Encapsulación

Encapsulación

- Protección de datos dentro de la clase (no sean modificados inapropiadamente)
- Metodología para:
  - Ocultar ciertos elementos de la implementación de una clase
  - Proporcionando una interfaz pública al cliente (usuario) del software (clase)
  - Generar "cápsulas seguras"

Encapsulación (II)

- Hay una sola **interfaz pública** segura para acceder al **contenido privado** del siguiente objeto

Encapsulación

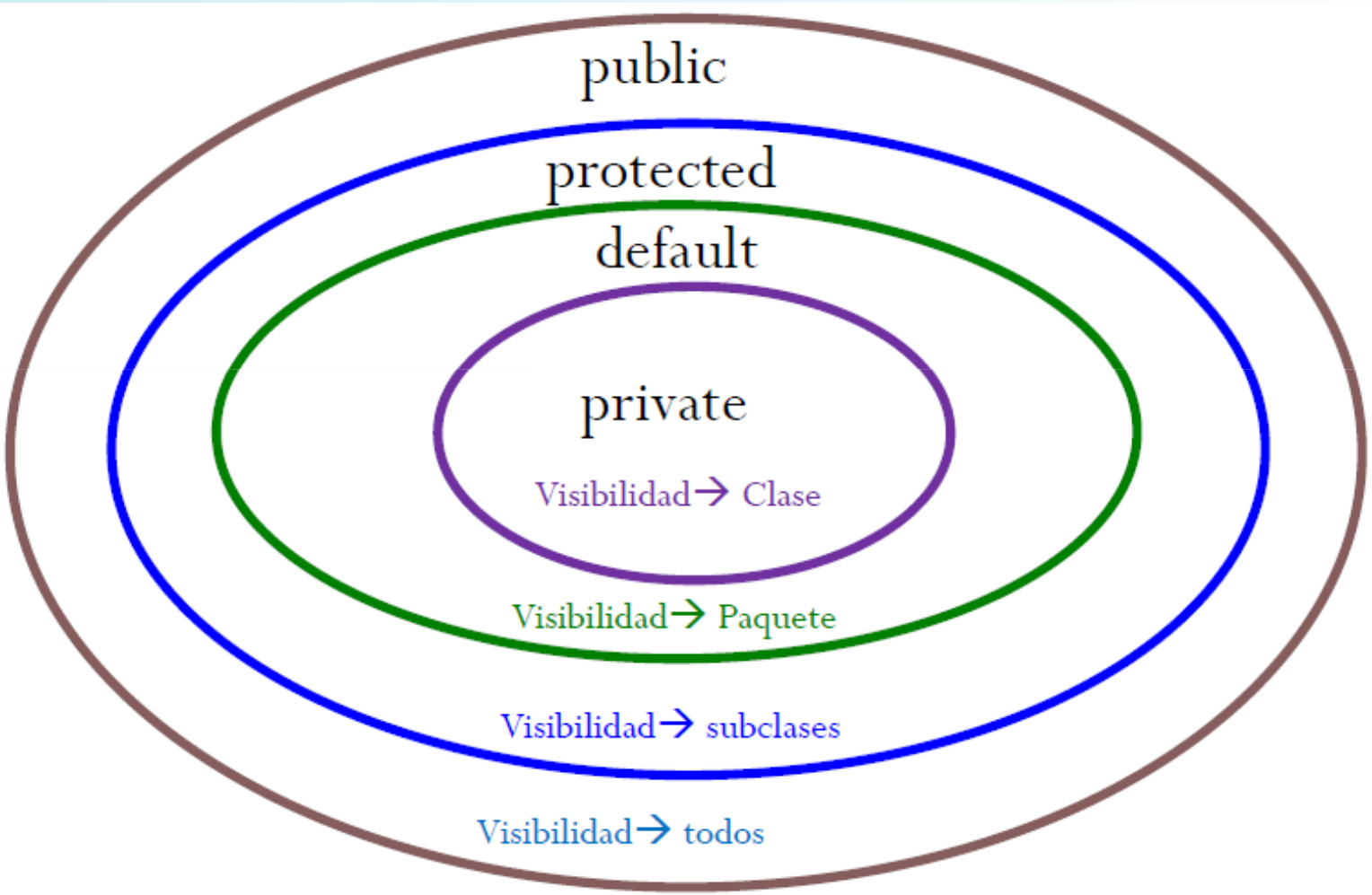
Encapsulación (III)

