UNIVERSIDAD TECNOLÓGICA DE PANAMÁ FACULTAD DE

INGENIERÍA DE SISTEMAS COMPUTACIONALES

LICENCIATURA EN CIBERSEGURIDAD

PROGRAMACION I

INVESTIGACION 1

(JAVA)

PREPARADO POR:

DANELIS CABALLERO, 4-772-754

A CONSIDERACIÓN DE

IBARRA NAPOLEON

GRUPO

2S3111

FECHA

20/8/2025

Procedimiento:

1. De manera individual, realizar la asignación.

2. Utilizando la herramienta Internet, una técnica conocida (desarrollo de investigación), resuelva los puntos solicitado.

3. Entregar el trabajo en formato digital (.PDF) en la plataforma utilizada.

TEMA:

1.3. Construcción de clase (JAVA)

1.3.1 Miembros de una clase

1.3.2 Modificadores de acceso

1.3.3. Otras opciones

Procedimiento: Utilice una técnica conocida para desarrollar las siguientes pregunta en base al

tema propuesto:

1. ¿Cuál es su concepto?

2. ¿Importancia y/o relevancia?

3. ¿Ventajas y Desventajas? Mínimo 3, Máximo 5 ítem.

4. Ejemplo de cada subpunto.

INVESTIGACIÓN 1

Construcción de clase JAVA

Una clase en Java es una plantilla o modelo que define las propiedades y comportamientos de un objeto. Es el núcleo de la programación orientada a objetos (POO). Se refiere al proceso de diseñar y codificar una estructura que define un tipo de objeto.

Importancia

- Organización del código: Esto mejora la claridad del código, facilita su lectura y reduce la posibilidad de errores.
- Reutilización y modularidad: Una clase bien diseñada puede ser utilizada para crear múltiples objetos con diferentes valores.
- Escalabilidad y mantenimiento: El uso de clases facilita la expansión del sistema sin afectar otras partes del programa. También permite localizar y corregir errores de manera más rápida y efectiva, lo que mejora el mantenimiento del software.
- Encapsulamiento y seguridad: Las clases permiten proteger los datos internos mediante modificadores de acceso. Esto garantiza que los datos solo puedan ser manipulados de forma controlada, aumentando la seguridad y robustez del sistema.
- Abstracción y modelado del mundo real: Las clases permiten representar entidades reales o conceptuales dentro del programa.

Ventajas

- Facilita la organización lógica del código
- Promueve la reutilización y modularidad
- Mejora la escalabilidad y el mantenimiento
- Permite el encapsulamiento y control de acceso
- Favorece la abstracción y el modelado de sistemas

Desventajas

- Requiere conocimientos previos en programación orientada a objetos
- Puede aumentar la complejidad del diseño
- Mal uso puede generar estructuras innecesarias
- Dificulta la depuración si no se estructura bien
- Puede implicar mayor tiempo de desarrollo inicial

Ejemplo:

```
public class Libro {

String titulo;

String autor;

void mostrarInfo() {

System.out.println(titulo + " por " + autor);
```

Miembros de una clase

Los miembros de una clase son los elementos que definen su estructura y comportamiento. Incluyen:

- Atributos (o campos): variables que representan el estado de un objeto.
- Métodos: funciones que definen el comportamiento del objeto.
- Constructores: métodos especiales que inicializan objetos.
- Bloques y clases internas: estructuras adicionales que pueden existir dentro de una clase.

Importancia

- Aplicación de principios de diseño orientado a objetos: Los miembros son esenciales para implementar conceptos como abstracción, herencia y polimorfismo, que permiten construir sistemas flexibles, extensibles y coherentes.
- Organización y legibilidad del sistema: Agrupar funcionalidades relacionadas en clases con miembros bien definidos, mejora la claridad del código, facilita su mantenimiento y promueve buenas prácticas de desarrollo.

Ventajas

- Encapsulamiento
- Reutilización de código
- Organización estructurada

Desventajas

- Complejidad inicial
- Sobrecarga de memoria

Ejemplo:

```
public class Producto {
String nombre;
double precio;
int cantidad;

double calcularTotal() {
return precio * cantidad;
  {
  {
```

Modificadores de acceso

Determinan el nivel de visibilidad o accesibilidad de los miembros de una clase (atributos, métodos, constructores, etc.) desde otras clases o paquetes.

En Java, los principales son:

- public: acceso desde cualquier clase.
- private: acceso solo dentro de la misma clase.
- protected: acceso desde la misma clase, subclases y clases del mismo paquete.

- (sin modificador): acceso por defecto, solo dentro del mismo paquete.

Importancia

- Permiten controlar el acceso a los datos y comportamientos de una clase, protegiendo la integridad del sistema.
- Ayudan a reducir errores al limitar el uso indebido de atributos o métodos sensibles.
- Mejoran la organización y seguridad del código, especialmente en sistemas grandes y colaborativos.
- Facilitan la mantenibilidad, al definir claramente qué partes del código pueden ser modificadas desde el exterior.

Ventajas

- Mayor seguridad en el acceso a datos
- Claridad en la estructura del código
- Facilita el mantenimiento del sistema
- Prevención de errores por acceso indebido

Desventajas

- Mayor complejidad para principiantes
- Restricciones que pueden dificultar pruebas o reutilización
- Posible sobreuso que limite la flexibilidad del diseño

Ejemplo:

```
public class Persona {

private String nombre;

public void saludar() {

System.out.println("Hola") ;
```

Otras opciones

- Constructor: Método especial que se ejecuta al crear un objeto. Sirve para inicializar valores.
- Bloque estático: Se ejecuta una vez cuando la clase se carga. Ideal para inicializar recursos.
- Método estático: Método que pertenece a la clase, no al objeto. Se puede usar sin crear instancias.
- Enumeración: Tipo especial que define un conjunto fijo de constantes con nombre propio.

Importancia

- Permiten inicializar objetos correctamente
- Ayudan a organizar el código dentro de una clase
- Facilitan la ejecución de tareas comunes sin crear objetos
- Mejoran la modularidad y reutilización
- Permiten definir constantes y estructuras específicas

Ventajas

- Mayor flexibilidad en el diseño
- Mejor organización interna
- Posibilidad de ejecutar código sin instanciar objetos
- Definición clara de constantes
- Facilita la reutilización de lógica común

Desventajas

- Puede aumentar la complejidad del código
- Requiere mayor conocimiento técnico
- Mal uso puede generar confusión
- Difícil de mantener si no se documenta bien

Ejemplo:

```
static {
System.out.println("Inicializacion de clase");
```

Referencias

1. Oracle. (2024). *Providing constructors for your classes*. Oracle Java Documentation. https://docs.oracle.com/javase/tutorial/java/javaOO/constructors.html

2. Oracle. (2024). *Inner classes*. Oracle Java Documentation. https://docs.oracle.com/javase/tutorial/java/javaOO/innerclasses.html

3. Bloch, J. (2018). *Effective Java* (3rd ed.). Addison-Wesley.

4. Gosling, J. (1995). *Java: Evolution, Features, and Its Impact on Modern Software Development*. *International Journal of Novel Research and Development*, 4(8), 123–130. https://www.ijnrd.org/papers/IJNRD2408023.pdf

5. SpringerLink. (2025). *Java - Recent Articles and Discoveries*. https://link.springer.com/subjects/java