

## TRABAJO PRÁCTICO Nº 2 – CLASES Y OBJETOS

Unidades 3 y 4 – Objetos y Clases

PROGRAMACIÓN 2 - 2022 – 2do cuatrimestre

TECNICATURA UNIVERSITARIA EN DESARROLLO WEB

### EL TRABAJO PRÁCTICO Nº 2 TIENE POR OBJETIVO QUE EL ALUMNO

- Aplique y refuerce los conceptos fundamentales y relacionados a la Programación Orientada a Objetos.
- Sea capaz de interpretar y traducir correctamente los diagramas de Clases en código Python.

### CONDICIONES DE ENTREGA

- El Trabajo Práctico deberá ser:
  - Realizado en forma individual o en **grupos de NO más de 4 (cuatro) alumnos.**
  - Cargado en la sección del Campus Virtual correspondiente, en un archivo ZIP o RAR con las soluciones a cada ejercicio que requiera una solución en código Python, y en un archivo Word las respuestas a las preguntas que no requieran una solución en código Python. Las soluciones para los ejercicios que requieran de código Python debe estar contenidas cada una en un archivo .py distinto.
  - En caso de realizar el Trabajo Práctico en grupo, deberá indicarse el apellido y nombre de los integrantes del mismo. Todos los integrantes del grupo deben realizar la entrega en el campus y deberá agregarse al comprimido con las soluciones un archivo *integrantes.txt* con los nombres de los participantes.
  - Entregado antes de la fecha límite informada en el campus.
- El Trabajo Práctico será calificado como Aprobado o Desaprobado.

- Las soluciones del alumno/grupo deben ser de autoría propia. De encontrarse soluciones idénticas entre diferentes grupos, dichos trabajos prácticos serán clasificados como **DESAPROBADO**, lo cual será comunicado en la devolución.

## EJERCICIOS:

Las siguientes consignas estarán relacionadas al universo de la película de Pixar / Disney, Monsters Inc, donde el alumno deberá modelar las distintas entidades de dicho universo. A continuación, resuelva:

1. Dado el siguiente diagrama de clases:

Monstruo
<pre>&lt;&lt;Atributos de clase&gt;&gt; maxEnergia: int &lt;&lt;Atributos de instancia&gt;&gt; nombre: string especie: string energía: int</pre>
<pre>&lt;&lt;Constructores&gt;&gt; Monstruo(nom, esp: string) &lt;&lt;Comandos&gt;&gt; establecerNombre(nom: string) establecerEspecie(esp: string) establecerEnergia(ene: int) asustar() &lt;&lt;Consultas&gt;&gt; obtenerNombre(): string obtenerEspecie(): string obtenerEnergia(): int</pre>

Genere la clase Monstruo, conteniendo los atributos y servicios mencionados en el diagrama de clases anterior.

- a. El atributo de clase **maxEnergía** debe tener un valor de **100**.
- b. El valor inicial de **energía** debe ser igual a **maxEnergía**.
- c. El método asustar debe decrementar la energía del monstruo en **10 unidades**.

2. Se agrega el siguiente diagrama, que intenta representar a los humanos del referenciado universo:

Humano
<<Atributos de clase>> especie: string <<Atributos de instancia>> nombre: string estadoAsustado: boolean
<<Constructores>> Humano(nom: string) <<Comandos>> establecerNombre(nom: string) establecerEstadoAsustado(est: boolean) <<Consultas>> obtenerNombre(): string obtenerEstadoAsustado(): boolean

Habiendo analizado el diagrama, genere la clase Humano con los atributos y servicios mencionados en dicho diagrama.

- El atributo de clase **especie** debe tener valor “humano”.
  - El valor inicial de **estadoAsustado** debe ser False.
3. Una vez codificadas en Python las Clases de los puntos anteriores, instancie los objetos tal como sucede en las siguientes instrucciones:

```
sullivan = Monstruo('James P. Sullivan', 'leon')  
mike = Monstruo('Mike Wazowski', 'ciclope')  
boo = Humano('Boo')
```

- Describa el estado interno de los objetos asociados a las variables *sullivan*, *mike* y *boo*, detallando los valores de sus atributos de instancia.
- ¿Cuál es el valor del atributo de clase **especie** asociado al objeto referenciado por el identificador *boo*?

c. Si se instanciase un segundo objeto como el siguiente:

```
sullivan2 = Monstruo('James P. Sullivan', 'leon')
```

- i. ¿Los identificadores *sullivan* y *sullivan2* hacen referencia al mismo objeto? ¿o son objetos idénticos completamente distintos?
- ii. ¿Son objetos equivalentes? Explique que significa que dos objetos lo sean.
- iii. ¿Los objetos ligados a *sullivan* y *sullivan2* comparten la misma posición de memoria?

