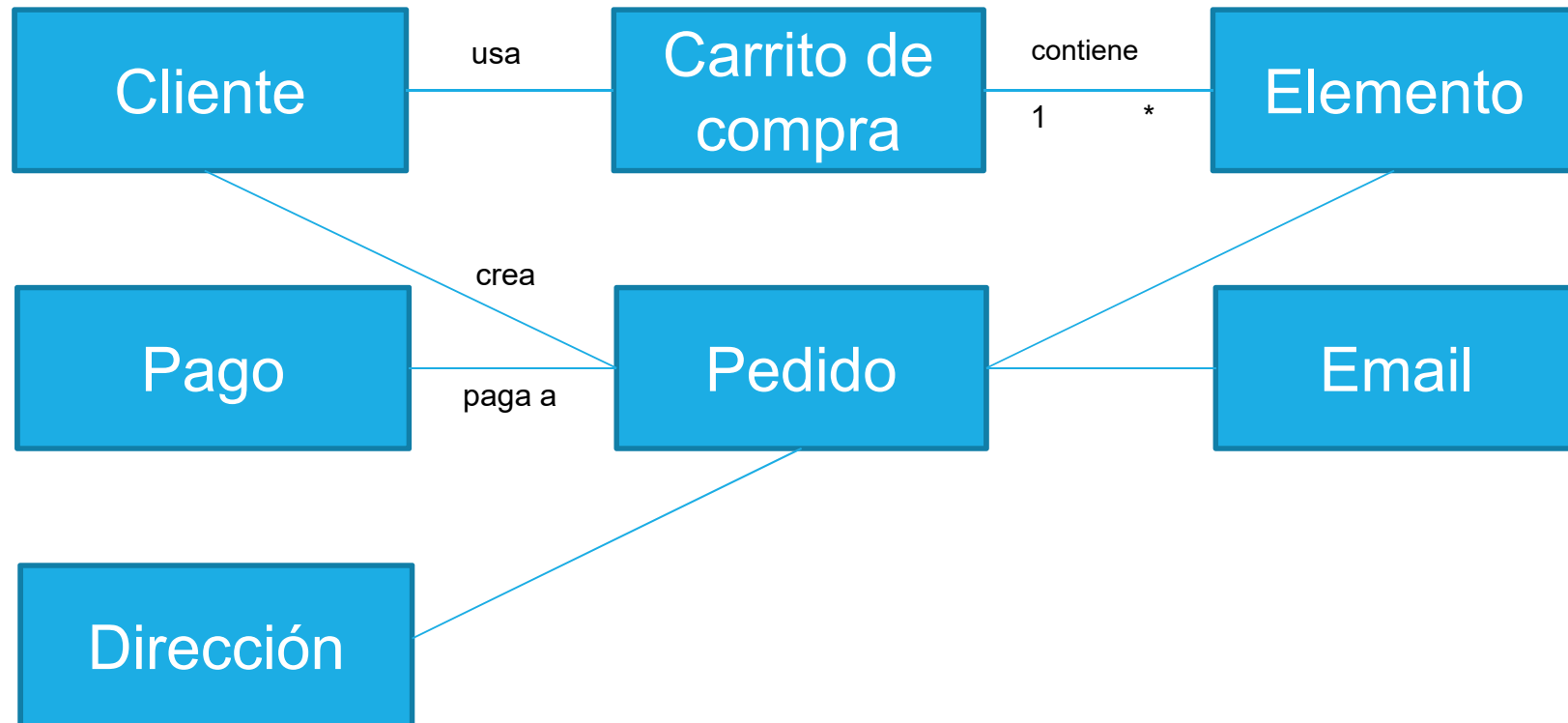
Lista de nombres

- Cliente
- Elemento
- Carrito de compra
- Pago
- Dirección
- ~~Venta~~
- Pedido
- ~~Numero de pedido~~
- ~~Estado del pedido~~
- ~~Detalles del pedido~~
- Email
- ~~Sistema~~



