

# ESCUELA POLITECNICA NACIONAL

Isaac Proaño

PROGRAMACION II

## MARKDOWN PRIMER BIMESTRE



Clase #1 --- 07/11/2023

### 1. IDE Visual Studio Code

Visual Studio Code (VS Code) es un editor de código fuente gratuito y de código abierto desarrollado por Microsoft. Es altamente personalizable, ligero y compatible con una variedad de lenguajes de programación, con características como resaltado de sintaxis, depuración integrada y soporte para extensiones.

#### 1.1 Funcionamiento

0. Instalar JDK.
1. Instalar git bash.
2. Instalar el VScode.
3. Crear una cuenta de github e integrarla al VScode.
4. Descargar las extensiones para el correcto funcionamiento de VScode (Extension pack for Java)

Tip:

Probar juegos de mecanografía para mejorar la velocidad, precisión y memorización a la hora de usar el teclado.

#### 1.2 Configuracion +\_+

0. Vincular la cuenta de github para que sea sencillo trabajar y guardar los trabajos realizados en la nube.
1. Crear un workdirectory para guardar los archivos de forma local.
2. Verificar que se encuentren instaladas correctamente todas las extensiones para que el VScode funcione correctamente, además de generar un mejor entorno para trabajar.
3. Configurar la consola para que el ambiente de trabajo se vuelva mas amigable, esto se puede lograr mediante extensiones del propio VScode o mediante el uso de [Oh my posh](#).
  - Tipografía: JetBrains Mono, MesloLGL Nerd Font
  - Theme
  - Icons

## Clase #2 --- 08/11/2023

---

### 1.\_ Atajos para Visual Studio Code (Shortcuts)

<b>Shortcut</b>	<b>Descripción</b>
Ctrl + P	Abrir archivo por nombre
Ctrl + Shift + N	Abrir una nueva ventana de código
Ctrl + B	Mostrar/Ocultar barra lateral
Ctrl + `	Abrir/cerrar la terminal integrada
Ctrl + D	Seleccionar la siguiente coincidencia de texto
Ctrl + Shift + L	Seleccionar todas las coincidencias de texto
Ctrl + F	Buscar en el archivo actual
Ctrl + Shift + F	Buscar en todos los archivos
Ctrl + /	Comentar o descomentar líneas seleccionadas
Alt + UP or DOWN	Mover la línea actual hacia arriba o abajo
Ctrl + Shift + K	Eliminar línea actual
Ctrl + D	Cursor de selección multiple

### 2.\_ MarkDown

#### 2.1 ¿Qué es Markdown?

Markdown es un lenguaje de marcado ligero que facilita la escritura y la lectura de textos con formato, permitiendo crear documentos con estilos básicos sin la complejidad del HTML.

## .2.2 Sintaxis básica

Elemento	Sintaxis	Ejemplo
Encabezados	# Encabezado	# Título Principal
Énfasis	*Itálicas* o **Negritas**	*éñfasis* o **éñfasis**
Listas	- Elemento de lista	- Elemento uno
Enlaces	[Texto del enlace](URL)	[Enlace a Google] ( <a href="https://www.google.com">https://www.google.com</a> )
Imágenes	![Texto alternativo](URL)	![Logo](logo.png)
Código	`código`	`print("Hola Mundo")`
Bloques de código	  Código 	  print("Hola Mundo") 
Citas	> Texto de la cita	> Esto es una cita

## 2.3 Colores

Para agregar colores en formato MarkDown existen varias formas, en este caso se uso conceptos básicos de html lo cual sirve para que se pueda agregar distintos colores en este tipo de formato.

Color	Código	Ejemplo
Blanco	#FFFFFF	
Negro	#000000	Texto negro
Rojo	#FF0000	Texto rojo
Verde	#00FF00	Texto verde
Azul	#00FFFF	Texto azul
Amarillo	#FFFF00	Texto amarillo
Naranja	#FFA500	Texto naranja
Rosa	#FFC0CB	Texto rosa
Morado	#800080	Texto morado
Cyan	#00FFFF	Texto cyan
Gris	#808080	Texto gris

Color	Código	Ejemplo
Dorado	#FFD700	Texto dorado

## 2.4 Características adicionales

- **Tablas:** Markdown permite crear tablas con alineación y contenido personalizado.
- **HTML:** Puedes incrustar código HTML directamente en documentos Markdown para funciones más avanzadas.
- **Extensiones:** Algunos editores admiten extensiones para Markdown que ofrecen funcionalidades adicionales, como diagramas o ecuaciones.

