Algoritmos y Estructura de Datos

Unidad 1: Clases y objetos

Tema 2: Control de acceso y encapsulamiento

Semana 02

Tema 2: Control de acceso y encapsulamiento

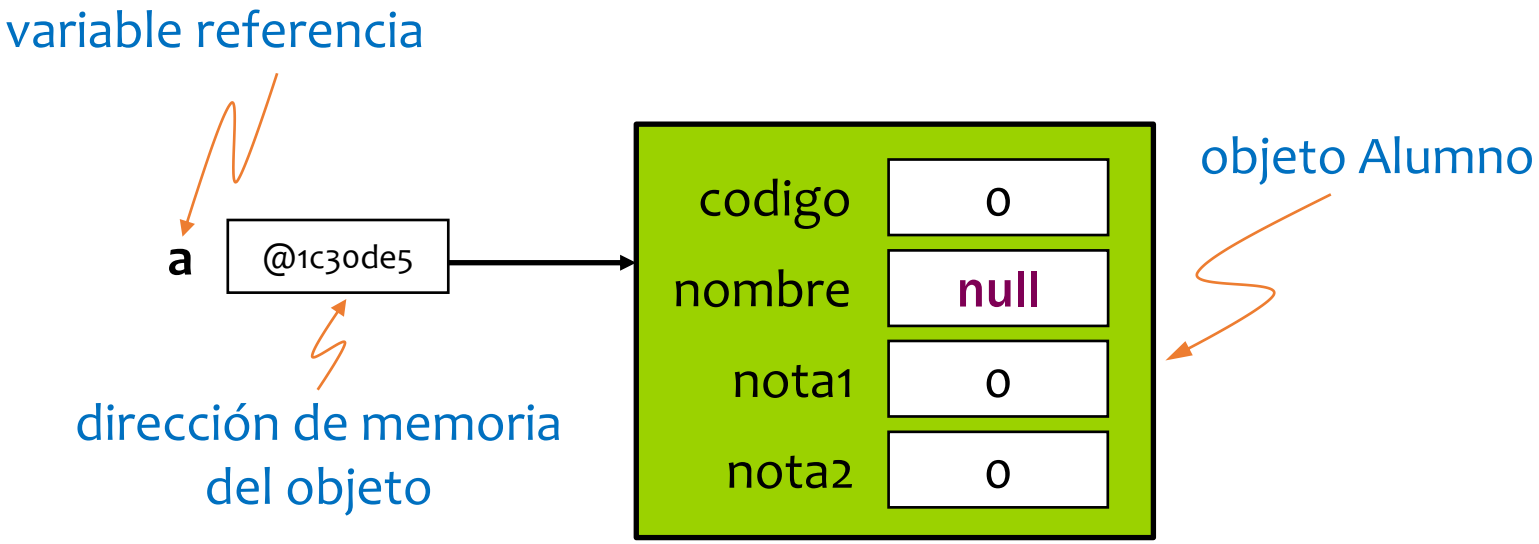
Índice

Capacidades

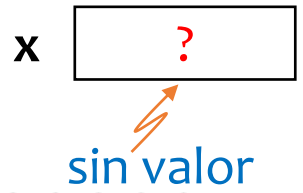
- Identifica los cambios en la forma de hacer un programa.
- Diseña clases y objetos.