DESCRIBIR LAS INTERACCIONES

Identificar las relaciones entre objetos

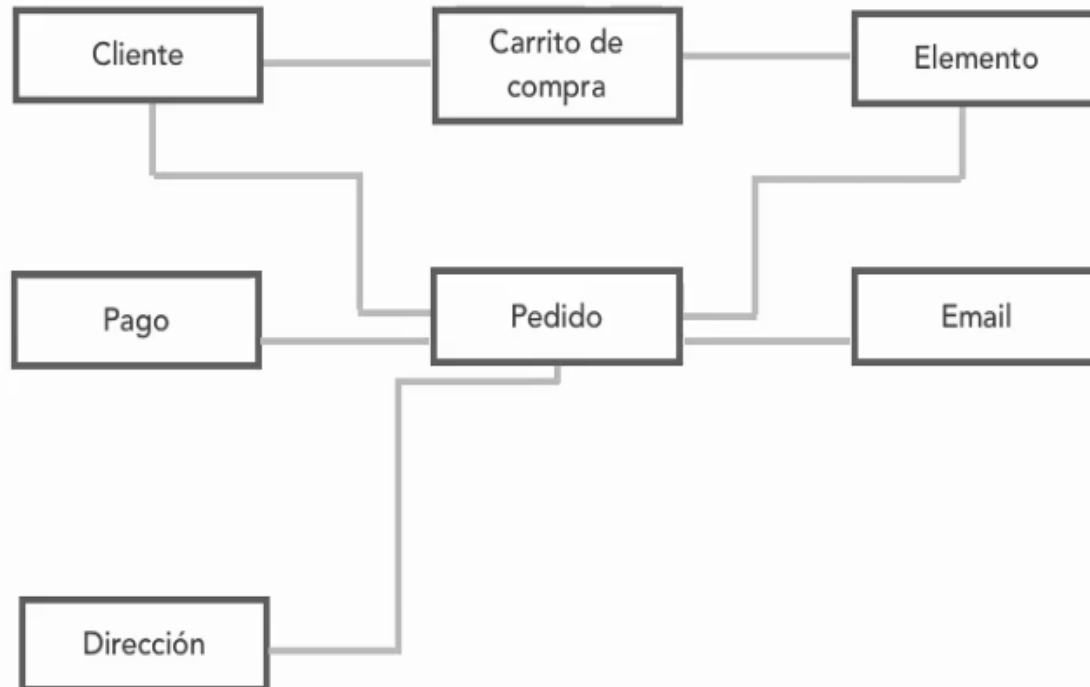


Identificando responsabilidades

- El cliente **verifica los elementos** en el carrito de compra. El cliente **proporciona datos de envío y de pago** para procesar la compra. El sistema **valida el pago** y responde **confirmando el pedido** y proporciona un número de pedido que el cliente puede utilizar para **confirmar el estado** del pedido. El sistema **envía un email** con una copia de los detalles del pedido

Verificar elemento
Procesa venta
Valida pago
Confirma pedido
Confirma estado
Envia email