## 2.5 Consejos

- Usar atajos de teclado para aumentar tu productividad al escribir en Markdown.
- Siempre es bueno probar la vista previa para asegurarte de que el formato sea el deseado.
- Explorar las extensiones de Markdown disponibles en editores para mejorar tu flujo de trabajo.

## GUÍA COMPLETA DE MARKDOWN

## 3.\_ GIT y GIT BASH

### ¿Qué es Git?

Git es un sistema de control de versiones que permite gestionar cambios en el código de manera eficiente. Permite realizar un seguimiento de las modificaciones, trabajar en ramas diferentes y fusionar cambios entre ellas.

#### Introducción a Git

- **Control de Versiones:** Permite mantener un historial de cambios en el código, lo que facilita revertir a versiones anteriores si es necesario.
- **Sistema Distribuido:** Cada usuario tiene una copia local del repositorio, lo que facilita trabajar de manera independiente y luego fusionar cambios.

#### Comandos Básicos de Git

- `git init`: Inicia un nuevo repositorio Git en el directorio actual.
- `git clone URL`: Clona un repositorio remoto en tu máquina local.
- `git add <archivo>`: Agrega archivos al área de preparación para ser confirmados.
- `git commit -m "mensaje"`: Confirma los cambios realizados con un mensaje descriptivo.
- `git push`: Sube los cambios confirmados al repositorio remoto.
- `git pull`: Obtiene los cambios del repositorio remoto y los fusiona con tu rama actual.

### ¿Qué es Git Bash?

Git Bash es una terminal que proporciona una interfaz de línea de comandos para interactuar con Git en sistemas operativos Windows. Permite ejecutar comandos de Git y también comandos de UNIX.

#### Introducción a Git Bash

- **Interfaz de Comandos:** Proporciona una forma poderosa de utilizar Git a través de comandos en lugar de una interfaz gráfica.
- **Funcionalidades de UNIX:** Permite utilizar comandos comunes de UNIX en sistemas Windows, lo que amplía sus capacidades.

## Comandos Útiles en Git Bash

- `ls`: Lista los archivos y directorios en el directorio actual.
- `cd <directorío>`: Cambia el directorio de trabajo.
- `mkdir <nombre_directorio>`: Crea un nuevo directorio.
- `touch <nombre_archivo>`: Crea un nuevo archivo.
- `cat <nombre_archivo>`: Muestra el contenido de un archivo en la terminal.

## Consejos

- **Practica en un entorno seguro:** Antes de realizar cambios importantes, practica en ramas separadas o clona repositorios de prueba para familiarizarte con los comandos y evitar errores en proyectos reales.
- **Aprende a leer el historial de commits:** Entender el historial de cambios te ayudará a comprender mejor el progreso del proyecto y a identificar errores o problemas específicos más fácilmente.
- **Personaliza tu entorno Git Bash:** Aprender a personalizar Git Bash ajustando colores, alias y configuraciones para mejorar tu experiencia de uso y eficiencia en la terminal.

## Clase #3 --- 09/11/2023

---

### 1. GitHub

#### ¿Qué es GitHub?

GitHub es una plataforma de alojamiento de código que utiliza Git como base. Permite a los desarrolladores colaborar en proyectos, alojar repositorios, realizar seguimiento de problemas, realizar revisiones de código y mucho más.

#### Introducción a GitHub

- **Colaboración:** Permite a equipos de desarrollo trabajar en proyectos de forma colaborativa, gestionando cambios y contribuciones.
- **Gestión de Proyectos:** Ofrece herramientas para la gestión de problemas, seguimiento de errores, administración de versiones y revisión de código.

#### Funcionalidades Principales de GitHub

- **Repositorios:** Aloja proyectos y permite el control de versiones con Git.
- **Issues:** Permite el seguimiento de problemas y solicitudes de funciones.
- **Pull Requests:** Facilita la revisión y fusión de cambios realizados en ramas.

