# Desarrollo Avanzado de Aplicaciones I

Programación Orientada a Objetos 2



# inicio desarrollo aplicación término



# Inicio

Logro de aprendizaje - Introducción



# Logro de Aprendizaje

"Al finalizar la sesión, el participante podrá analizar los conceptos de encapsulamiento en las clases, objetos, y utilizar constructores e inicializadores de clases."



#### Introducción



- Revisión Rápida de Temas de Sesión Anterior
- Revisión de Ejercicios de Sesión Anterior
- Inquietudes y/o Preguntas
- Agenda de Sesión
  - Encapsulamiento
  - Métodos Getters y Setters
  - Métodos Constructores e Inicializadores
  - Referencia this



# Desarrollo

Desarrollo del Contenido de la Sesión



# Encapsulamiento



- ✓ Encapsulamiento es la capacidad de ocultar los detalles internos de la estructura y comportamiento de una clase y exponer o hacer visible o accesible sólo los detalles que sean necesarios para su uso.
- ✓ Este ocultamiento nos permite restringir y controlar la utilización de la clase. Restringir, porque habrá ciertos atributos y métodos privados o protegidos y controlar, porque habrá ciertos mecanismos (operaciones) para modificar el estado (atributos) de nuestra clase.
- ✓ Beneficios:
  - > Mejor control de los atributos y operaciones de la clase.
  - > Se puede tener atributos de solo lectura o de solo escritura.
  - > Mayor flexibilidad en la programación (reutilización).
  - > Mayor seguridad de los datos.

# Ejemplo de Encapsulamiento



#### ✓ Clase Persona (atributos públicos):

La edad de Carlos Gutierrez es: 150 años.

# Ejemplo de Encapsulamiento (cont.)



```
class Persona {
   private String nombre;
   private String apellido; - Aplicando Encapsulamiento
   private int edad;
   void identificarPersona(String nombre, String apellido, int edad) {
        this.nombre = nombre:
        this.apellido = apellido;
        if (edad >= 1 && edad <= 120)
            this.edad = edad:
        else
            System.out.println(x: "Edad incorrecta.");
   void mostrarDatos() {
        System.out.println("La edad de " + nombre + " " + apellido +
                " es: " + edad + " años.");
```

# Ejemplo de Encapsulamiento (cont.)



```
public class MiClase3 {
    public static void main(String[] args) {
        Persona per = new Persona();
        per.identificarPersona(nombre: "Carlos", apellido: "Gutierrez", edad: 150);
        per.mostrarDatos();
}
```

Edad incorrecta.

La edad de Carlos Gutierrez es: 0 años.

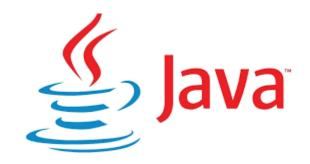


## Ejemplo de Encapsulamiento (cont.)



```
public class MiClase3 {
    public static void main(String[] args) {
        Persona per = new Persona();
        per.identificarPersona(nombre: "Carlos", apellido: "Gutierrez", edad: 50);
        per.mostrarDatos();
}
```

La edad de Carlos Gutierrez es: 50 años.



# Métodos Getters y Setters



- ✓ Los getters y setters son métodos de acceso directo a los atributos encapsulados.
- ✓ Estos métodos siempre deben ser públicos.
- ✓ Los getters (de la palabra inglés get obtener) sirve para obtener (recuperar o acceder) y utilizar el valor ya asignado de un atributo. Se utiliza sin parámetros.
- ✓ Los setters (de la palabra ingles set establecer) sirven para asignar un valor a un atributo, nunca retorna algún valor (siempre es void) y debe tener como parámetro el valor que se asignará al atributo.
- ✓ Por convención, el nombre del método debe anteponer la palabra get o set al nombre del atributo utilizando la notación camello (En caso del tipo de dato boolean se antepone is en lugar de get).