Comprueba estado del pedido
Envía detalles del pedido
Proporciona datos de envío



Verifica elementos

Provee pago y dirección

Procesa venta

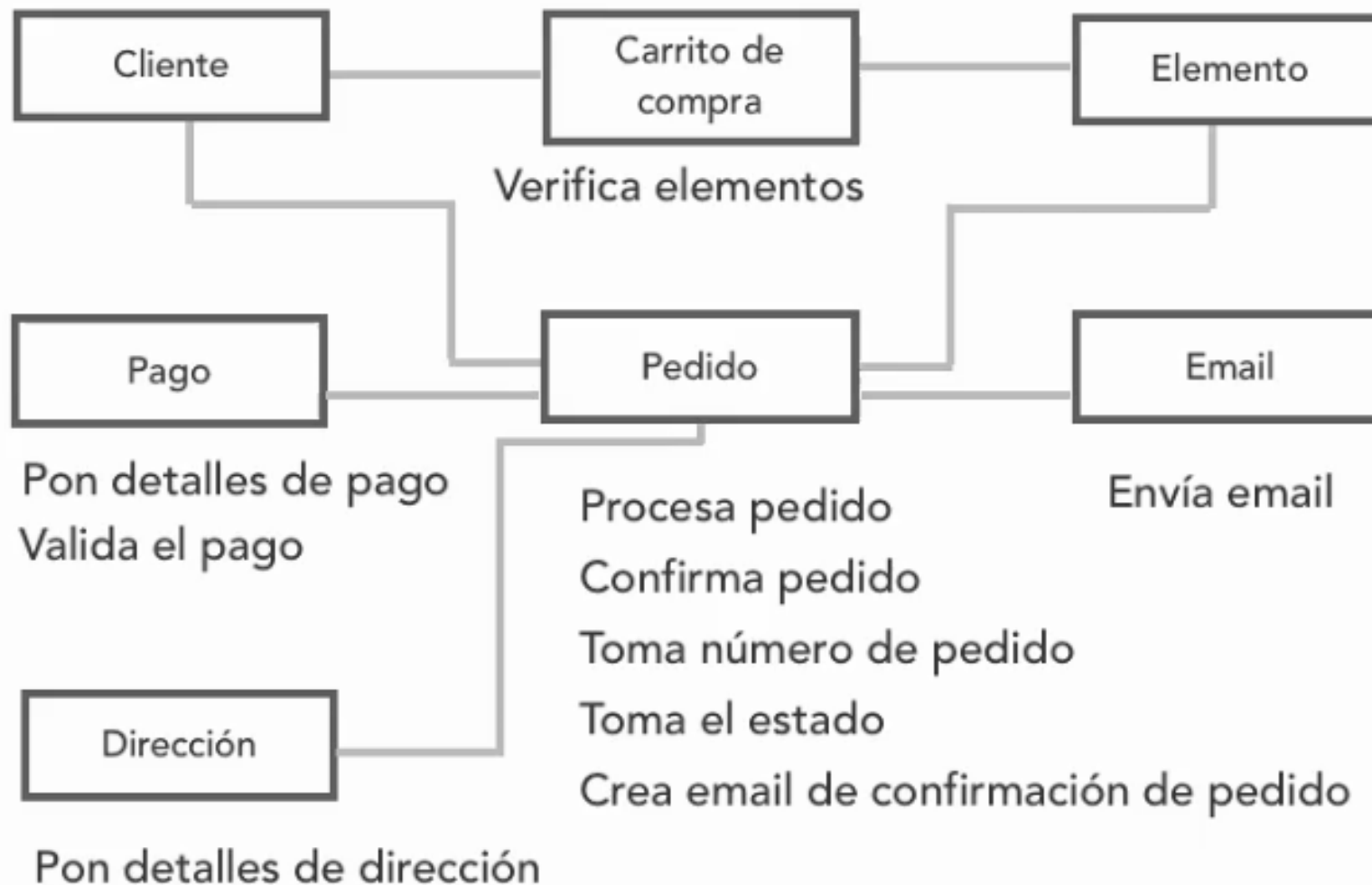
Valida el pago

Confirma pedido

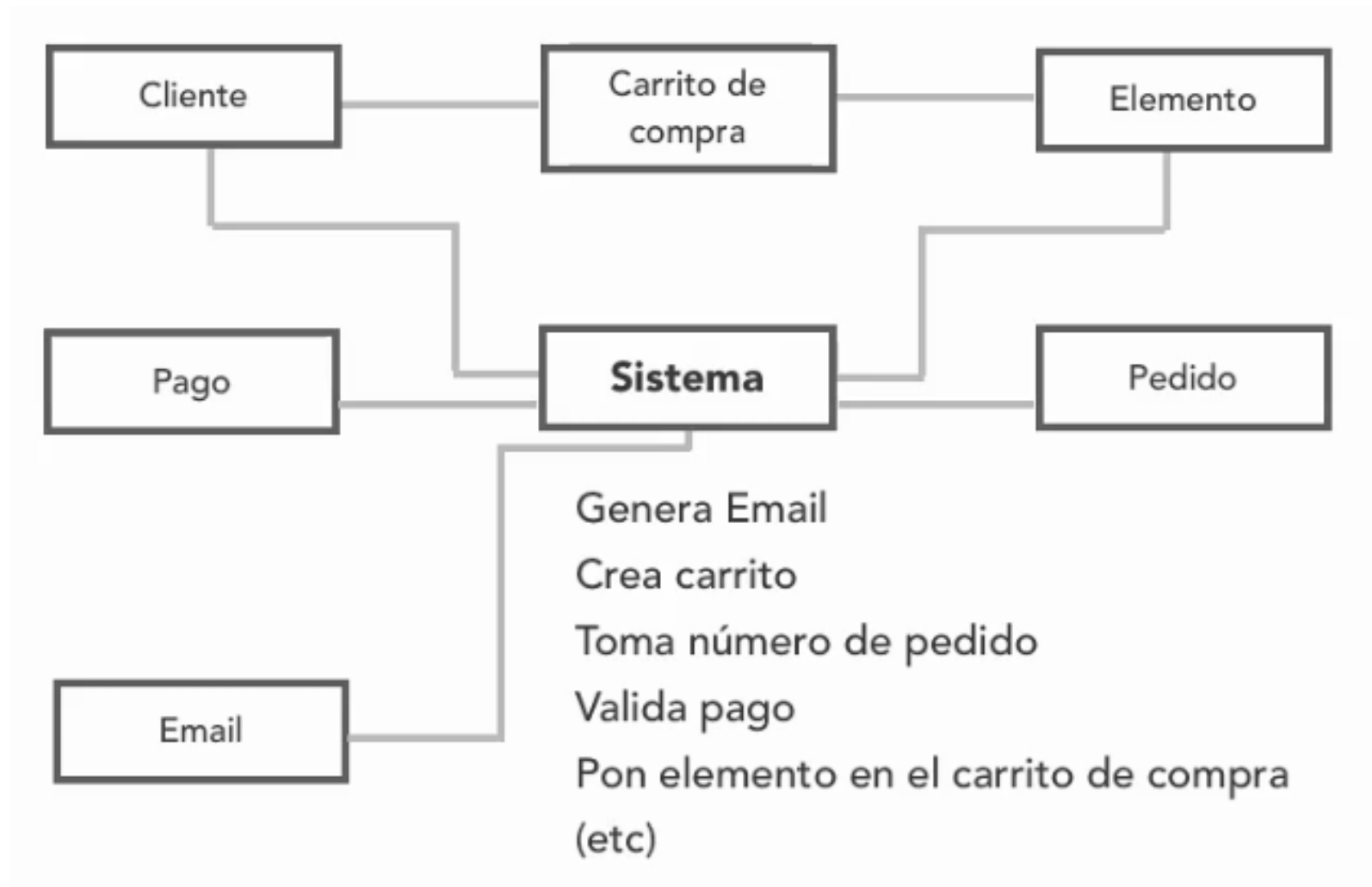
Provee número de pedido

Comprueba el estado del pedido

Envía email con datos del pedido

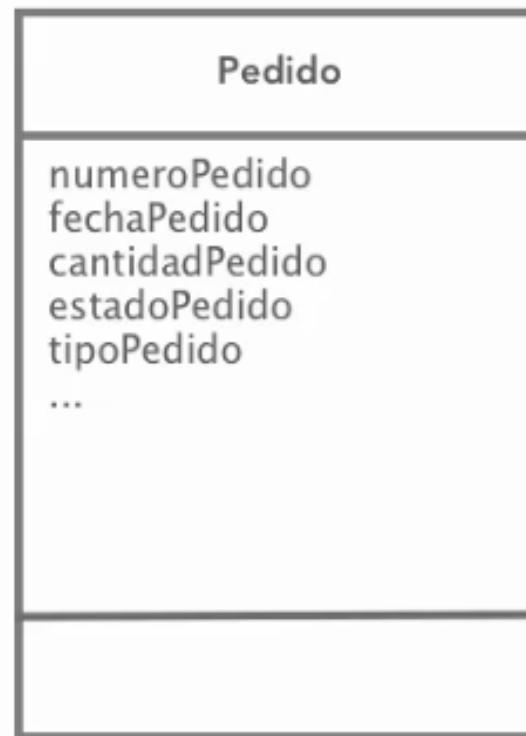


Evitar objetos globales



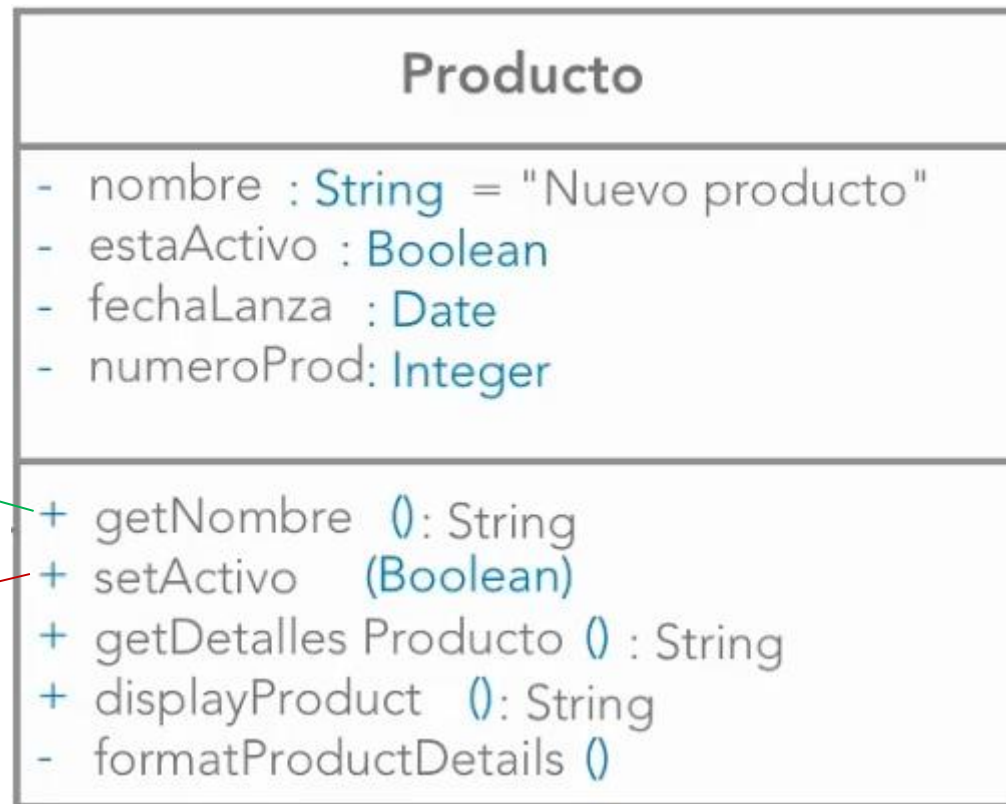
CREAR EL DIAGRAMA DE CLASE

- Tener una lista de clases
- Crear las clases para cada una



GETTER

SETTER



MAQUINA DE REFRESCOS

Definición del problema

Desarrollar un sistema para una máquina expendedora de refrescos

Descripción de la solución

El sistema permite a los usuarios solicitar un refresco introduciendo en la máquina una moneda, cada refresco cuesta 5 Bs, en caso de que la moneda introducida tenga el corte previsto proporciona un refresco, en caso contrario devuelve la moneda. La máquina tiene una cantidad definida de refrescos y almacena todas las monedas en una caja.

Identificación de posibles objetos

El sistema permite a los usuarios solicitar un refresco introduciendo en la máquina una moneda, cada refresco cuesta 5 Bs, en caso de que la moneda introducida tenga el corte previsto proporciona un refresco, en caso contrario devuelve la moneda. La máquina tiene una cantidad definida de refrescos y almacena todas las monedas en una caja.