1.2.1 Dirección de memoria de un objeto

```
Alumno a = new Alumno();
```

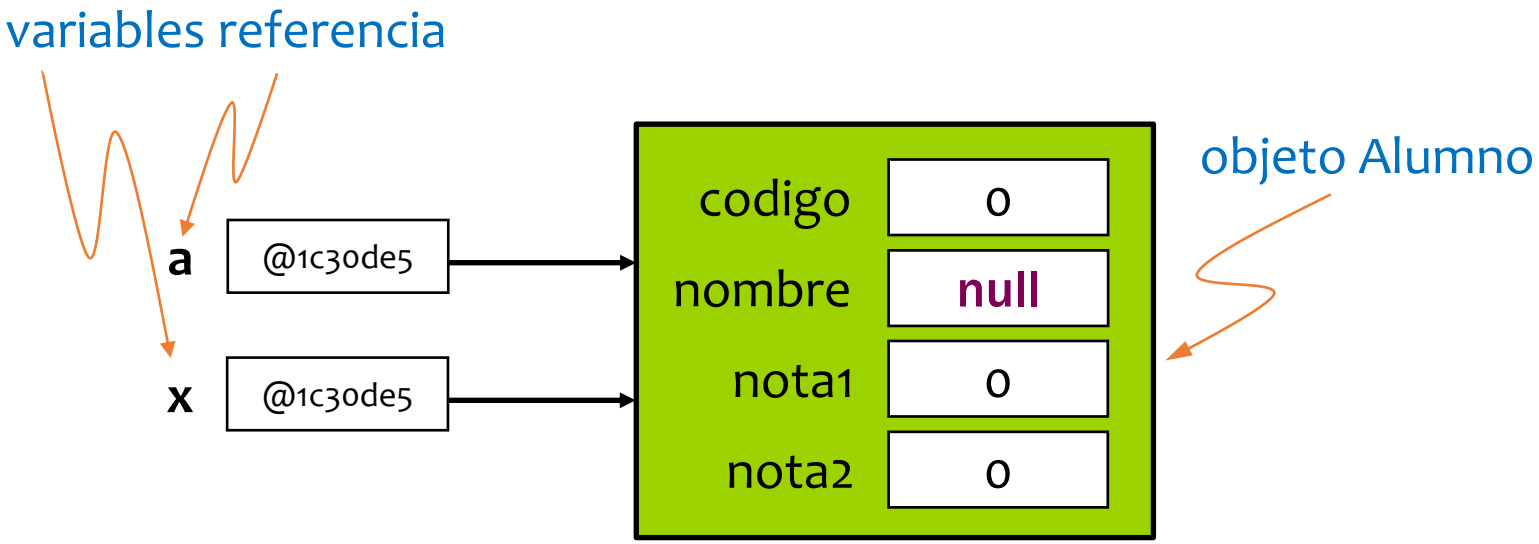



```
Alumno x;
```



1.2.2 Asignación entre referencias

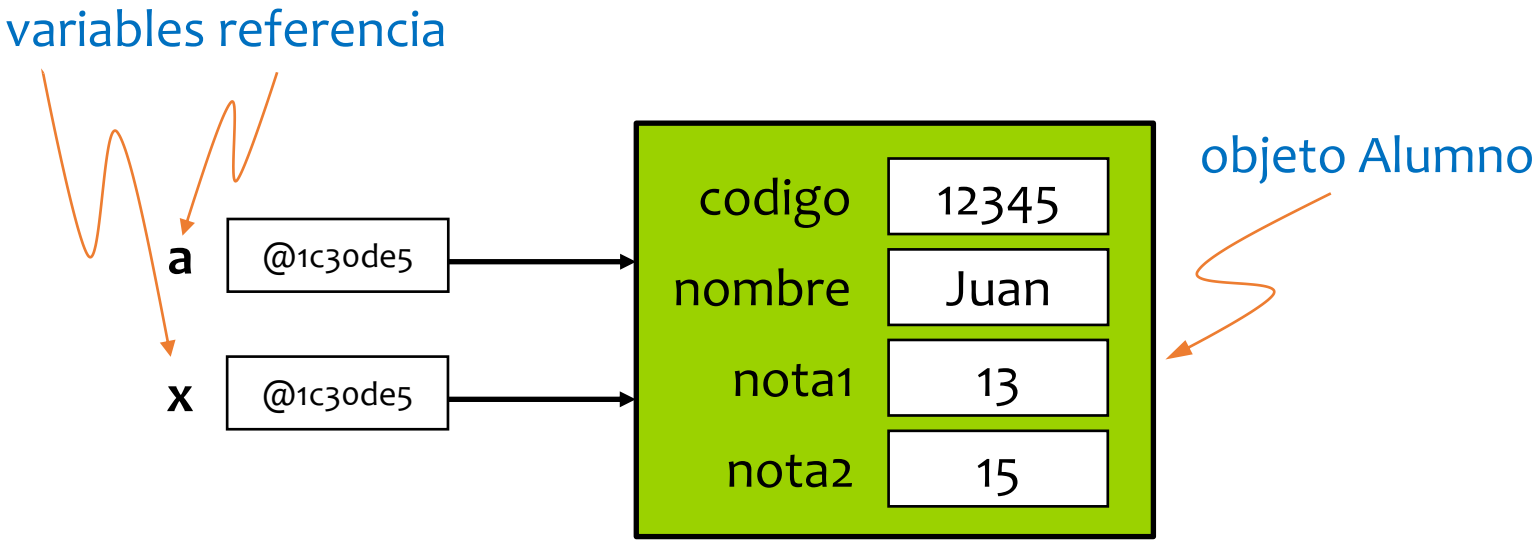
```
x = a;
```



x recibe la dirección de memoria almacenada en **a**,
luego de lo cual, tanto **x** como **a** observan o controlan al mismo objeto.