- Para asegurarse que la clase está correctamente encapsulada:
  - Aplicar modificadores de visibilidad a:
    - **Clases**
    - Variables **atributo**
    - **Métodos declarados**
- Todas las clases, métodos y variables poseen un modificador de acceso

Modificadores de acceso

- En Java existen **cuatro tipos de acceso y sólo tres** modificadores:
  - **Public**
  - **Private**
  - **Protected**
  - *Default* (amistoso)

Modificadores de acceso en atributos y métodos



Modificador *public*

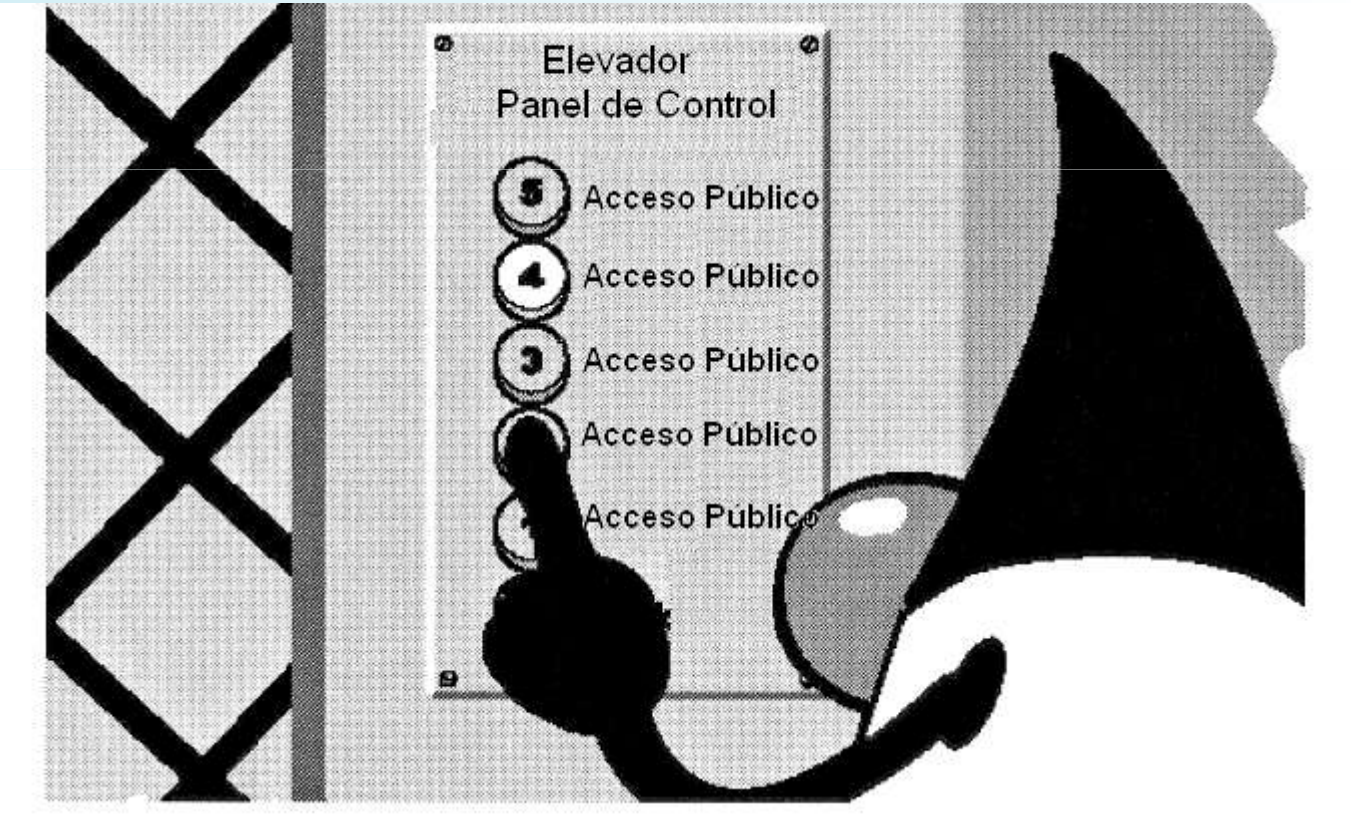
- **public** permite a cualquier objeto acceder a
  - Clases
  - Atributos
  - Métodos