Maquina es un concepto importante y por tanto objeto

Usuario, moneda, refresco también sin embargo no tiene la relevancia para constituirse en objetos

Asociación de atributos

El sistema permite a los usuarios solicitar un refresco introduciendo a la máquina una moneda, cada refresco cuesta 5 Bs, en caso de que la moneda introducida tenga el corte previsto proporciona un refresco, en caso contrario devuelve la moneda. La máquina tiene una cantidad definida de refrescos y almacena todas las monedas en una caja.

| Objeto | Atributos |
|---------|---------------------------|
| Maquina | Costo Cantidad Caja |

Identificación de métodos

El sistema permite a los usuarios solicitar un refresco introduciendo a la máquina una moneda, cada refresco cuesta 5 Bs, en caso de que la moneda introducida tenga el corte previsto proporciona un refresco, en caso contrario devuelve la moneda. La máquina tiene una cantidad definida de refrescos y almacena todas las monedas en una caja.

| Objeto | Métodos |
|---------|--|
| Maquina | Solicitar Entregar IngresarRefrescos ActualizarPrecio ActualizarCaja |

MaquinaDeRefrescos

- costo: int
- cantidad: int
- caja: int

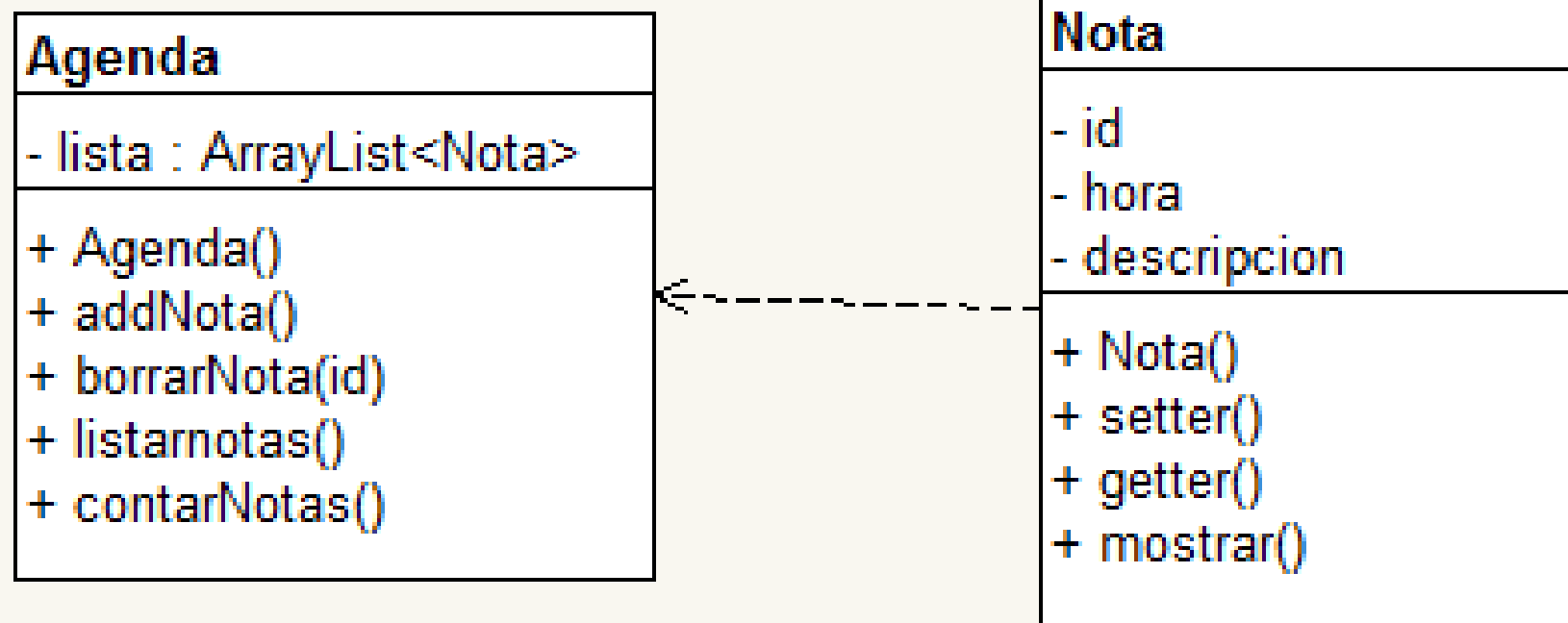
- + MaquinaDeRefrescos()
- + GETTER
- + SETTER
- + solicitarRefresco(dinero)
- + actualizarPrecio(monto)
- + llenarMaquina(refrescos)
- + entregarRefresco()

