Algoritmos y Estructura de Datos

Unidad 1: Clases y objetos

Tema 3: Miembros de clase, constantes y sobrecarga

Tema 3: Miembros de clase

Índice

1.3 Tema 3: Miembros de clase, constantes y sobrecarga

1.3.1 Referencia ***this***

1.3.2 Modificador ***static***

1.3.3 Bloque de inicialización ***static***

1.3.4 Modificador ***final***

1.3.5 Sobrecarga

1.3.6 Uso del ***this*** en sobrecarga

1.3.7 Ejemplo

Capacidades

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

1.3.1 Referencia *this*

- La palabra reservada ***this*** se utiliza para diferenciar el atributo del parámetro cuando la sintaxis coincide. Esto suele ocurrir con el constructor y los métodos ***set***.

```
public class Alumno {
    …
    //  Atributos privados
    private int codigo, nota1, nota2;
    private String nombre;
    …
    //  Constructor
    public Alumno(int codigo, String nombre, int nota1, int nota2) {
        this.codigo = codigo;
        this.nombre = nombre;
        this.nota1 = nota1;
        this.nota2 = nota2;
    }
    //  Métodos de acceso público: set/get
    public void setCodigo(int codigo) {
        this.codigo = codigo;
    }
    …
}
```

1.3.2 Modificador *static*

- Se usa para hacer que un atributo o método se convierta en único para todos los objetos.
- Este mecanismo permite manejar contadores y acumuladores desde el interior de la clase.
- Al anteponer ***static*** a un elemento entonces pasa a poder de la clase.
- Al anteponer ***static*** a un atributo, el elemento se convierte en ***variable de clase***.
- Al anteponer ***static*** a un método, el elemento se convierte en ***método de clase***.
- Si la ***variable de clase*** es pública se accede a ella por medio de la misma clase o mediante un objeto. Java sugiere solicitarlo directamente a la clase.
- Si un ***método de clase*** es público se accede a él por medio de la misma clase o mediante un objeto. Java sugiere solicitarlo directamente a la clase.
- Las ***variables de clase*** no pueden hacer uso de la referencia ***this*** porque dicha referencia es aplicable sólo a los atributos de la clase.

1.3.2 Modificador *static*

```
public class Alumno {
    …
    //  Variable de clase privada
    private static int cantidad = 0;
    …
    //  Constructor
    public Alumno(…, …, …, …) {
        …
        cantidad ++;
    }
    …
    //  Métodos públicos de clase: set/get
    public static void setCantidad(int cantidad) {
        Alumno.cantidad = cantidad;
    }
    public static int getCantidad() {
        return cantidad;
    }
    …
}
```

```
Alumno a = new Alumno(12345, "Juan", 13, 15);
Alumno b = new Alumno(67890, "Pedro", 12, 13);
Alumno c = new Alumno(38214, "María", 14, 20);
…
imprimir("Cantidad : " + Alumno.getCantidad());
```

1.3.3 Bloque de inicialización *static*

- Lo conforman una serie de códigos que se ejecutan antes de cualquier llamado a la clase o algún objeto de la clase.
- Se utiliza para inicializar variables, crear conexiones a base de datos o ejecutar cualquier código que sea prioritario antes de cualquier tipo de ejecución en la clase que se define.

```
public class Alumno {
    …
    //  Variable de clase privada
    private static int cantidad;
    …
    …
    //  Bloque de inicialización static
    static {
        cantidad = 0;
        …
    }
    …
}
```

1.3.4 Modificador *final*

- Se usa para hacer que un atributo ***static*** se convierta en constante para todos los objetos. Es decir, una vez que el atributo asume un valor no podrá ser modificado.
- Al anteponer ***static final*** a un atributo público, la variable única se convierte en ***constante pública de clase***.

```
public class Alumno {
    …
    //  Variable de clase privada
    private static int cantidad;
    //  Constante pública de clase
    public static final String ENTIDAD;
    //  Bloque de inicialización static
    static {
        cantidad = 0;
        ENTIDAD = "Cibertec";
    }
    …
}
```

1.3.5 Sobrecarga

- Una clase puede tener más de un constructor siempre y cuando no coincidan los parámetros en orden de tipología. El riesgo de aplicar sobrecarga es que más de un atributo se pueda quedar con el valor de inicialización por defecto asumido por Java.

```
public class Alumno {
    …
    //  Constructores
    public Alumno(int codigo, String nombre, int nota1, int nota2) {
        this.codigo = codigo;
        this.nombre = nombre;
        this.nota1 = nota1;
        this.nota2 = nota2;
    }
    public Alumno(int codigo, String nombre) {
        this.codigo = codigo;
        this.nombre = nombre;
    }
    public Alumno() {
    }
    …
}
```

1.3.5 Sobrecarga

- También se puede aplicar sobrecarga a los métodos.

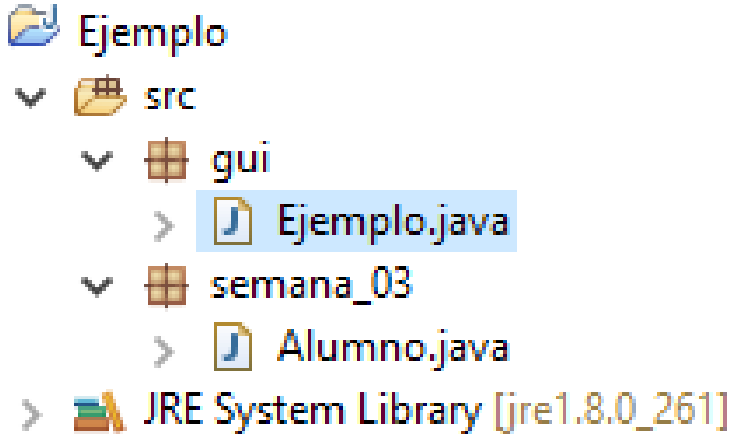
```
//  Métodos tipo void sin parámetros
void imprimir() {
    imprimir("");
}
//  Métodos tipo void con parámetros
void imprimir(String s) {
    txtS.append(s + "\n");
}
```

1.3.6 Uso del *this* en sobrecarga

- También se puede utilizar la referencia ***this*** para hacer que un constructor invoque a otro constructor. Esto evita que algún atributo quede inicializado por defecto.

```
public class Alumno {
    …
    //  Constructores
    public Alumno(int codigo, String nombre, int nota1, int nota2) {
        this.codigo = codigo;
        this.nombre = nombre;
        this.nota1 = nota1;
        this.nota2 = nota2;
    }
    public Alumno(int codigo, String nombre) {
        this(codigo, nombre, 88, 88);
    }
    public Alumno() {
        this(88888, “ninguno”);
    }
    …
}
```

1.3.7 Ejemplo



a) Implementa la clase **Alumno** en el *package* semana_03 aplicando encapsulamiento (hace uso de la referencia *this*), modificadores *static*, *final* y sobrecarga de constructores.

b) Coloca en la clase **Ejemplo**:
import semana_03.Alumno;

c) Declara, crea e inicializa los objetos de tipo Alumno **a**, **b**, **c** (con datos fijos), usando tres constructores distintos y visualiza la información de cada uno de ellos.

d) Muestra la cantidad de objetos creados y el valor de una constante.

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