1.2.2 Asignación entre referencias

```
x.codigo = 12345; x.nombre = "Juan"; x.nota1 = 13; x.nota2 = 15;
```



A través de la variable referencia **x**
se han asignado los datos al alumno

1.2.3 Constructor

- Es un mecanismo que posee el mismo nombre de la clase pero no tiene tipo de retorno.
- Si una clase no define un constructor, Java define un constructor conocido como "constructor por defecto" que es un constructor sin parámetros que no hace nada.
- El "constructor por defecto" definido por el lenguaje no es visible en el código fuente.

```
//  Constructor por defecto
public Alumno() {

}
```

- El "constructor por defecto" puede ser definido explícitamente en el código fuente.

1.2.3 Constructor

```
public class Alumno {
    //  Atributos públicos
    public int codigo, nota1, nota2;
    public String nombre;

    //  Constructor por defecto
    public Alumno() {
    }

    //  Operaciones públicas
    public double promedio() {
        return (nota1 + nota2) / 2.0;
    }
}
```

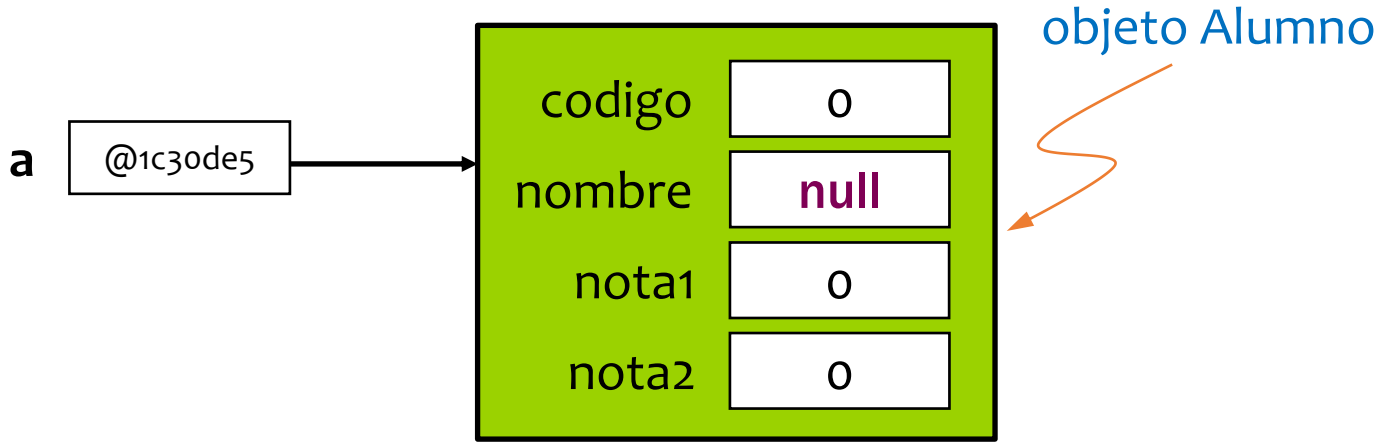
Constructor por defecto invisible en el código fuente. Si se requiere que el constructor por defecto haga algo, debe definirse en el código de manera explícita.