AGENDA PERSONAL

Aplicación agenda personal

Modelar una aplicación que represente una agenda personal con las siguientes características básicas.

- Almacenar notas, cada nota tiene id, hora, descripción
- El número de notas que se pueden almacenar no tiene límite
- Mostrará la notas de forma individual
- Nos informará sobre la cantidad de notas que tiene actualmente almacenadas



Colecciones en Java

Librería requerida para el uso de colecciones

```
import java.util.ArrayList;
```

Declaración de una colección

```
private ArrayList<String> notas;
```

Inicialización de una colección

```
notas = new ArrayList<String>();
```



Adicionar elementos en colección

```
notas.add(nota);
```

Obtener la cantidad de elementos que tiene la colección

```
notas.size();
```

Eliminar elementos de una colección

```
notas.remove(numeroDeNota);
```

Mostrar un elemento de la colección a través del índice

```
notas.get(numeroDeNota)
```

Recorrer una colección - foreach

```
for (TipoDelElemento elemento : colección) {  
    cuerpo del ciclo  
}
```

```
/**  
 * Imprime todas las notas de la agenda  
 */  
public void imprimirNotas()  
{  
    for(String nota : notas) {  
        System.out.println(nota);  
    }  
}
```

Recorrido con While

```
while (condición del ciclo) {  
    cuerpo del ciclo  
}
```

```
int indice = 0;  
while(indice < notas.size()) {  
    System.out.println(notas.get(indice));  
    indice ++;  
}
```

Recorrido con Iterator

```
import java.util.ArrayList;
import java.util.Iterator;
```

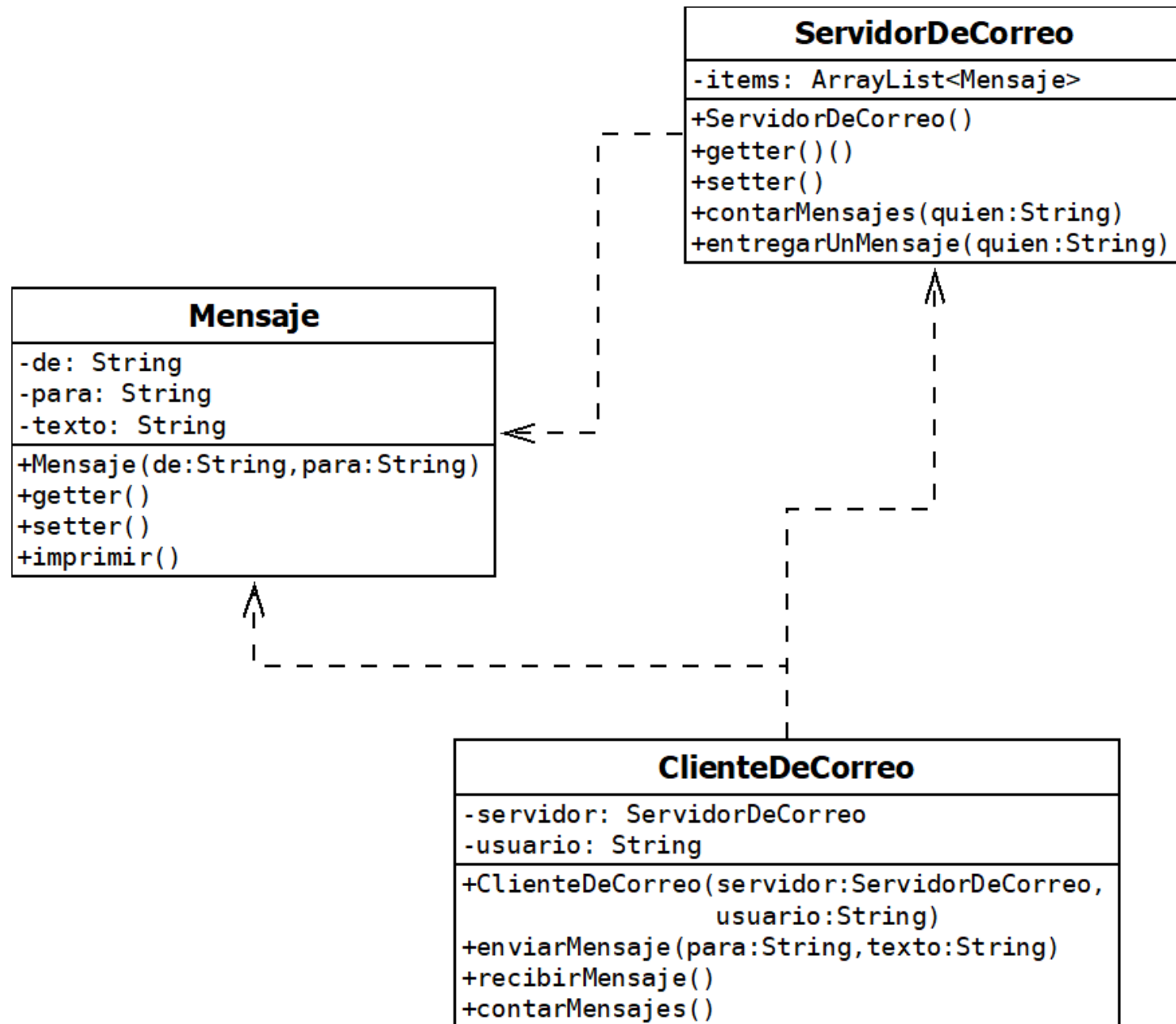
```
Iterator<TipoDelElemento> it = miColeccion.iterator();
while (it.hasNext()) {
    Invocar it.next() para obtener el siguiente elemento
    Hacer algo con dicho elemento
}
```

```
public void listarTodasLasNotas()
{
    Iterator<String> it = notas.iterator();
    while (it.hasNext()) {
        System.out.println(it.next());
    }
}
```