4. Modificar la clase Monstruo de acuerdo a lo especificado al siguiente diagrama:

Monstruo
<pre>&lt;&lt;Atributos de clase&gt;&gt; maxEnergia: int &lt;&lt;Atributos de instancia&gt;&gt; nombre: string especie: string energía: int</pre>
<pre>&lt;&lt;Constructores&gt;&gt; Monstruo(nom, esp: string) &lt;&lt;Comandos&gt;&gt; establecerNombre(nom: string) establecerEspecie(esp: string) establecerEnergia(ene: int) asustar(hum: Humano) divertir(hum: Humano) &lt;&lt;Consultas&gt;&gt; obtenerNombre(): string obtenerEspecie(): string obtenerEnergia(): int</pre>

- a. El comando **asustar** debe, además de decrementar la energía del monstruo en **10 unidades**, cambiar el valor del atributo **estadoAsustado** de *hum* a **True**.
- b. El comando **divertir** debe decrementar la energía del monstruo en **20 unidades**, cambiando el valor del atributo **estadoAsustado** de *hum* a **False**.

- c. Una vez codificadas las modificaciones sobre la clase del punto anterior, ejecute el siguiente programa Python y verifique que se cumpla la salida esperada.

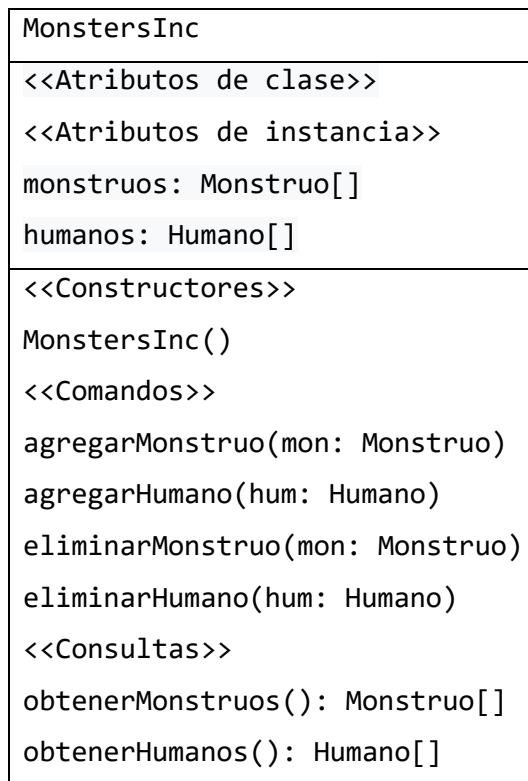
El programa:

```
sullivan = Monstruo('James P. Sullivan', 'leon')
mike = Monstruo('Mike Wazowski', 'ciclope')
boo = Humano('Boo')
print(sullivan.obtenerEnergia())
print(mike.obtenerEnergia())
print(boo.obtenerEstadoAsustado())
sullivan.asustar(boo)
print(sullivan.obtenerEnergia())
print(mike.obtenerEnergia())
print(boo.obtenerEstadoAsustado())
mike.divertir(boo)
print(sullivan.obtenerEnergia())
print(mike.obtenerEnergia())
print(boo.obtenerEstadoAsustado())
```

La salida esperada es la siguiente:

```
100
100
False
90
100
True
90
80
False
```

5. Construya una clase MonstersInc como la especificada en el siguiente diagrama de clases:



- a. Los comandos **agregarMonstruo** y **agregarHumano** deben agregar un objeto de tipo Monstruo o un objeto de tipo Humano a las listas *monstruos* y *humanos* de dicho objeto, respectivamente.
  - b. Los comandos **eliminarMonstruo** y **eliminarHumano** deben eliminar un objeto de tipo Monstruo o un objeto de tipo Humano de las listas *monstruos* y *humanos* de dicho objeto, respectivamente.
  - c.
6. Construya un programa, utilizando la clase proveedora MonstersInc que, a partir de la entrada del usuario permita:
- a. Llamar a los métodos **agregarMonstruo** y **agregarHumano**.
  - b. Filtrar a los monstruos con un nivel de energía específico por debajo de algún valor.
  - c. Filtrar a los humanos por sus dos posibles estados: asustado y no asustado.

Para la construcción de dicho programa crear una clase de nombre `TesterMonstersInc` que actúe como cliente de la clase proveedora `MonstersInc`, cuyo único servicio sea de nombre ***main***, sin parámetros, que ejecute los puntos descritos anteriormente. A continuación, un ejemplo de como dicho programa puede ser construido:

```
class TesterMonstersInc:
    def main():
        # Solución de los puntos 6.a., 6.b, 6.c, ...

if __name__ == '__main__':
    testerMonstersInc = TesterMonstersInc()
    testerMonstersInc.main()
```