# Juan Manuel Navia Castillo

# Taller de nivelación PI a PII

## Objetivos del Taller

* Reforzar los conceptos fundamentales de PI básica en Java.
* Introducir conceptos básicos que se desarrollarán en PII.
* Fomentar la resolución de problemas y el pensamiento algorítmico.

## Estructura del Taller

El taller estará dividido en dos partes: teórica y práctica. Adicionalmente se debe entregar la solución de los ejercicios prácticos del taller en un repositorio, siguiendo los pasos descritos en la guía.

# Parte Teórica

1. ¿Cuáles son los tipos de datos primitivos en Java?
2. ¿Cómo funcionan las estructuras de control de flujo como if, else, switch y bucles en Java?
3. ¿Por qué es importante usar nombres significativos para variables y métodos?
4. ¿Qué es la Programación Orientada a Objetos (POO)?
5. ¿Cuáles son los cuatro pilares de la Programación Orientada a Objetos?
6. ¿Qué es la herencia en POO y cómo se utiliza en Java?
7. ¿Qué son los modificadores de acceso y cuáles son los más comunes en Java?
8. ¿Qué es una variable de entorno y por qué son importantes para Java o la programación en general?

# Parte Práctica

1. Crear un programa que utilice estructuras de control para resolver un problema sencillo, como una calculadora básica que realice operaciones de suma, resta, multiplicación y división.
2. Escribir un programa que cuente el número de vocales y consonantes en una palabra. La palabra no contendrá símbolos, caracteres especiales, acentos, ni números y siempre estará en minúsculas.
3. Escribir un programa que invierta una cadena de texto ingresada por el usuario.

# Guía para Entregar Ejercicios en GitHub

### Paso 1: Crear una Cuenta en GitHub

1. Visita [GitHub](https://github.com/).
2. Regístrate para obtener una cuenta gratuita si aún no tienes una.

### Paso 2: Instalar Git

1. Descarga e instala Git desde [git-scm.com](https://git-scm.com/).
2. Configura tu nombre de usuario y correo electrónico en Git:

git config --global user.name "Tu Nombre"

git config --global user.email "[tuemail@example.com](mailto:tuemail@example.com)"

### Paso 3: Crear un Nuevo Repositorio

1. Inicia sesión en tu cuenta de GitHub.
2. Haz clic en el botón **New** en la página principal para crear un nuevo repositorio.
3. Nombra tu repositorio (por ejemplo, “**Ejercicios”**) y añade una descripción breve.
4. Marca la casilla para inicializar el repositorio con un archivo “**README.md”**.
5. Haz clic en **Create repository.**

### Paso 4: Clonar el Repositorio en tu Computadora

1. Copia la URL de tu nuevo repositorio desde GitHub.
2. Abre una terminal o consola en tu computadora.
3. Clona el repositorio usando Git:

git clone <URL-del-repositorio>

Reemplaza **<URL-del-repositorio>** con la URL de tu repositorio de GitHub.

### Paso 5: Añadir los Archivos del Proyecto

1. Navega al directorio del repositorio clonado:

cd Ejercicios

1. Añade tus archivos de código al directorio del repositorio.

### Paso 6: Crear el Archivo README.md

1. Abre el archivo **“README.md”** en un editor de texto.
2. Añade información relevante, como el propósito del proyecto, instrucciones de instalación y uso, y cualquier otra información importante.

Ejemplo de contenido para el **“README.md”**:

# Ejercicios de Programación

Este repositorio contiene los ejercicios de programación en Java del taller de nivelación. ## Contenido

* Calculadora Básica
* Contador de Vocales y Consonantes
* Invertir una Cadena ## Instrucciones

1. Clona el repositorio.
2. Navega a cada directorio de ejercicios.
3. Compila y ejecuta los archivos `.java` para probar los ejercicios.

### Paso 7: Realizar el Commit de los Cambios

1. Añade los archivos al área de preparación (staging):

git add .

