**UCSM Esc. Prof. de Ingeniería de Sistemas**

**Agosto - 2025**

INFORME DE PRÁCTICAS

**LENGUAJES DE**

**PROGRAMACIÓN III**



**Práctica N° 01: POO Avanzada, Principios SOLID**

Elaborado por: Alarcón Cervantes Fernando Santiago

**© IEEE 2013 The Institute of Electrical and Electronics Engineers, Inc.**

**GRUPO N° 05**

**PRÁCTICAS DE LENGUAJES DE PROGRAMACIÓN III**

Presentado por:

|  |  |  |
| --- | --- | --- |
| 2024001333 | Alarcón Cervantes, Fernando Santiago | 100% |

**RECONOCIMIENTOS**

**Quiero expresar mi gratitud al ingeniero Mario Santillana por su orientación en el tema que nos aportó conocimientos útiles para poder desarrollar la practica y aplicar estos conocimientos en nuestra vida universitaria y laboral**

**PALABRAS CLAVES**

**ÍNDICE**

1. [RESÚMEN 1](#_bookmark0)
2. [INTRODUCCIÓN 1](#_bookmark1)
3. [MARCO TEÓRICO 1](#_bookmark2)
   1. [Tipos de lenguaje de programación 1](#_bookmark3)
   2. [Categorías de Java 1](#_bookmark4)
   3. [Etapas de programación en Java 2](#_bookmark5)
   4. [Variables 2](#_bookmark6)
   5. [Estructuras Secuenciales 2](#_bookmark7)
   6. [Estructuras condicionale 3](#_bookmark8)
   7. [Estructuras Repetitivas 3](#_bookmark9)
4. [ACTIVIDADES 5](#_bookmark10)
   1. [Expericia de Práctica N° 01: tema 1 5](#_bookmark11)
   2. [Expericia de Práctica N° 02: tema 2 6](#_bookmark12)
   3. [Expericia de Práctica N° 03: tema 3 6](#_bookmark13)
   4. [Expericia de Práctica N° 04: tema 4 7](#_bookmark14)
5. [EJERCICIOS PROPUESTOS 10](#_bookmark15)
   1. [Ejercicio N° 01: Fundamentos de Java 10](#_bookmark16)
   2. [Ejercicio N° 02: Fundamentos de Java 12](#_bookmark17)
   3. [Ejercicio N° 03: Fundamentos de Java 12](#_bookmark18)
   4. [Ejercicio N° 04: Fundamentos de Java 13](#_bookmark19)
   5. [Ejercicio N° 05: Fundamentos de Java 14](#_bookmark20)
   6. [Ejercicio N° 06: Fundamentos de Java 15](#_bookmark21)
6. [CONCLUSIONES DE LA PRÁCTICA 15](#_bookmark22)
7. [CUESTIONARIO 15](#_bookmark23)
8. [REFERENCIAS 16](#_bookmark24)
9. **RESÚMEN**

En esta práctica se trabajará la aplicación de los principios SOLID en java y como estos son una buena practica a la hora de poder escalar o modificar el código sin que ocurran errores como la opacidad o viscosidad en este además de esto se ve como se puede aplicar en casos de la vida real como sistemas para poder enviar mensajes, un código que necesite ser escalable o se opte por usar el mismo formato solo que dirigido a otra área.

# INTRODUCCIÓN

# Los principios SOLID son reglas que se aplican a la programación orientada a objetos, para asi poder mejorar su diseño y arquitectura, esto también se ve en el ámbito de cooperación ayudando que se pueda añadir código sin complicaciones, en la practica se usara ára poder aplicarlo a códigos donde se modificaran estos para que se puedan aplicar las buenas prácticas, con cada uno de los principios que nos ofrece.

# MARCO TEÓRICO

# Existen 5 principos SOLID estos ayudan a poder modelar un problema o mala practica en un código



# ACTIVIDADES

## Expericia de Práctica N° 01: tema 1

Números primos: Escribe un programa que incluya una función para verificar si un número es primo y otra función para imprimir todos los números primos entre 1 y un número dado n.

