	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor:	José Antonio Ayala Barbosa
Asignatura:	Programación Orientada a Objetos
Grupo:	Gpo 1 22-2
No de Práctica(s):	09
Integrante(s):	Rosillo Montijo Emmanuel Alonso
No. de Equipo de cómputo empleado:	
No. de Lista o Brigada:	
Semestre:	03
Fecha de entrega:	17/02/2022
Observaciones:	

CALIFICACIÓN: _____

Previo

Markdown:

Markdown es un lenguaje de marcado ligero que puede usar para agregar elementos de formato a documentos de texto sin formato. Creado por John Gruber en 2004, Markdown es ahora uno de los lenguajes de marcado más populares del mundo.

La Guía de Markdown es una guía de referencia gratuita que explica cómo usar Markdown, el lenguaje de marcado simple y más fácil de usar para formatear prácticamente cualquier tipo de documentos.

El objetivo de diseño primordial de la sintaxis de formato de Markdown es hacerlo lo más legible posible. La idea es que un documento con formato Markdown se pueda publicar tal cual, como texto sin formato, sin que parezca que ha sido marcado con etiquetas o instrucciones de formato.

VS Code

Visual Studio Code es un editor de código fuente que permite trabajar con diversos lenguajes de programación, admite gestionar tus propios atajos de teclado y refactorizar el código. Es gratuito, de código abierto y nos proporciona una utilidad para descargar y gestionar extensiones con las que podemos personalizar y potenciar esta herramienta.

Actualmente, es uno de los editores de código más utilizados y uno de los motivos es la cantidad de plugins útiles de los que dispone.

Plugins VS code

En esta práctica se instalaron los siguientes plugins para su elaboración:

- Marp for VS Code
- Markmap
- PlantUML o PlantUML Previewer

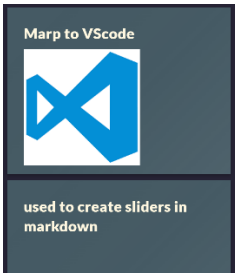
Objetivo

Utilizar UML como herramienta para diseñar soluciones de software para un lenguaje de programación orientado a objetos.

Actividades

Cómo primera actividad dentro de laboratorio, se creó un archivo markdown Y dónde lo de este archivo se colocó una etiqueta con la siguiente forma:

Está etiqueta nos permitió especificar a *markdown* que íbamos a trabajar con *marpdown* y de esta forma empezar a crear las diapositivas de la siguiente manera:



Empezamos experimentar con las diferentes herramientas con las cuales nos permite hacer un código más personalizado, como, por ejemplo:

```
---
# Bienvenidos a Markdown
## Subtitulo
### subsubtitulo
#### subsubsubtitulo
##### subsubsubtitulo

texto normal
---
---
# Listas/viñetas
- Vi 1
- Vi 2
    - viñeta 2.1
    - **negritas**
    - *italica*
    - ~tachado~
    - `quoted`
---
```

Nos permite darle ciertos estilos a las letras también a los encabezados para hacer un texto formato de documento en presentación.

También insertamos imágenes con formatos y estilos diferentes, de la siguiente forma:

```
---
# Deep Learning

![pus no mas](
https://www.investopedi
a.com/thmb/5MRjyUTzTN0jS
jftLssHXpM0ZXo=/660x0/fi
lters:no_upscale(
):max_bytes(150000):stri
p_icc()/dotdash_Final_Ne
ural_Network_Apr_2020-01
-5f4088dfda4c49d99a4d927
c9a3a5ba0.jpg)
---
---
# Deep Learning mood

![width:500](image.jpg)
---
```

De esta forma podemos insertar una imagen desde una liga y también de forma local.

Cómo herramientas extra aprendemos a utilizar las tablas de esta herramienta y algunos otros remarcados de texto que nos permitieron tener un documento completo

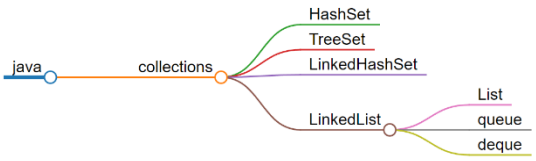
```
# Hiper

<https://www.unam.mx>
<fb.verify.com@gmail.com
>
yo asisto a la [fi](
http://www.fi-b.unam.mx)
---
---
# TABLE

|Tit 1|tit 2|tit 3|
|---|---|---|
|info|info|info|
```

Markmap

Diseñamos dos documentos de markdown, específicamente para probar los marps qué nos proporciona el plugin del título, la estructura de los documentos queda de la siguiente forma:

```
# java
## collections
### HashSet
### TreeSet
### LinkedHashSet
### LinkedList
#### List
#### queue
#### deque
```

Y al exportarlo queda de la siguiente forma:



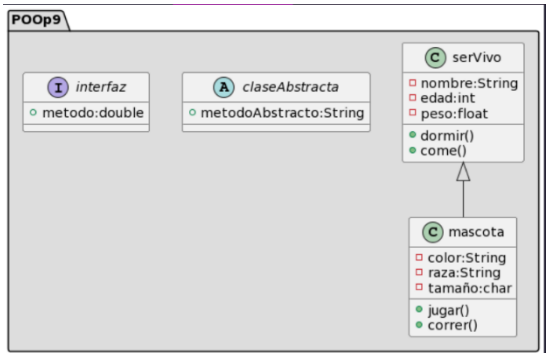
PlantUML o PlantUML Previewer

No sirvió para crear uml dentro de un lenguaje de marcado y de esta forma poder hacer la preview de uml de las clases que ya tenemos dentro del código.

La sintaxis del lenguaje de marcado:

```
@startuml
package "POOp9" #DDDDDD{
class serVivo{
    -nombre:String
    -edad:int
    -peso:float
    +dormir()
    +come()
}
class mascota{
    -color:String
    -raza:String
    -tamaño:char
    +jugar()
    +correr()
}
abstract class claseAbstracta
{
    +metodoAbstracto:String
}
interface interfaz{
    +metodo:double
}

serVivo <|-- mascota
}
@enduml
```

Y quedó de la siguiente forma:



De esta forma se puede crear un UML de la clase que tenemos generada en el código, para que nos permita documentar el diagrama también.

Actividades extra

Con las actividades anteriormente vistas se crearon la explicación de todas las herramientas dentro del desarrollo de la práctica para así incrementar la habilidad utilizada dentro de estas mismas herramientas estás pueden ejemplificarse en el código fuente y lo apartamos dentro del repositorio de GitHub para que se pueda observar:

https://github.com/emmanuel-rosillo/POO/tree/main/POOp9

Conclusión:

En esta práctica pudimos descubrir varios lenguajes de marcados los cuales nos permitieron integrar documentación dentro de los repositorios para las aplicaciones que sean creadas en los diferentes lenguajes de programación de esta forma podemos tener una manera interactiva de explicar cómo funciona un código con las diferentes herramientas que se tocaron en esta práctica y así ampliar y enriquecer el código que se está elaborando para la solución de un problema.