1. Realiza un commit de los cambios:

git commit -m "Añadir ejercicios de programación y actualizar README"

### Paso 8: Subir los Cambios a GitHub

1. Sube los cambios a tu repositorio en GitHub:

git push origin main

Nota: Si creaste una rama diferente a main, usa el nombre correcto de tu rama.

### Paso 9: Compartir el Enlace del Repositorio

1. Copia la URL de tu repositorio desde GitHub.
2. Envía el enlace a tu profesor o al lugar designado para la entrega.

O también puedes seguir este video para mayor facilidad: <https://www.youtube.com/watch?v=q6RKq91FKC4> Recursos:

[https://netbeans.apache.org/](https://www.youtube.com/redirect?event=video_description&redir_token=QUFFLUhqazEzRDdEYmFIWFpqYlplczFjSXlYUjhXNWpYZ3xBQ3Jtc0tsa0M1ZFNZclRBa1ZHLVByZ1ZTMVhkZjZ4Rlg3YXl3UnhlZ1lSdXN4b3kzQ0U3WDJqVXJMNXRBbS1TcFJMa2RUNmlzbGNBMG9GVXpqUWJUSzZ5eTdIbkI2VnUwVzZoZnhVdmxWTTRTa3oyMUJEaHNMOA&q=https%3A%2F%2Fnetbeans.apache.org%2F&v=q6RKq91FKC4) <https://netbeans.apache.org/tutorial/main/kb/docs/java/>

Desarrollo de taller:

Parte teórica:

1)los tipos de datos primitivos en java son:

* byte
* short
* int
* long
* float
* double
* char
* boolean

2)

* condicionales if.else,if-else:

Estos condicionales se utilizan para evaluar una condición y ejecutan un bloque de código dependiendo si la condición es verdadera. La estructura básica consiste en evaluar una condición si dicha condición es verdadera se ejecuta un bloque de instrucciones. Si no se cumple podemos especificar un bloque alternativo.

* Switch-case:

Esta estructura evalúa el valor de una variable con varios casos predefinidos, cada uno de los casos significa una posibilidad y se ejecutara uno de esos casos si la variable coincide.

* Blucle FOR:

Es utilizado para repetir un conjunto de código según la condición, su estructura tiene 3 partes: la INICIALIZACION

Que se realiza antes del for y se realiza una sola vez, la CONDICIÓN que se evalúa antes de cada iteración y si es falsa el bucle concluye, en caso de que la condición sea verdadera el bloque de código se ejecutara y por último la ACTUALIZACION que modifica la variable de control después de cada iteración.

* Bucle while:

Este bucle permite evaluar una condición antes de la iteración y continúa ejecutándose si la condición es verdadera.

* Do-while:

Es muy similar al while, la diferencia es que evalúa la condición después de ejecutar el bloque de instrucciones, esto permite que el código se ejecute al menos una vez.

3)es importante por que nos permite que haya más claridad en nuestro código y también, que sea mucho más fácil de entender, nombres descriptivos dan a entender la función de cada elemento facilitando la lectura de este y reduciendo los errores.

También podríamos decir que mejora el mantenimiento del código ya que habiendo tal claridad en el programa nos permite identificar y corregir errores de una manera más ágil y sencilla.

4)la programación orientada a objetos es un paradigma de programación que consiste en clases y objetos. El principal objetivo es organizar el código en estructuras que representen entidades del mudo real.

5)

* ABTRACCION.
* Encapsulamiento
* Herencia
* Polimorfismo

6) la herencia es una función que nos permite que una clase herede los atributos de otra y método de otra permitiéndonos reutilizar código, en java lo podemos implementar utilizando la palabra clase extends.

7) los modificadores de acceso más comunes son:

* Private
* Public
* Protected
* Default

Estos

8) una variable de entorno es un valor dinámico que el sistema operativo proporciona y son importantes ya que

1. Enlace repositorio github: <https://github.com/Juanmanuelnavia/calculadora.git>
2. Enlace 2:

<https://github.com/Juanmanuelnavia/contadprletras.git>

1. Enlace 3:

<https://github.com/Juanmanuelnavia/intercavioletras.git>