Pantalla

## Expericia de Práctica N° 02: tema 2

Palíndromos: Escribe un programa que verifique si un número entero es un palíndromo. Un número es un palíndromo si se lee igual de izquierda a derecha que de derecha a izquierda.

Código:

Pantalla:

## Expericia de Práctica N° 03: tema 3

Números perfectos: Escribe un programa que encuentre y muestre todos los números perfectos menores a un número dado n. Un número perfecto es un número entero positivo que es igual a la suma de sus divisores propios positivos, excluyendo el mismo número. Así, 6 es un número perfecto porque sus divisores propios positivos son 1, 2 y 3; y 6 = 1

+ 2 + 3. Un divisor propio positivo de un número es un factor positivo de ese número que no sea el propio número. Por ejemplo, los divisores propios de 6 son 1, 2 y 3, pero no 6. Los siguientes números perfectos son 28, 496 y 8128.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 28 |  | = |  | 1 |  | + |  | 2 |  | + |  | 4 |  | + |  | 7 | + |  | 14 |
| 496 | = | 1 | + | 2 | + | 4 | + | 8 | + | 16 | + |  | 31 | + | 62 | + | 124 | + | 248 |

8128 = 1 + 2 + 4 + 8 + 16 + 32 + 64 + 127 + 254 + 508 + 1016 + 2032 + 4064

Codigo:

Pantalla:

## Expericia de Práctica N° 04: tema 4

Arreglos: Escribe un programa que permita al usuario introducir una serie de números en un arreglo y luego calcule el promedio.

Pantalla:

* 1. **Expericia de Práctica N° 05: tema 5**

Arreglos: Crea un arreglo con los números del 1 al 5 en java, python y c++ e identifica sus diferencias

Diferencias clave Tipado:

1. Java y C++: Tipado estático; el tipo de datos del arreglo debe declararse y es inmutable. 2’
2. Python: Tipado dinámico; no es necesario declarar el tipo de la lista. Tamaño:
   1. Java y C++: Los arreglos tienen un tamaño fijo una vez creados.
   2. Python: Las listas son dinámicas y pueden cambiar de tamaño. Declaración:
      1. Java y C++: Los arreglos se crean con una sintaxis específica que requiere definir el tipo.
      2. Python: Las listas se crean directamente con valores entre corchetes. Flexibilidad:
         1. Java y C++: Los arreglos son menos flexibles en términos de tamaño y tipo.
         2. Python: Las listas son mucho más flexibles, permitiendo operaciones como la inserción, eliminación y mezcla de tipos de datos.

# EJERCICIOS PROPUESTOS

* 1. **Ejercicio N° 01: Fundamentos de Java**

Python:

C++:

Java:

* 1. **Ejercicio N° 02: Fundamentos de Java**
  2. **Ejercicio N° 03: Fundamentos de Java**
  3. **Ejercicio N° 04: Fundamentos de Java**
  4. **Ejercicio N° 05: Fundamentos de Java**
  5. **Ejercicio N° 06: Fundamentos de Java**

# CONCLUSIONES DE LA PRÁCTICA:

# CUESTIONARIO

1. **Crea un programa que salude con el texto “Bienvenido a Java, seguido de tu nombre”. Explica cómo es que se ha procesado el código del ejercicio anterior**

1. **¿Java es compilado o interpretado? ¿Por qué?**

1. **¿Qué diferencias hay entre un lenguaje compilado e interpretado?**

1. **¿Qué ventajas tiene java sobre otros lenguajes de programación?**
2. **¿Qué versión de JDK es la más usada?**

# REFERENCIAS

1. J. K. Ousterhout, “Scripting: Higher-Level Programming for the 21st Century,”

*IEEE Computer*, vol. 31, no. 3, marzo 1998.

1. “High-level programming language,” *ScienceDirect*, artículo reciente.

GitHub url: <https://github.com/Karasu01/ACTIVIDADES-Y-EJERCICIOS.git>