Modificador public (Acceso a clases)

- Restricciones
  - Una clase no puede ser privada ni protegida (excepto clases internas)
  - Sólo puede existir **una clase publica por cada archivo de código fuente**
  - El nombre del archivo debe coincidir con el nombre de la clase pública

Modificador *public* (II)

- Acceso público a cualquier piso, aún áreas donde no deben ingresar visitantes



Modificador *public* (III)

- Colocar enfrente de la variable atributo o método el modificador **public**

```
public int pisoActual = 1;

public void setPiso(int pisoDeseado)
{
    …
}
```

Problemas potenciales *public*

- **Clase sin encapsular**

```
/* Archivo: ElevadorPublico.java */
public class ElevadorPublico {
    // definición de atributos
    public boolean ptasAbiertas = false;
    public int pisoActual =1;
    public int peso=0;
    // definición de constantes
    public static final int CAPACIDAD = 500;
    public static final int PISO_MAX= 5;
    public static final int PISO_MIN= 1;
}
```

```
// Clase TestElevadorPublico.
public class TestElevadorPublico {
   public static void main (String[] args) {
        ElevadorPublico ep = new ElevadorPublico( );
        ep.ptasAbiertas=true;
        ep.ptasAbiertas=false;
        System.out.println("piso actual:"+ ep.pisoActual);
        // bajar al piso 0
        ep.pisoActual--;
        System.out.println("piso actual:"+ ep.pisoActual);
        ep.pisoActual++;
        System.out.println("piso actual:"+ ep.pisoActual);
        // pasar al piso 7 (sólo hay 5 pisos)
        ep.pisoActual=7;
        System.out.println("piso actual:"+ ep.pisoActual);
        ep.ptasAbiertas=true; // los pasajeros suben/bajan
        ep.ptasAbiertas=false; // los pasajeros suben/bajan
        ep.pisoActual=1; // pasa al piso 1
        ep.ptasAbiertas=true; // los pasajeros suben/bajan
        System.out.println("puertas abiertas->"+ ep.ptasAbiertas);
        ep.pisoActual++;
        System.out.println("piso actual:"+ ep.pisoActual);
   } // fin main
} // fin de la clase
```

Ejercicio I

- Analizar el código anterior, y comentar qué ocurre:
  - ¿Cómo funciona el elevador programado a comparación de uno real?
  - ¿El funcionamiento obtenido es diferente al funcionamiento esperado?

Modificador *private*

- No permite que otros objetos accedan a:
  - Atributos y
  - Métodos

  de ninguna clase.

Modificador *private* (II)

- Se coloca el modificador **private** delaten de la variable o método

```
private int pisoActual = 1;

private void getCapacidad()
{
    …
}
```

Clase ElevadorPrivado

- **Hacer privados todos los atributos**

```
/* Archivo: ElevadorPublico.java */
public class ElevadorPublico {
    // definición de atributos
    private boolean ptasAbiertas = false;
    private int pisoActual =1;
    private int peso=0;
    // definición de constantes
    private static final int CAPACIDAD = 500;
    private static final int PISO_MAX= 5;
    private static final int PISO_MIN= 1;
}
```