1.2.3 Constructor

Ejemplo

```
Alumno a = new Alumno();
```

Constructor por defecto



1.2.3 Constructor

```java
public class Alumno {
    //  Atributos públicos
    public int codigo, nota1, nota2;
    public String nombre;
    //  Constructor personalizado
    public Alumno(int cod, String nom, int n1, int n2) {
        codigo = cod;
        nombre = nom;
        nota1 = n1;
        nota2 = n2;
    }
    //  Operaciones públicas
    public double promedio() {
        return (nota1 + nota2) / 2.0;
    }
}
```

1.2.4 Creación e inicialización de objetos

Forma 1

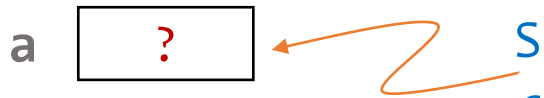
```
//  Declarar una variable referencia
NombreClase nombreObjeto;
//  Crear el objeto usando el operador new
nombreObjeto = new NombreClase(valor1, valor2, ...);
```

Ejemplo

```
//  Declarar una variable referencia
Alumno a;
//  Crear el objeto usando el operador new
a = new Alumno(12345, “Juan”, 13, 15);
```

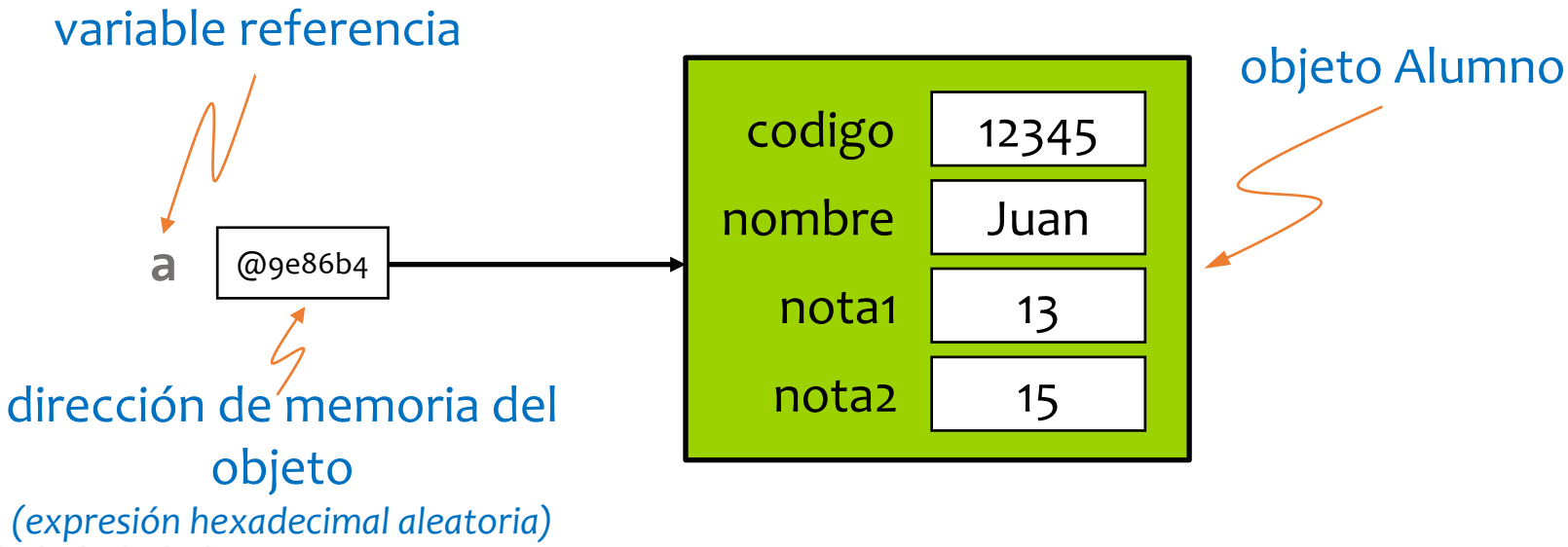
1.2.4 Creación e inicialización de objetos

```
Alumno a;
```



Si es global, se inicializa automáticamente con el valor **null**. Si es local, no contiene nada.

```
a = new Alumno(12345, "Juan", 13, 15);
```



1.2.4 Creación e inicialización de objetos

Forma 2

```
//  Declarar una variable referencia, crear e inicializar el objeto
NombreClase nombreObjeto = new NombreClase(valor1, valor2, ...);
```