## Consejos

- **Usa comentarios claros y descriptivos:** Al abrir issues o pull requests, proporcionar comentarios claros y detallados para que otros colaboradores comprendan rápidamente el propósito y contexto de tus cambios.
- **Aprovecha las ramas y los pull requests:** Utilizar ramas separadas para trabajar en nuevas funcionalidades o correcciones. Los pull requests son útiles para revisar y discutir cambios antes de fusionarlos en la rama principal.
- **Explora las integraciones y herramientas:** Aprovechar las integraciones de GitHub con otras herramientas como servicios de integración continua, despliegue automatizado o herramientas de gestión de proyectos para mejorar la eficiencia y calidad del desarrollo.

## 2.\_ Linux

### 2.1 ¿Qué es Linux?

Linux es un sistema operativo de código abierto basado en UNIX, mientras que Git Bash es una emulación de la línea de comandos de UNIX en sistemas Windows. Git Bash proporciona una interfaz de línea de comandos similar a la de Linux, permitiendo ejecutar comandos y scripts similares a los utilizados en entornos Linux.

### 2.2 Introducción - Comandos Básicos

- **Terminal Bash:** Git Bash ofrece una terminal que emula la experiencia de línea de comandos de Linux en Windows, lo que permite ejecutar varios comandos.
  - **Shell Bash:** Utiliza el shell Bash, un intérprete de comandos común en sistemas Linux, lo que facilita la transición entre comandos en ambientes Linux y Windows.
1. `ls`: Lista los archivos y directorios en el directorio actual.
  2. `cd <directorio>`: Cambia el directorio de trabajo.
  3. `mkdir <nombre_directorio>`: Crea un nuevo directorio.
  4. `rm <archivo>`: Elimina un archivo.
  5. `rm -r <directorio>`: Elimina un directorio y su contenido de manera recursiva.
  6. `touch <nombre_archivo>`: Crea un nuevo archivo.
  7. `cp <archivo_origen> <archivo_destino>`: Copia archivos o directorios.
  8. `mv <archivo_origen> <archivo_destino>`: Mueve archivos o directorios.
  9. `cat <nombre_archivo>`: Muestra el contenido de un archivo en la terminal.
  10. `grep <patrón> <archivo>`: Busca un patrón en un archivo.

### 2.3 Características de Linux en Git Bash

- **Emulación de Comandos:** Permite ejecutar una variedad de comandos y scripts de Linux directamente en un entorno Windows, facilitando la transición para usuarios familiarizados con Linux.
- **Utilidades de UNIX:** Proporciona herramientas y utilidades comunes presentes en sistemas UNIX/Linux, lo que permite un flujo de trabajo similar al de Linux en entornos Windows.

### 2.4 Ventajas y Consideraciones

- **Compatibilidad y Versatilidad:** Permite a los usuarios acostumbrados a Linux utilizar comandos familiares en un entorno Windows, facilitando la transición entre plataformas.
- **Limitaciones:** Aunque ofrece una experiencia similar, algunas diferencias y limitaciones pueden surgir debido a las diferencias entre sistemas operativos.

## 2.5 Consejos

- **Practica con Comandos Básicos:** Familiarízate con los comandos comunes de Linux para aprovechar al máximo Git Bash en Windows.
- **Explora las Similitudes y Diferencias:** Aprende sobre las similitudes y diferencias entre Linux y Git Bash para comprender mejor su funcionamiento y evitar confusiones.
- **Aprovecha Recursos en Línea:** Hay numerosos recursos y tutoriales disponibles para aprender a utilizar Git Bash en Windows y mejorar tu experiencia.

# Clase #4 --- 10/11/2023

---

## 1.\_ JAVA

### 1.1 Origen de Java

Java fue desarrollado por Sun Microsystems en la década de 1990 como un lenguaje de programación versátil y orientado a objetos. Su objetivo era crear un lenguaje que pudiera ejecutarse en cualquier dispositivo independientemente de la plataforma.

### 1.2 ¿Qué es Java?

Java es un lenguaje de programación de alto nivel, orientado a objetos y con una sintaxis similar a C++. Es conocido por su portabilidad, ya que el código escrito en Java puede ejecutarse en diferentes plataformas sin necesidad de recompilarlo.