Sugerencias para la encapsulación

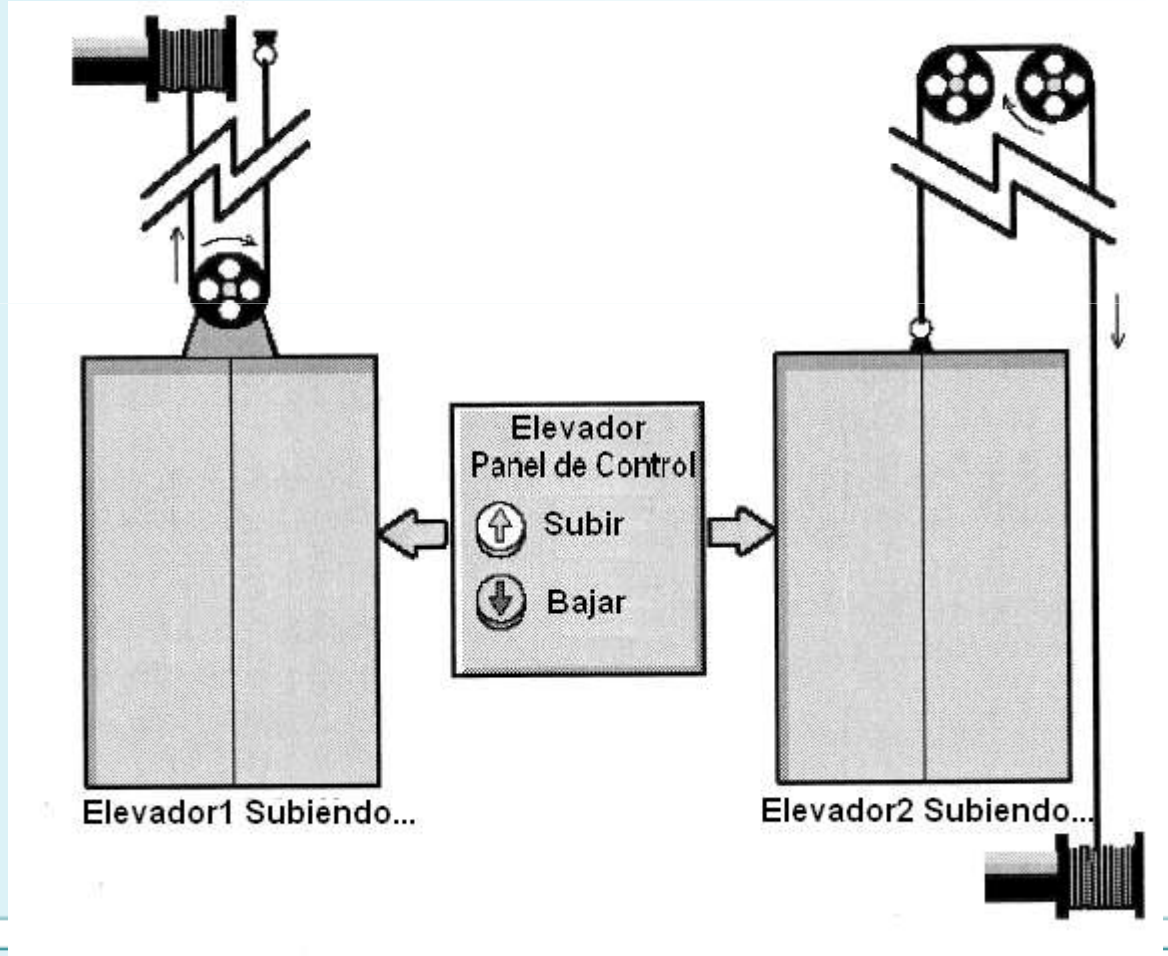
- En un **programa ideal**, la mayoría de los atributos deben ser **privados**
- Los atributos se deben modificar y/o consultar por medio de los métodos que implemente la clase
- Los métodos deben:
  - Proporcionar la lógica de negocio
  - Aesgurar la asignación válida de datos a los atributos