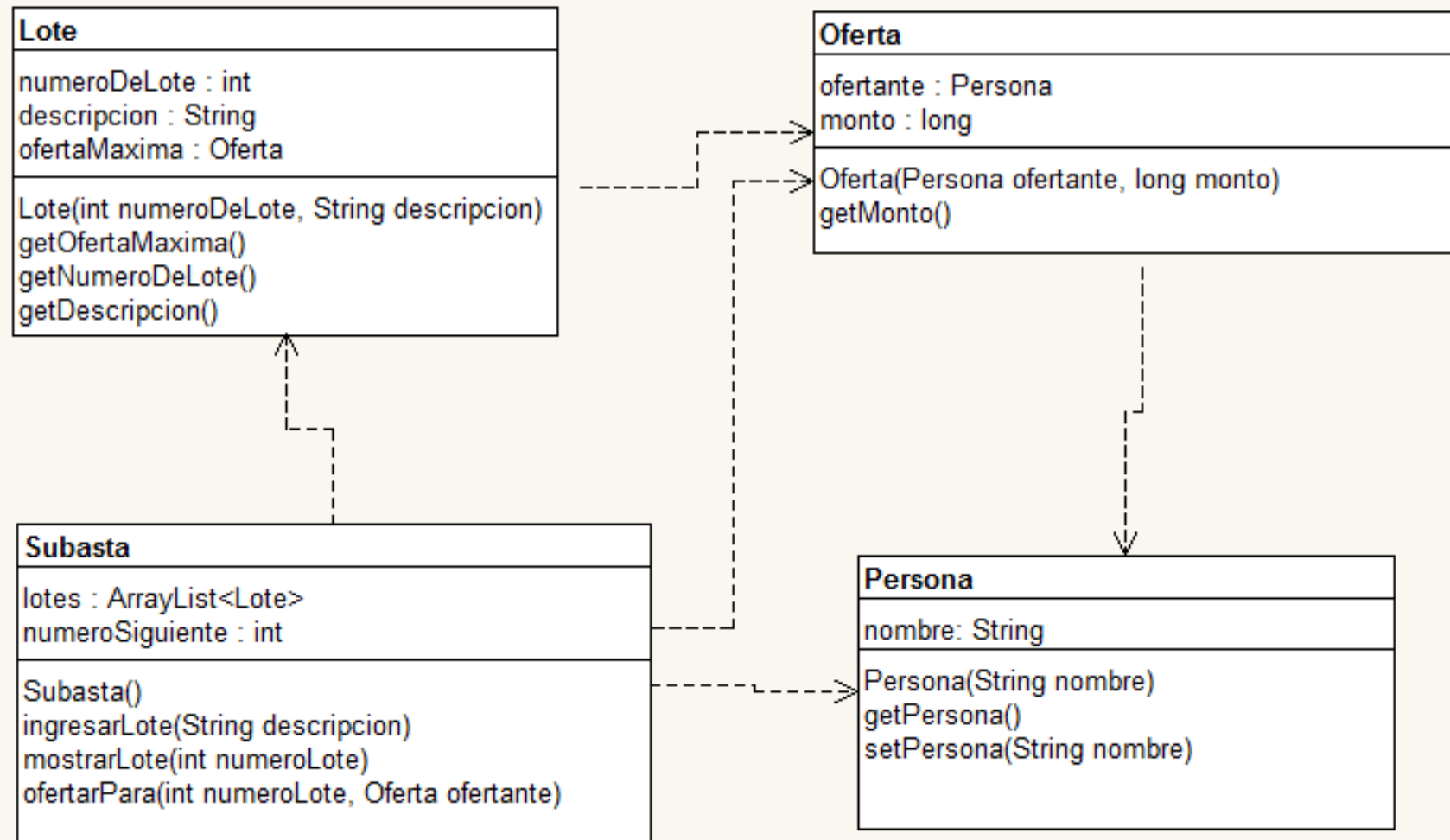
Sistema de correo electrónico

- La idea de la aplicación es simular las acciones para que usuarios se puedan enviar mensajes entre ellos. Un usuario utilizar un cliente de correo para enviar mensajes a un servidor que se encarga de despacharlos al cliente de correo de otro usuario.



