ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

PROGRAMACION ORIENTADA A OBJETOS

ASIGNATURA:	POO
PROFESOR:	Ing. Yadira Franco R
PERÍODO ACADÉMICO:	2025-B

TALLER GRUPAL

CONSULTA

1. Qué es un método

Definición: Es un bloque de código que realiza una tarea especifica y puede ser realizado en diferentes partes de un programa. En Java, los métodos permiten dividir el programa en partes más pequeñas y manejables.

Su propósito principal es encapsular la lógica o comportamiento que puede ser invocado varias veces.

Una ventaja es: reutilización del código

Estructura básica en Java

Modificador tipoDeRetorno nombreDelMetodo(parámetros) {

```
        // cuerpo del metodo
        // instrucciones a ejecutar
        Return valor; // si el tipo de retorno no es void
    }
```

EJ:

```
Public static void saludar(String nombre){
System.out.println("¡Hola, " + nombre + "! Bienvenido.");
    }
```

Diferencia entre método y bloque de código

Características	**Método**	**Bloque de código**
Definición	Función con nombre que puede ser invocada	Conjunto de instrucciones sin nombre
Reutilizable	Sí, puedes llamarlo varias veces.	No, se ejecuta una sola vez
Parámetros	Puede recibir parámetros	No recibe parámetros directamente como una función
Retorno	Puede devolver un valor con return a excepción de void	No devuelve un valor directamente, mas bien solo ejecuta acciones
Ubicación	Dentro de una clase	Dentro de métodos, constructores o bloques estáticos

2. **Qué es una función**

- **Definición general (programación estructurada):** Es un bloque de código independiente que realiza una tarea especifica que suele tener:

  Nombre, puede recibir parámetros, puede devolver un valor y se puede invocar desde varias partes diferentes de un programa.

- **Diferencias entre función y método en programación orientada a objetos**

Características	**Función(estructurada)**	**Método (POO)**
Contexto	Independiente del objeto.	Asociado a una clase u objeto.
Ubicación	Fuera de clase.	Dentro de clases.

	No puede acceder a atributos de objetos	Puede acceder a atributos y otros métodos
Reutilización	Reutilizable en cualquier parte del programa	Reutilizable pero solo en el contexto de la clase.
Encapsulamiento	No aplica directamente	Parte del encapsulamiento de POO

- Ejemplos sencillos

  Función en Python (estructurada):

  def saludar(nombre):

  return f"Hola, {nombre}"

  Metodo en Java

  Public class Persona {

  String nombre;

  Públic void saludar (){

  System.out.print("Hola soy, "+nombre);

  }

  }

3. **Estructura de una función o método**

- Palabras clave (public, static, tipo de retorno, nombre, parámetros, return)

  EJ de función o método:

  Public static int sumar(int a, int b) {

  Return a + b;

  }

- **Ejemplo explicado línea por línea**

  public: Modificador de acceso, que indica que el método puede ser usado desde cualquier clase.

  static: No requiere crear un objeto para usar el método. Pertenece a su clase directamente.

  Int: Tipo de retorno: este método devuelve un numero o entero.

  nombre (sumar): Nombre del método: Identifica a la función. Debe ser algo descriptivo o relacionado.

Parámetros (int a, int b): Se recibe como datos de entrada.

{ ... } : Cuerpo del método: Contiene desde donde inician las instrucciones hasta donde terminan, delimitan un método o función.

return: Sentencia de retorno, que devuelve un valor en este caso resultado de a + b

Y si fuera “void”, lo que significa que no devuelve ningún valor, solo ejecuta una acción.

4. **Encapsular métodos y atributos**

- **Concepto de encapsulamiento:** Consiste en ocultar detalles internos de una clase y solo mostrar lo necesario a través de métodos públicos. Esto ayuda a proteger los datos y permite controlar como se accede o modifica el estado de un objeto.
- **Modificadores de acceso (private, public, protected)**

  Private: Solo dentro de la misma clase.

  Pubic: Se puede acceder desde cualquier clase

  Protected: Desde la misma clase, subclases y mismo paquete.