Sugerencias para la encapsulación

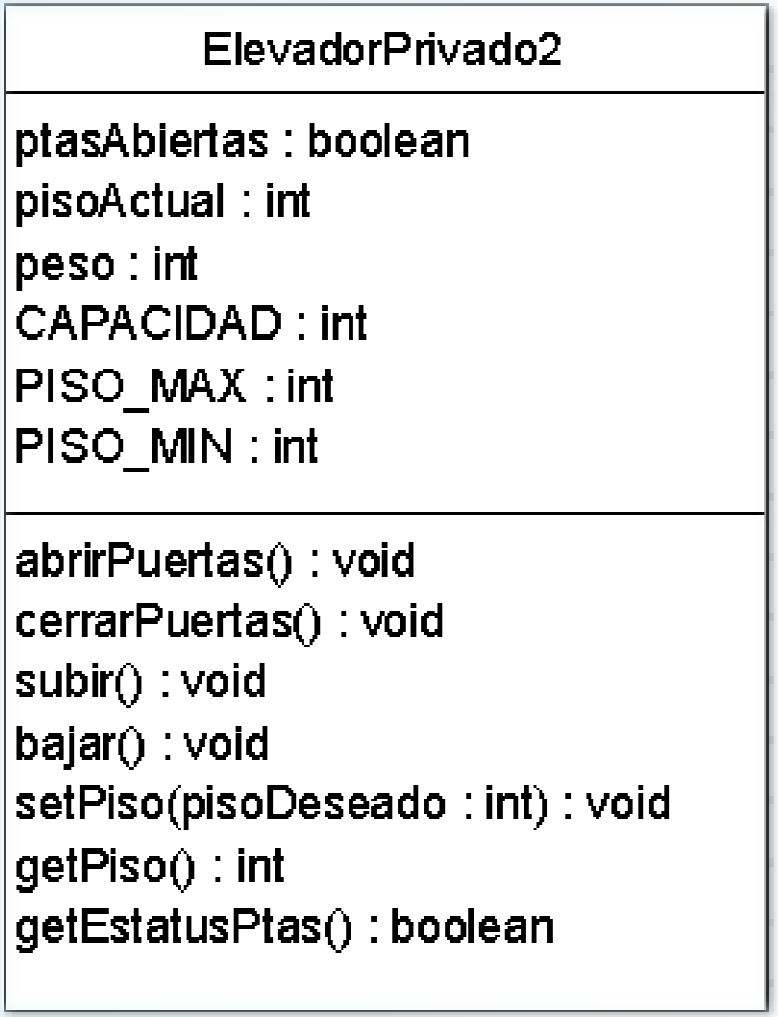
- Los métodos de una clase constituyen su interfaz.
- La interfaz es el medio por el que se puede comunicar (enviar/recibir mensajes) a un objeto.
- La implementación de un método es el detalle de cómo una clase realiza una tarea en específico.
- Un objetivo de la POO es crear métodos públicos cuya implementación puede cambiar **sin afectar la interfaz**.

Interfaz e Implementación

- Misma **interfaz** diferente **implementación**



Ejercicio II



Aplicabilidad de modificadores

	Private	Default	Protected	Public
Clase	No	Sí	No	Sí
Método	Sí	Sí	Sí	Sí
Atributo	Sí	Sí	Sí	Sí

Ejercicio III

- Considere el concepto de encapsulación para crear una clase *PCuadrado* que represente un cuadrado dividido por sus ángulos opuestos.

  setLados → indica la longitud de los lados
  getArea → calcula el área de la figura dibujada
  getPerimetro → calcula el perímetro de la figura dibuj.
  dibujar → dibuje la figura usando asteriscos (*)

  Nota: validar que los objetos construidos no rebasen una longitud de 50 unidades.

Ejercicio III (continuación)

- Una figura del tipo *Cuadrado* de 5x5 se dibuja de la siguiente manera:

```
* * * * *
* * * *
* * *
* *
*
```

A= 12.5
P= 17.071