Programación orientada a objetos Individual - POO - Unidad 2
Actividad: Taller sobre Constructores en java
Tutor: JOHN CARLOS ARRIETA ARRIETA DOCENTE

Taller: Constructores en Java

Objetivo del Taller:

En este taller, los estudiantes aprenderán a comprender, diseñar y utilizar correctamente los constructores en Java. Se abordarán los tipos de constructores, su sintaxis y cómo pueden ser utilizados para inicializar objetos con diferentes valores.

Temario:

1. **Introducción a los Constructores**
    - Definición y propósito de un constructor en Java.
    - Diferencias entre un constructor y un método.
2. **Tipos de Constructores en Java**
    - Constructor por defecto.
    - Constructor parametrizado.
    - Sobrecarga de constructores.
3. **Uso de `this()` y `super()`**
    - Llamada a otro constructor dentro de la misma clase.
    - Llamada al constructor de la clase padre (herencia).
4. **Buenas Prácticas con Constructores**
    - Encapsulamiento.
    - Validación de datos en constructores.
5. **Ejercicios Prácticos**

---

1. Introducción a los Constructores

Los constructores en Java son bloques de código que se ejecutan cuando se crea una instancia de una clase. Su principal función es inicializar el objeto con valores iniciales.

Características de un Constructor:

- No tiene un tipo de retorno (ni siquiera `void`).
- Tiene el mismo nombre que la clase.
- Puede ser sobrecargado para aceptar diferentes conjuntos de parámetros.

Ejemplo de Constructor Simple:

```
public class Persona {
    String nombre;

    // Constructor
    public Persona() {
        nombre = "Sin nombre";
    }
}
```

2. Tipos de Constructores en Java

2.1. Constructor por defecto

Es un constructor sin parámetros que inicializa las propiedades con valores predeterminados.

```
public class Coche {
    String marca;

    // Constructor por defecto
    public Coche() {
        marca = "Desconocida";
    }
}
```

2.2. Constructor Parametrizado

Se usa para inicializar propiedades con valores proporcionados por el usuario.

```java
public class Coche {
    String marca;

    // Constructor parametrizado
    public Coche(String marca) {
        this.marca = marca;
    }
}
```

2.3. Sobrecarga de Constructores

Es posible definir múltiples constructores dentro de la misma clase, cada uno con un conjunto diferente de parámetros.

```java
public class Coche {
    String marca;
    String modelo;

    // Constructor por defecto
    public Coche() {
        this.marca = "Desconocida";
    }

    // Constructor parametrizado
    public Coche(String marca) {
        this.marca = marca;
    }

    // Constructor sobrecargado con dos parámetros
    public Coche(String marca, String modelo) {
        this.marca = marca;
        this.modelo = modelo;
    }
}
```

3. Uso de `this()` y `super()`

3.1. `this()`: Llamada a otro constructor en la misma clase

```java
public class Coche {
    String marca;
    String modelo;

    // Constructor con un parámetro
    public Coche(String marca) {
        this.marca = marca;
    }

    // Llamada al constructor anterior usando `this()`
    public Coche(String marca, String modelo) {
        this(marca); // Llama al constructor de un solo parámetro
        this.modelo = modelo;
    }
}
```

3.2. `super()`: Llamada al constructor de la clase padre

Se utiliza en herencia para llamar al constructor de la clase base.

```java
public class Vehiculo {
    String tipo;

    public Vehiculo(String tipo) {
        this.tipo = tipo;
```

```
    }
}

public class Coche extends Vehiculo {
    String marca;

    // Llama al constructor de la clase base usando `super()`
    public Coche(String tipo, String marca) {
        super(tipo);
        this.marca = marca;
    }
}
```

4. Buenas Prácticas con Constructores

- **Encapsulamiento:** Utilizar los constructores junto con los métodos `get` y `set` para controlar el acceso a las propiedades.
- **Validación de datos:** Validar los datos en los constructores para evitar asignar valores inconsistentes a las propiedades del objeto.

5. Ejercicios Prácticos

Ejercicio 1: Crear una clase `Libro`

Crea una clase `Libro` con las siguientes propiedades: `titulo`, `autor`, `numeroPaginas`. Define:

1. Un constructor por defecto que asigne valores predeterminados.
2. Un constructor parametrizado que permita asignar valores específicos.
3. Métodos para mostrar los detalles del libro.

Ejercicio 2: Crear una clase `CuentaBancaria`

Define una clase `CuentaBancaria` con las propiedades: `numeroCuenta`, `saldo` y `tipoCuenta`. Define:

1. Un constructor por defecto.
2. Un constructor parametrizado con dos parámetros (`numeroCuenta` y `tipoCuenta`).
3. Un constructor sobrecargado con tres parámetros (`numeroCuenta`, `saldo`, y `tipoCuenta`).

Ejercicio 3: Crear una clase `Estudiante`

Crea una clase `Estudiante` que contenga las propiedades: `nombre`, `edad`, y `curso`. Define:

1. Un constructor por defecto que asigne valores genéricos.
2. Un constructor que acepte como parámetros el nombre y la edad.
3. Un constructor que acepte todos los parámetros y utilice `this()` para llamar a otro constructor.

Ejercicio Final: Proyecto Integrador

Desarrolla una aplicación Java que contenga las clases que definiste en los ejercicios anteriores (`Libro`, `CuentaBancaria`, y `Estudiante`). Crea un archivo `main` que:

1. Cree al menos un objeto de cada clase utilizando diferentes constructores.
2. Permita ingresar datos por consola para inicializar los objetos.
3. Muestre los detalles de los objetos utilizando el método `toString()`.

Conclusión del Taller:

Este taller ha proporcionado una base sólida sobre el uso de constructores en Java, desde los conceptos básicos hasta las mejores prácticas. La habilidad de crear y utilizar correctamente los constructores es fundamental para la creación de objetos coherentes y seguros en aplicaciones orientadas a objetos.