## Sintaxis de Métodos Getters y Setters

this.atributo = nombreParametro;



```
    ✓ Método getter:

            public tipoDevuelto getNombreAtributo() {
            return atributo;
            }

    ✓ Método setter:
```

public void setNombreAtributo(tipoDato nombreParametro) {

**S** Java

# Ejemplo de Métodos Getters y Setters



✓ Recordemos nuestra clase Persona:

```
class Persona {
    String nombre;
    String apellido;
    int edad:
    void mostrarDatos() {
        System.out.println("La edad de " +
                nombre + " " + apellido +
                " es: " + edad + " años.");
```



# Ejemplo de Métodos Getters y Setters



✓ Le ponemos el modificador de acceso privado a las variables:

```
private String nombre;
private String apellido;
private int edad;
```

✓ Creamos los métodos getters y setters para los atributos:

```
public String getNombre() {
    return nombre;
}
public String getApellido() {
    return apellido;
}
public int getEdad() {
    return edad;
}
```

```
public void setNombre(String nombre) {
    this.nombre = nombre;
}

public void setApellido(String apellido) {
    this.apellido = apellido;
}

public void setEdad(int edad) {
    this.edad = edad;
}
```

# Ejemplo de Métodos Getters y Setters



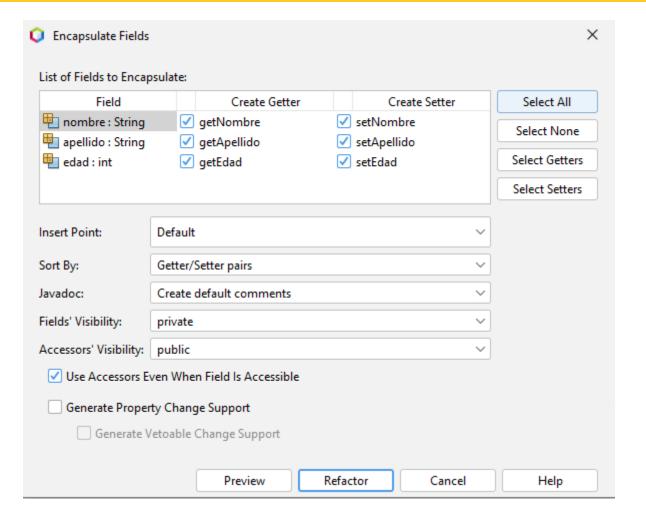
✓ Modificamos el código donde usamos la clase Persona:

```
public class MiClase3 {
    public static void main(String[] args) {
        Persona per = new Persona();
        per.setNombre(nombre: "Carlos");
        per.setApellido(apellido: "Gutierrez");
        per.setEdad(edad:50);
        System.out.println("La edad de " + per.getNombre() + " " +
                per.getApellido() + " es: " + per.getEdad() + " años.");
```

# Usando NetBeans para Encapsular



✓ A través del menú "Refactor\Encapsulate Fields..." se puede generar los getters y setters de una clase automáticamente.



#### Métodos Constructores e Inicializadores



- ✓ Un constructor es un método especial de una clase que se llama automáticamente siempre que se declara un objeto de esa clase.
- ✓ La principal misión del constructor es reservar memoria e inicializar las variables miembros de la clase.
- ✓ Si una clase no define un constructor, Java crea un constructor por defecto que no tiene parámetros.
- ✓ Cuando se crea un objeto en java se realiza las siguientes operaciones de forma automática:
  - > Se asigna memoria para el objeto.
  - Se inicializa los atributos de estos objetos con los valores predeterminados por el sistema.
  - > Se llama al constructor de la clase que puede ser uno **Java** entre varios.

#### Métodos Constructores e Inicializadores (cont.)



- ✓ Los métodos constructores tienen las siguientes características:
  - > Debe tener el mismo nombre que la clase a la que pertenece.
  - En una clase puede haber varios constructores con el mismo nombre y con distinta cantidad o tipos de argumentos. Es decir permite sobrecarga.
  - > Los constructores no se heredan.
  - > Un constructor no puede devolver ningún valor incluyendo el void.
  - Un constructor debería declararse público, para que pueda ser invocado desde cualquier parte donde se desee crear un objeto de su clase.
  - El constructor es el primer método que se ejecuta, se ejecuta en forma automática y se ejecuta una única vez.