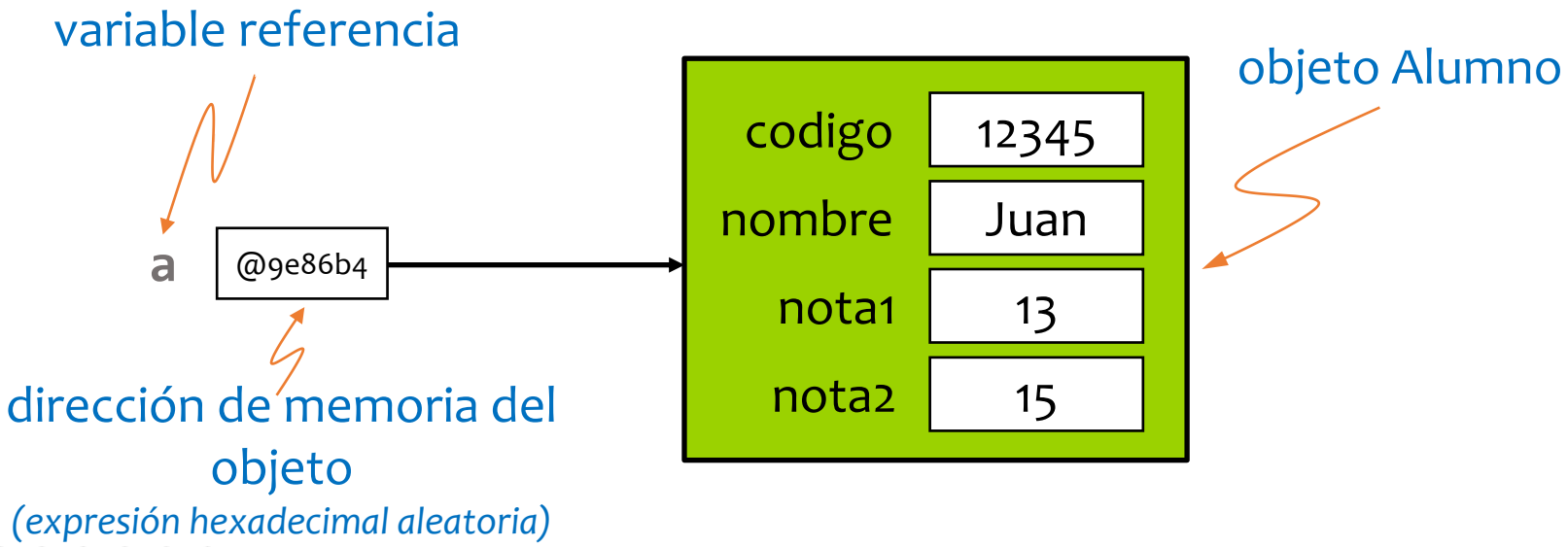
Ejemplo

```
//  Declarar una variable referencia
Alumno a = new Alumno(12345, “Juan”, 13, 15);
```

Constructor

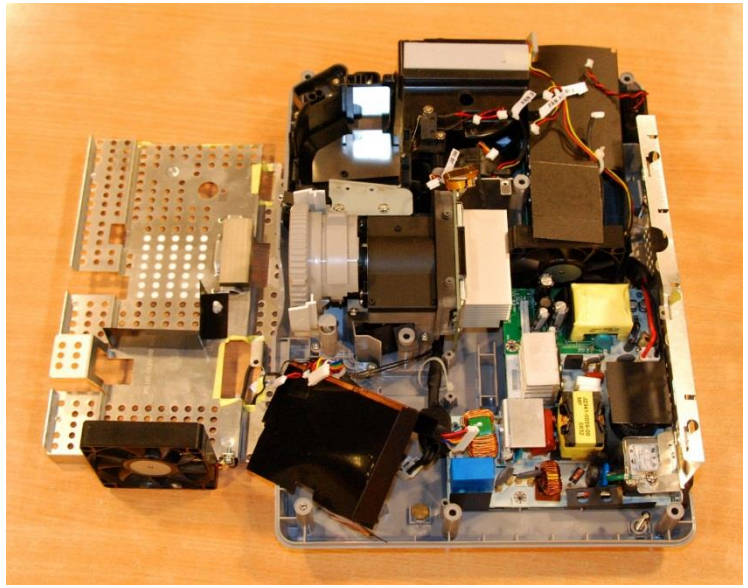
1.2.4 Creación e inicialización de objetos

```
Alumno a = new Alumno(12345, "Juan", 13, 15);
```