  (vacio): Solo dentro del mismo paquete

- **Ejemplo con atributos privados y métodos públicos**

```
public class Persona {

    // Atributos privados (ocultos)
    private String nombre;
    private int edad;

    // Método Constructor
    public Persona(String nombre, int edad) {
        this.nombre = nombre;
        setEdad(edad); // Usamos el setter para validar
    }

    // Método público para obtener el nombre o Getter
    public String getNombre() {
```

```
        return nombre;
    }

    // Método público para cambiar el nombre
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    // Getter de edad
    public int getEdad() {
        return edad;
    }

    // Setter con validación
    public void setEdad(int edad) {
        if (edad >= 0) {
            this.edad = edad;
        } else {
            System.out.println("Edad no válida");
        }
    }
}
```

5. **Cuándo utilizamos static**

Un método o atributo static pertenece a la clase, que se puede usar sin crear un objeto.

- **Diferencia entre métodos estáticos y de instancia**

Característica	Metodo statico (static)	Metodo de instancia (no static)
Pertenece a	La clase	Un objeto creado a partir de una clase.
Acceso	Clase.metodo()	objeto.metodo()

Puede usar atributos no estáticos	No puede acceder directamente.	Si puede acceder.

- **Cuando conviene usar static (por ejemplo, en utilidades o cálculos globales)**

  Funciones utilitarias: Como cálculos matemáticos, validaciones o conversiones.

  Constantes Globales: Como (public static final double PI = 3.1459;)

  - Cuando no necesitas acceder a atributos de una instancia.

  - O también dentro del método main, porque no hay objetos creados aún.

- Ejemplo comparativo

```
public static int sumar(int a, int b) {
     return a + b;
     }
     public static boolean esPar(int numero) {
  return numero % 2 == 0;
}
```

public static final double PI = 3.1459;)

6. **Paso de parámetros y paso por valor o referencia**

- Explicación de ambos conceptos

  Paso de parámetros: Cuando llamas a un método, puedes enviarle valores o referencias como argumentos.

  Paso por valor

  - Se copia el valor del argumento.
  - El método trabaja con una copia, no con el original.
  - Cambios dentro del método no afectan al valor externo.

  Paso por referencia

  - Se pasa la dirección de memoria del objeto.
  - El método pude modificar el objeto original.
  - Cambios dentro del método si afectan al valor externo

- **Cómo funciona en Java (paso por valor de referencia)**

  Java siempre usar paso por valor, pero se distingue por:

  Tipo de dato

Primitivos (int, double, boolean), se copia el valor directamente y no puede modificar el original.

Objetos (String, Persona, ArrayList), se copia la referencia del objeto y se puede modificar el objeto, pero no cambiar la referencia.

- Ejemplo con variables y objetos

Ejemplo con variable primitiva:

```
public class Ejemplo {

    public static void cambiarValor(int x) {
        x = 99;
    }

    public static void main(String[] args) {
        int numero = 10;
        cambiarValor(numero);
        System.out.println(numero);  // Imprime 10
    }
}
```

Aquí, aunque “x” se cambia dentro del método, numero no se ve afectado porque se pasó por valor.

Ejemplo con objeto:

```
public class Persona {
    String nombre;
}

public class Ejemplo {

    public static void cambiarNombre(Persona p) {
        p.nombre = "Kevin";
```

```
    }

    public static void main(String[] args) {
        Persona persona = new Persona();
        persona.nombre = "Luis";
        cambiarNombre(persona);
        System.out.println(persona.nombre);  // Imprime "Kevin"
    }
}
```

Aquí se pasa la referencia al objeto, así que si se modifca el estado interno.

Lo que no se puede hacer es reemplazar la referencia desde dentro del método.

7. Cuándo utilizamos `return`

- Tipos de valores devueltos

Una función usa return para devolver un valor y finalizar su ejecución. La misma puede retornar números (int, float, double), texto (String), booleanos (true, false), objetos (las instancias de clase), listas, arreglos y colecciones.