# Ejemplos de Método Constructor



√ Método constructor declarado de forma explícita:

```
class Persona {
   private String nombre;
   private String apellido;
   private int edad;
    public Persona() {
       this.nombre = "----";
       this.apellido = "----";  Constructor que inicializa los atributos
        this.edad = 18;
   void mostrarDatos() {...5 lines ]
    public String getNombre() {...3 lines }
    public String getApellido() {...3 lines }
   public int getEdad() {...3 lines }
   public void setNombre(String nombre) {...3 lines }
    public void setApellido(String apellido) {...3 lines }
   public void setEdad(int edad) {...3 lines }
```



## Ejemplos de Método Constructor (cont.)



#### √ Utilización del método constructor explícito:

```
public class MiClase3 {
    public static void main(String[] args) {
        Persona per = new Persona();
       per.mostrarDatos();
       per.setNombre(nombre: "Carlos");
       per.setApellido(apellido: "Gutierrez");
       per.setEdad(edad:50);
       per.mostrarDatos();
La edad de
La edad de Carlos Gutierrez es: 50 años.
```

Uso del constructor

Valores asignados a los atributos en el constructor

Valores asignados a los atributos en el método main



## Ejemplos de Método Constructor (cont.)



#### ✓ Método constructor con parámetros:

```
class Persona {
   private String nombre;
   private String apellido;
   private int edad;
   public Persona() { ← Método constructor explícito sin parámetros
       this.nombre = "----";
                                      Método constructor explícito con parámetros
       this.apellido = "----";
       this.edad = 18:
   public Persona(String nombre, String apellido, int edad) {
       this.nombre = nombre:
       this.apellido = apellido;
       this.edad = edad:
```



## Ejemplos de Método Constructor (cont.)



#### √ Utilización del método constructor con parámetros:

```
public class MiClase3 {
    public static void main(String[] args) {
        Persona per = new Persona(nombre: "Carlos", apellido: "Gutierrez", edad: 50);
        per.mostrarDatos();
        Persona otra = new Persona();
        otra.mostrarDatos();
    }
        Constructor sin parámetros
}
```

```
La edad de Carlos Gutierrez es: 50 años.
La edad de ----- es: 18 años.
```



#### Referencia this



- ✓ La palabra reservada "this" la utilizamos para llamar a un constructor desde otro constructor de la misma clase.
- ✓ La llamada a otro constructor con la palabra reservada "this" debe ser la primera línea del constructor.
- ✓ El orden no importa en el encadenamiento de constructores.
- ✓ La clase debe tener al menos un constructor que no utilice la palabra reservada "this".
- ✓ El constructor al que se invoca debe soportar el conjunto de parámetros que le pasamos.

## Ejemplos de Referencia this



```
class Persona {
   private String nombre;
   private String apellido;
   private int edad;
    public Persona() {
       this.nombre = "----";
       this.apellido = "----";
       this.edad = 18;
    public Persona(String nombre, String apellido, int edad) {
       this(); 
Llamada al constructor sin parámetros
       if (edad>=18 && edad<=120) {
           this.nombre = nombre;
           this.apellido = apellido;
           this.edad = edad:
```



# Ejemplos de Referencia this (cont.)



```
public class MiClase3 {
    public static void main(String[] args) {
        Persona per = new Persona(nombre: "Carlos", apellido: "Gutierrez", edad: 150));
        per.mostrarDatos();
    }
}
La edad de ----- es: 18 años.
```

```
public class MiClase3 {
    public static void main(String[] args) {
        Persona per = new Persona(nombre: "Carlos", apellido: "Gutierrez", edai: 50);
        per.mostrarDatos();
    }
}
La edad de Carlos Gutierrez es: 50 años.
```

# Aplicación

Revisar ejemplos y realizar ejercicios prácticos



# Término

Indicaciones generales y/o Resumen de Sesión



#### Resumen de Sesión



- Encapsulamiento
- Métodos Getters y Setters
- Métodos Constructores e Inicializadores
- Referencia this
- Ejemplos y ejercicios



# GRACIAS