1.2.5 Encapsulamiento

- Es el mecanismo que consiste en ocultar los detalles internos de una clase y exponer o dar a conocer sólo los detalles que sean necesarios para el resto de clases.
- Esto permite **restringir** y **controlar** el uso de la clase para lo cual se usan los especificadores de acceso **private** y **protected** (explicado más adelante).



public

private

1.2.6 Especificador de acceso *private*

- Prohíbe el acceso externo a los *miembros* (variables y métodos) declarados de esta forma.

```
public class Alumno {
    //  Atributos privados
    private int codigo, nota1, nota2;
    private String nombre;
    //  Constructor
    public Alumno(int cod, String nom, int n1, int n2) {
        codigo = cod;
        nombre = nom;
        nota1 = n1;
        nota2 = n2;
    }
    //  Operaciones públicas
    public double promedio() {
        return (nota1 + nota2) / 2.0;
    }
}
```

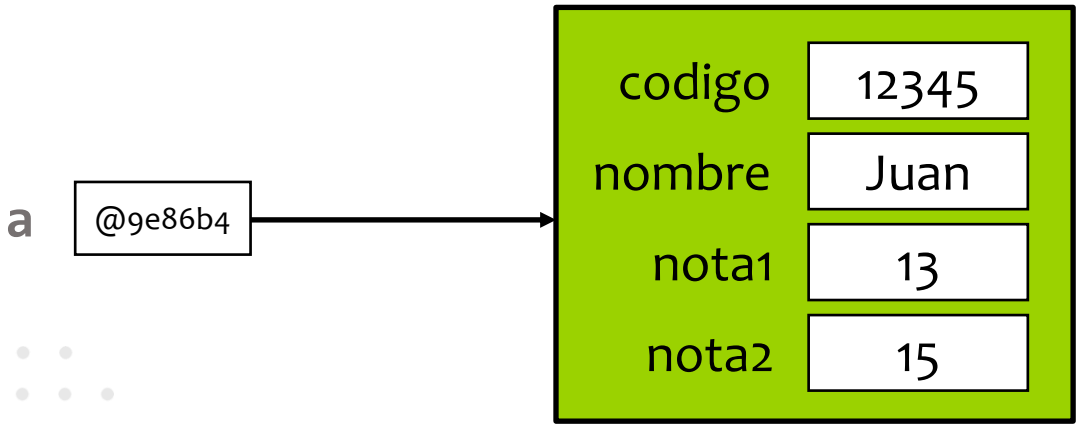
1.2.6 Especificador de acceso *private*

- Ahora, **codigo**, **nombre**, **nota1** y **nota2** son inaccesibles.

```
Alumno a = new Alumno(12345, “Juan”, 13, 15);

//  Intento de modificar nota1
//  ¡Error! nota1 es inaccesible
a.nota1 = 19;

//  Intento de obtener el valor de nota1
//  ¡Error! nota1 es inaccesible
imprimir("Nota 1 : " + a.nota1);
```



1.2.7 Métodos de acceso público: *set* / *get*

- **Modificación del valor de un atributo privado – Método *set***

  Para modificar el valor de un atributo privado, la clase debe tener un método ***public*** de tipo void que a través de un parámetro reciba un valor para ser impuesto al atributo.

```
public void setCodigo(int cod) {
    codigo = cod;
}
```

- **Obtención del valor de un atributo privado – Método *get***

  Para obtener el valor de un atributo privado, la clase debe tener un método ***public*** sin parámetros y con tipo de retorno igual al del atributo, que retorne el valor del atributo.

```
public int getCodigo() {
    return codigo;
}
```