- Qué sucede si no se usa `return`

Si en la función no se usa return hay dos casos, uno donde la función sea un tipo void lo cual no presentaría problema ya que esta no retorna ningún valor. Por otra parte, están las funciones String, int, etc. Estas funciones si retornan valores por lo que al no poner un return el código simplemente no compilaría.

- Ejemplo con retorno de valores numéricos y texto

CÓDIGO

```
public class Pregunta7 {
    int numero = 12;
    String texto = "hola";
    public int getNumero() {
        return numero;
    }
    public String getTexto() {
        return texto;
    }
    public static void main(String[] args) {
        Pregunta7 nuevo = new Pregunta7();
        System.out.println(nuevo.getNumero());
        System.out.println(nuevo.getTexto());
    }
}
```

🔷 8. Sobrecarga de métodos (Overloading)

- Concepto

La sobrecarga de métodos o overloading se refiere a la colocación de varios métodos con el mismo nombre, pero con diferentes parámetros lo cual permite que existan caso contrario estas pueden generar un error.

- Reglas básicas (mismo nombre, distintos parámetros)
- Ejemplo práctico: sistema de mensajería personalizada

CÓDIGO

```
public class Pregunta8 {

    String nombre;

    public Pregunta8(String nombre){

        this.nombre=nombre;

    }

    public void saludar(){

        System.out.println("Hola "+nombre);

    }

    public void saludar(String ciudad){

        System.out.println("Hola "+nombre+" eres de "+ciudad);

    }

    public static void main(String[] args) {

        Pregunta8 nuevo = new Pregunta8("Gabriel");

        nuevo.saludar();

        nuevo.saludar("Quito");

    }

}
```

🔷 9. Identificar el llamado correcto: función o método

- Cómo se invoca un método estático vs. uno de instancia

La invocación de un método viene de la clase, el método estático es aquel que se encuentra en la clase, mientras que el método de la instancia pertenece a un objeto creado a partir de la clase.

- Uso de objetos e instancias
- Ejemplo: `Math.sqrt()` vs. `objeto.metodo()`

CÓDIGO

```
public class Pregunta9 {

    public static void main(String[] args) {

        double raiz = Math.sqrt(25);

        String texto = "hola";

        int longitud = texto.length();

        System.out.println("Raiz: "+raiz+"\nLongitud del texto: "+ longitud);

    }

}
```

PREGUNTAS

¿Qué diferencia hay entre una función que existe sola y un método dentro de una clase?

Las funciones son típicamente usadas en lenguajes como C++, Python y JavaScript. En estos lenguajes se permiten la existencia de funciones no ligadas a una clase. Por otra parte, en Java los métodos no pueden existir fuera de una clase ya que como Java es un lenguaje orientado a objetos, estos métodos van ligados a una clase por el concepto de herencia de clase, etc.

¿Por qué en Java no se puede ejecutar un método sin crear antes un objeto?

Esto es principalmente por que los métodos no estáticos no pertenecen a una estancia específica de clase y como el lenguaje de java es orientado a objetos, estos están sujetos a las clases. Por lo tanto, solo los métodos estáticos tienen la facilidad de ejecutarse sin crear el objeto, como se muestra en el ejercicio 9.

¿Cómo se llama correctamente un método de instancia y uno estático?

Para llamar de manera correcta a cada uno de los métodos se tiene que tener en cuenta la siguiente sintaxis:

Estático

Clase.metodo();

De instancia

Objeto.metodo();

En Java puedes invocar un método desde otro método.? eJEMPLO

CÓDIGO

```
public class Invocacion {
   String nombre;
   public Invocacion(String nombre){
      this.nombre=nombre;
   }
   public String saludo(){
      return "Hola "+ nombre;
   }
   public void edad(int edad){
      String c = saludo();
      System.out.println(c+" tienes "+edad+" anios");
   }

   public static void main(String[] args) {
      Invocacion nuevo = new Invocacion("Gabriel");
      nuevo.edad(21);
   }
}
```

Link de github

https://github.com/gabrielt007/DeberesPOO.git