Sistema de subastas

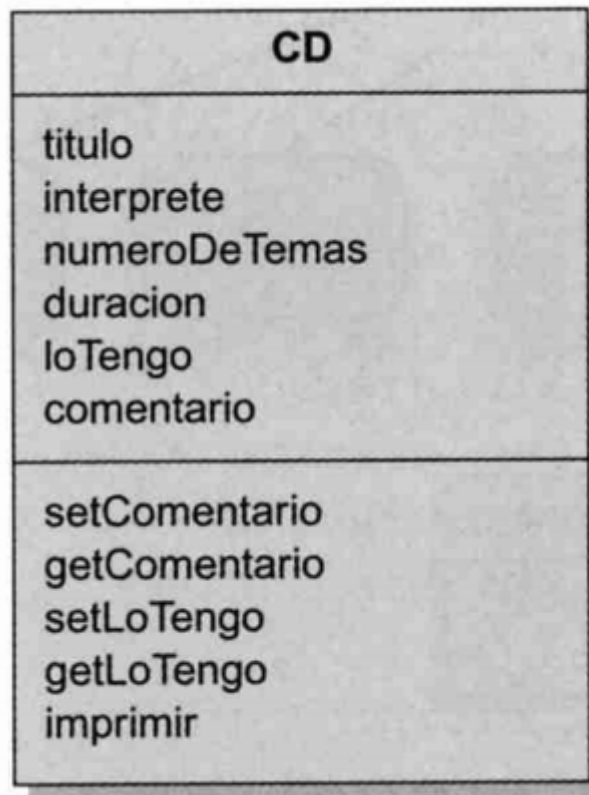
- La idea central consiste en un conjunto de elementos que se ofrecen para su venta. Estos elementos se denominan "Lotes" y a cada lote se le asigna un número único que lo identifica. Una persona trata de comprar el lote que desea ofreciendo cierta cantidad de dinero por él. Nuestras subastas son ligeramente diferentes de otras porque ofrecen todos los lotes por tiempo limitado. Al finalizar este tiempo, se cierra la subasta y se considera compradora del lote a la persona que ofertó la mayor cantidad de dinero. Si, al cierre de la subasta el lote no tiene ofertantes, se lo considera no vendido y estos lotes pueden ser ofrecidos en posteriores subastas.



EJEMPLO HERENCIA

Proyecto Entretenimientos Multimediales

DoME



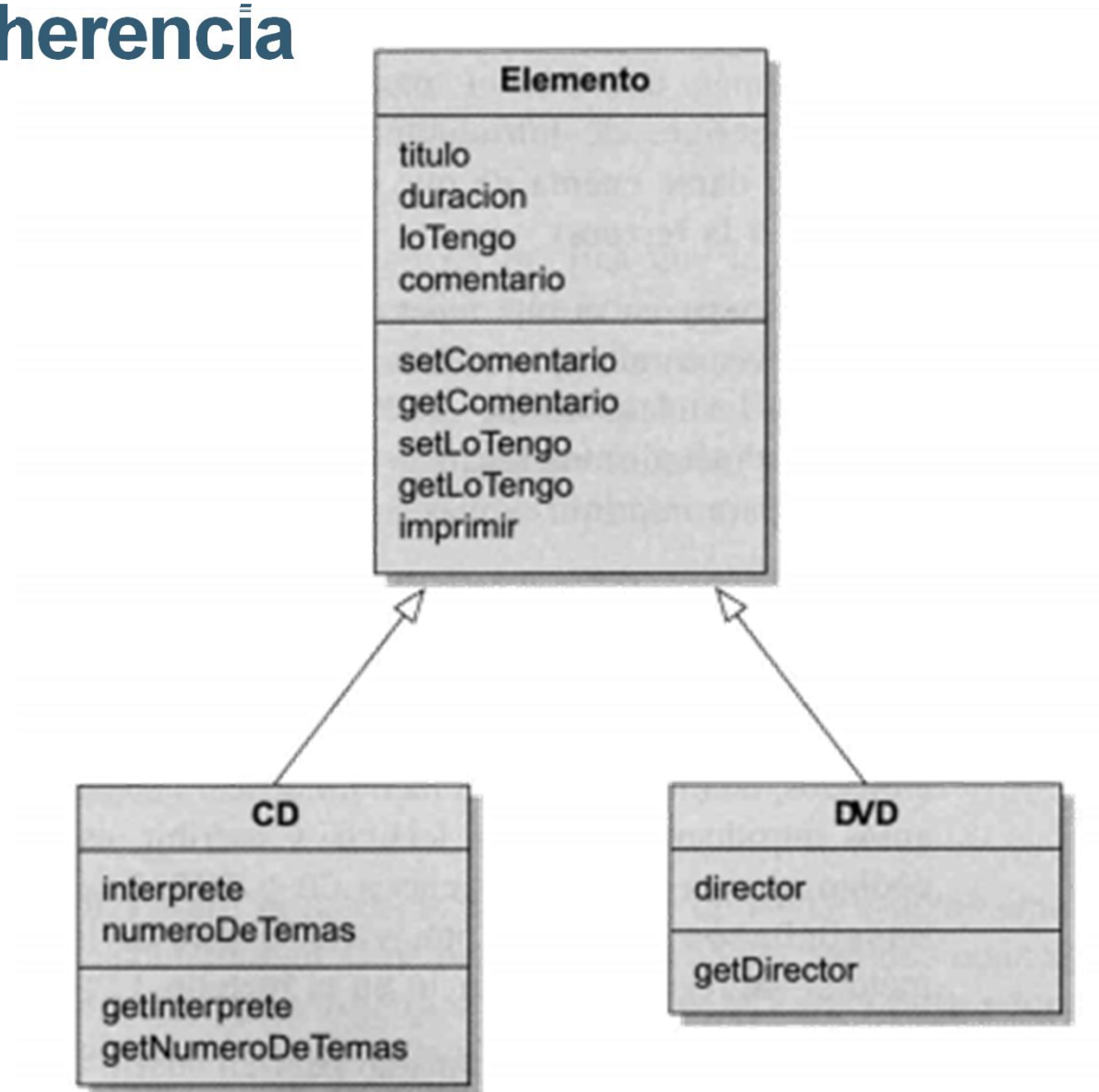
Críticas a la aplicación

- Las clases son muy parecidas
- Duplicación de código en clases similares
- Existe riesgo de errores en mantenimiento
- El mantenimiento es mas tedioso si el código es repetido en diferentes clase