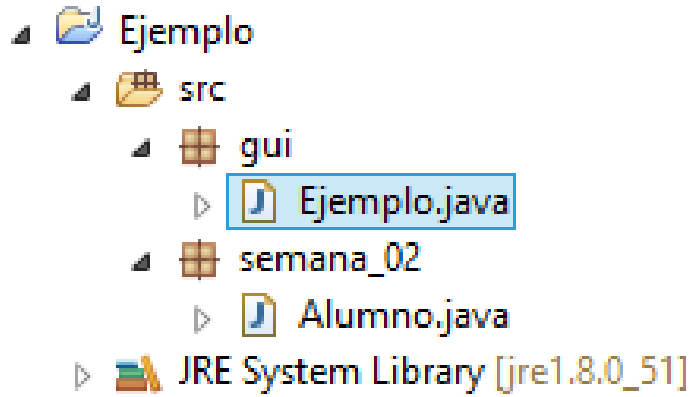
1.2.7 Métodos de acceso público: *set* / *get*

```java
public class Alumno {
    //  Atributos privados
    private int codigo, nota1, nota2;
    private String nombre;
    //  Constructor
    public Alumno(int cod, String nom, int n1, int n2) {
        codigo = cod;
        nombre = nom;
        nota1 = n1;
        nota2 = n2;
    }
    //  Métodos de acceso público: set/get
    public void setCodigo(int cod) {
        codigo = cod;
    }
    public void setNombre(String nom) {
        nombre = nom;
    }
    public void setNota1(int n1) {
        nota1 = n1;
    }
```

1.2.7 Métodos de acceso público: *set* / *get*

```
    public void setNota2(int n2) {
        nota2 = n2;
    }
    public int getCodigo() {
        return codigo;
    }
    public String getNombre() {
        return nombre;
    }
    public int getNota1() {
        return nota1;
    }
    public int getNota2() {
        return nota2;
    }
    //  Operaciones públicas
    public double promedio() {
        return (nota1 + nota2) / 2.0;
    }
}
```

1.2.8 Ejemplo



a) Implementa la clase **Alumno** en el *package* semana_02 aplicando encapsulamiento (atributos privados, constructor que inicializa a los atributos privados y métodos de acceso público set/get)

b) Coloca en la clase **Ejemplo:**
import semana_02.Alumno;

c) Declara, crea e inicializa el objeto **a** de tipo Alumno con datos fijos

d) Visualiza la información invocando a un método listado que recibe (como parámetro) en la variable referencia **x** la referencia del objeto **a**

e) Modifica luego las notas del alumno

f) Visualiza la información actualizada del objeto **a**

Conclusiones

- Un objeto es una entidad independiente que posee **atributos** y **operaciones**.
- Los **atributos** son las características del objeto que se representan mediante variables.
- Las **operaciones** son las formas de operar del objeto que se representan mediante métodos.
- Un objeto es un ejemplar creado en base a una clase.
- El constructor es un mecanismo que posee el mismo nombre de la clase y hace posible la creación de objetos.
- El encapsulamiento consiste en ocultar detalles internos de una clase y exponer sólo detalles que sean necesarios para el resto de clases.

Referencias bibliográficas

- ***Joyanes Aguilar Luis***. *Fundamentos de programación: algoritmos, estructuras de datos y objetos. Madrid, España: McGraw-Hill (005.1 JOYA/A 2021)*
- ***Lewis John***. *Estructuras de datos con Java: diseño de estructuras y algoritmos. Madrid, Pearson Educación (005.73 LEWI/E 2021)*
- ***Deitel Harvey***. *Cómo programar en Java. México, D.F.: Pearson Educación (005.133J DEIT 2021)*

GRACIAS

SEDE MIRAFLORES
Calle Diez Canseco Cdra 2 / Pasaje Tello
Miraflores – Lima
Teléfono: 633-5555

SEDE INDEPENDENCIA
Av. Carlos Izaguirre 233
Independencia – Lima
Teléfono: 633-5555

SEDE BREÑA
Av. Brasil 714 – 792
(CC La Rambla – Piso 3)
Breña – Lima
Teléfono: 633-5555

SEDE TRUJILLO
Calle Borgoño 361
Trujillo
Teléfono: (044) 60-2000

SEDE SAN JUAN DE LURIGANCHO
Av. Próceres de la Independencia 3023-3043
San Juan de Lurigancho – Lima
Teléfono: 633-5555

SEDE LIMA CENTRO
Av. Uruguay 514
Cercado – Lima
Teléfono: 419-2900

SEDE BELLAVISTA
Av. Mariscal Oscar R. Benavides 3866 – 4070
(CC Mall Aventura Plaza)
Bellavista – Callao
Teléfono: 633-5555

SEDE AREQUIPA
Av. Porongoche 500
(CC Mall Aventura Plaza)
Paucarpata - Arequipa
Teléfono: (054) 60-3535