### 1.3 ¿Para qué sirve Java?

Java se utiliza en una amplia gama de aplicaciones, desde desarrollo de aplicaciones web hasta aplicaciones móviles, sistemas embebidos y software empresarial. Es el lenguaje principal para el desarrollo de aplicaciones Android y se utiliza ampliamente en servidores web.



## 1.4 Elementos importantes en java

- Paquetes:** Son una forma de organizar el código en Java. Un paquete es un conjunto de clases que están relacionadas entre sí.
- Clases:** Son la unidad básica de construcción en Java. Una clase define un tipo de dato, que puede ser utilizado para crear objetos.
- Objetos:** Son instancias de clases. Un objeto tiene sus propios datos y métodos.
- Métodos:** Son las acciones que pueden realizar los objetos.
- Variables:** Se utilizan para almacenar datos.
- Constantes:** Las constantes son variables que no pueden cambiar su valor.
- Operadores:** Los operadores se utilizan para realizar operaciones matemáticas y lógicas.
- Sentencias:** Las sentencias son instrucciones que se ejecutan en Java.

## 1.4 Empezar con Estructuras y Sintaxis en Java

### Estructuras Fundamentales

- Variables:** Declaración y asignación de variables.

```
int numero = 10;
String nombre = "Juan";
```

- Tipos de datos:** Enteros, decimales, caracteres, booleanos, etc.

```
int edad = 25;
double altura = 1.75;
char inicial = 'J';
boolean esEstudiante = true;
```

- Flujos de control:** Ejemplo básico de un condicional.

```
int numero = 5;
if (numero > 0) {
    System.out.println("El número es positivo.");
} else {
    System.out.println("El número es cero o negativo.");
}
```

- **Bucles:** Ejemplo de un bucle.

```
for (int i = 0; i < 5; i++) {
    System.out.println("El valor de i es: " + i);
}
```

- **Hola mundo en java:**

```
public class HolaMundo {
    public static void main(String[] args) {
        System.out.println("¡Hola Mundo!");
    }
}
```

## 1.5 Comando para ejecutar un programa desde la consola

- Java "Nombre\_del\_archivo.java"

## 1.6 Tips y Consejos

- **Practica Regularmente:** La práctica constante es fundamental para mejorar en Java. Realiza pequeños proyectos o desafíos para aplicar lo aprendido.
- **Explora la Documentación Oficial:** La documentación de Java es una gran fuente de información. Aprende a usarla para comprender mejor las clases y métodos disponibles.
- **Participa en Comunidades:** Únete a foros, grupos en redes sociales o comunidades de desarrolladores para obtener ayuda, consejos y mantenerte actualizado.



**Todos los archivos pertenecen a un paquete**

**Importa los paquetes para el proyecto**

**Java usa clases para ejecutar el código**

**Se debe indicar el tipo de dato**

**Modificadores de acceso:**  
private, public, protected o por defecto ninguno

El método principal en Java es **el método main**

La palabra reservada **new** **crea un objeto del tipo de dato especificado**

```

1 package team.ed.course;
2 import java.lang.*;
3 public class Person {
4     private String name;
5     public static void main(String args[]){
6         Person friend = new Person();
7         friend.name = "Peter";
8         System.out.println("Hola " +
9             friend.name);
10    }
11 }
  
```

Se utilizan **;** para cada sentencia

Se usan **{}** para identificar el **bloque de código**

## Clase #5

Flujo de control y diagramas de flujo

- Secuencia
- Condicional
- De control

Tipos de datos - P.O.O

Documentacion

Ambitos

public/protect/private >

Propiedades

Metodos

Eventos

## Clase #6

---

¿Como funciona la ram?

- Instancia = {constructor -> new()}
  - Que es un constructor?
  - En funcion de la clase se crea el objeto/variable
  - Propiedades de un constructor:
    - + el constructor tiene el mismo nombre de la clase.
    - + Se ejecuta la primera vez y es unica.
    - + "Es un método especial"
    - + Debe ser público si se quiere utilizar la clase (si se coloca privado no se podra utilizar el objeto)
    - + Debe ser "único"
    - + Sirve para inicializar valores.
- = F5 para ejecutar el archivo y guardararlo

## Clase #7

---

## Clase #8

---

Practicar y conocer el lenguaje mediante la realizacion de los poliretos.

- Refactorizar
- Se crea una clase llamada cadena

## Clase #9

---