Herencia

- La herencia es uno de los mecanismos de los lenguajes de POO basados en clases, por medio del cual una clase se deriva de otra de manera que extiende su funcionalidad.
- La clase de la que se hereda se suele denominar *clase base*, *clase padre*, *superclase*
- La clase derivada se suele denominar clase derivada, clase hijo, subclase

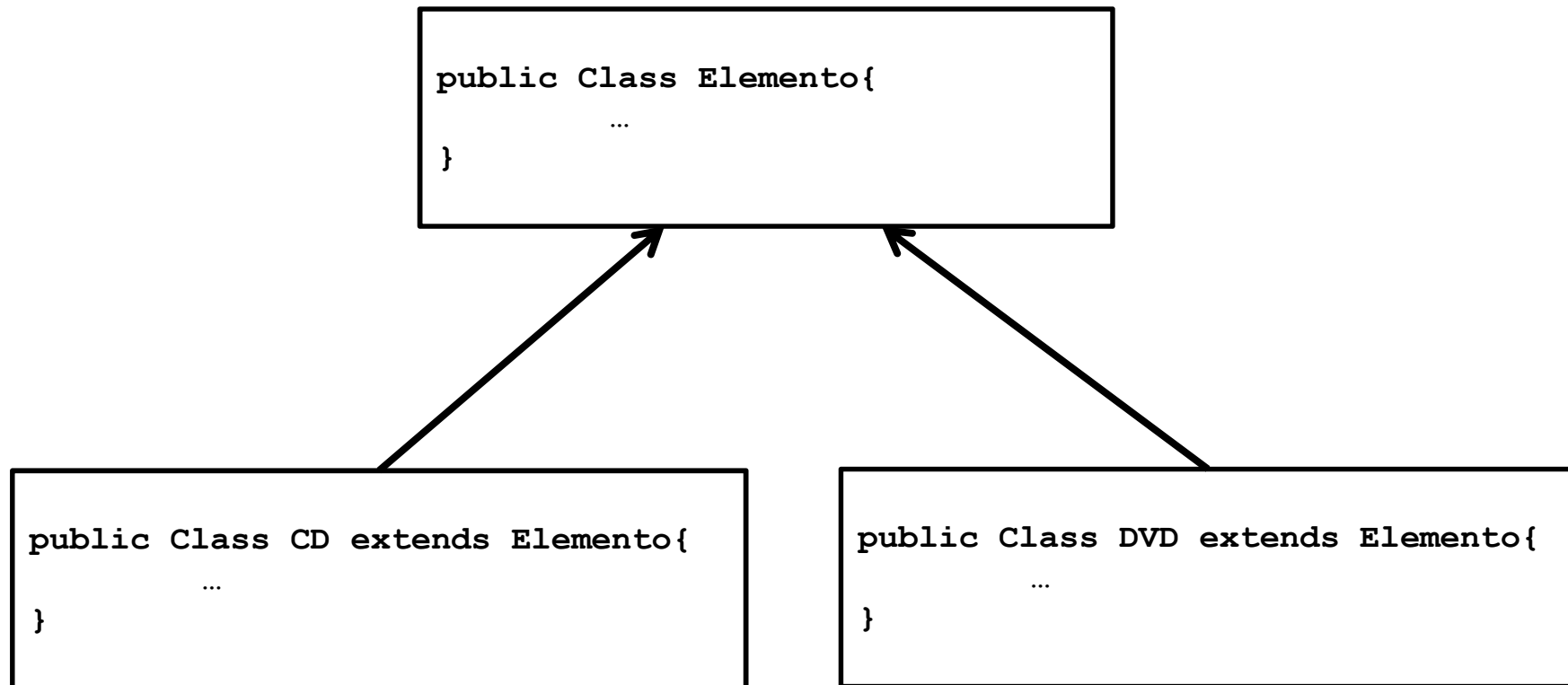
Uso de la herencia



Uso de la herencia

- Se define una superclase: Elemento
- Se definen subclases para CD y DVD
- La superclase define atributos comunes
- Las subclases heredan los atributos de la superclase
- Las subclases pueden tener sus propios atributos

Definición de herencia en Java



La superclase Elemento

```
public class Elemento
{
    private String titulo;
    private int duracion;
    private boolean loTengo;
    private String comentario;

    public Elemento(String elTitulo, int tiempo)
    {
        titulo = elTitulo;
        duracion = tiempo;
        loTengo = false;
        comentario = "";
    }

    // Falta el resto de los metodos
}
```

La subclase CD

```
public class CD extends Elemento
{
    private String interprete;
    private int numeroDeTemas;

    public CD(String elTitulo, String elInterprete, int temas, int tiempo)
    {
        super(elTitulo, tiempo);
        interprete = elInterprete;
        numeroDeTemas = temas;
    }

    public String getInterprete() {
        return interprete;
    }

    public int getNumeroDeTemas() {
        return numeroDeTemas;
    }

    public void imprimir()
    {
        System.out.println("CD");
        System.out.println("El interprete es:" + interprete);
        System.out.println("Número de temas:" + numeroDeTemas );
    }
}
```

La subclase DVD

```
public class DVD extends Elemento
{
    private String director;

    public DVD(String elTitulo, int tiempo, String director)
    {
        super(elTitulo, tiempo);
        this.director = director;
    }

    public void setDirector(String director)
    {
        this.director = director;
    }

    public String getDirector()
    {
        return director;
    }

    public void imprimir()
    {
        System.out.println("DVD");
        System.out.println("El director es:" + director);
    }
}
```



Blog

Desarrollar una aplicación para la administración de un Blog. Un Blog es un sitio Web donde uno o varios autores escriben textos o artículos que son almacenados de forma cronológica. Cada uno de los artículos escritos se denominan Entradas y existen diferentes tipos de entradas (para el caso en particular se consideran textos y fotos). También cada artículo puede tener comentarios y la posibilidad de calificar una entrada con un “Me gusta”.

