

## PRÁCTICO DE PROGRAMACIÓN ORIENTADA A OBJETOS

### UNIDAD 3

#### Requerimientos generales

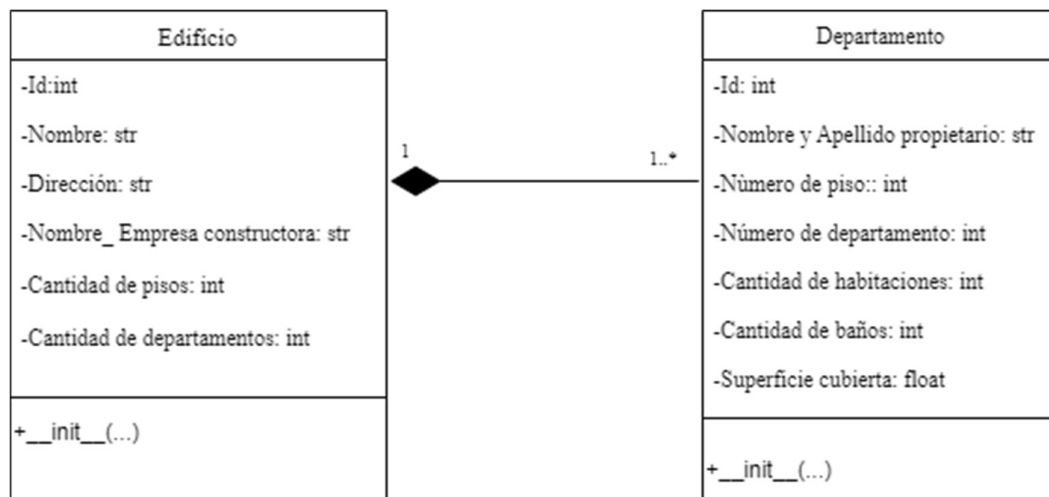
La resolución de cada ejercicio debe:

- Usar **clases independientes**.
- Incluir manejo de **excepciones**.

#### Ejercicio Nº 1

##### Composición

Dado el siguiente diagrama.



- Definir las clases **Edificio** y **Departamento** con sus atributos y métodos necesarios.
- Definir una **Gestor de Edificios**, para instanciar los objetos desde el archivo "EdificioNorte.csv".
- Implementar un menú de opciones que permita:
  - Dado el nombre de un edificio mostrar: Nombre y apellido de los propietarios de cada uno de los departamentos del edificio.
  - Mostrar la superficie total cubierta de un edificio.
  - Dado el nombre de un propietario, mostrar la superficie total cubierta de su departamento e indicar que porcentaje representa del total de la superficie cubierta del edificio.
  - Dado un número de piso: contar y mostrar la cantidad de departamentos que tienen 3 dormitorios y más de un baño.

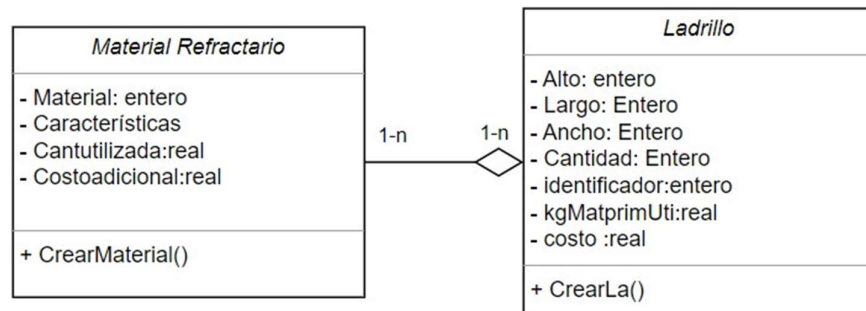
## Ejercicio Nº 2

### Agregación

La fábrica de Ladrillos Nuestro Hogar, ha implementado la elaboración de un novedoso sistema de construcción utilizando ladrillos refractarios y tradicionales. En su elaboración utilizan materiales reciclados y los tradicionales cada uno con sus características y costos de elaboración vinculado.

Para eso nos detalla sus componentes y dimensiones para cada caso. Sus medidas se mantienen estándares en todos sus ejemplares. Considerando este detalle fabrican ladrillos con las siguientes medidas, largo 25 cm, ancho 15 cm y alto 7cm. El ladrillo fabricado puede o no tener material reciclado en su fabricación, en el caso que lo contenga tiene el ladrillo un costo adicional de fabricación

Se solicita construir la relación existente y el diagrama de clase correspondiente a fin de poder registrar la fabricación de estos materiales de construcción.



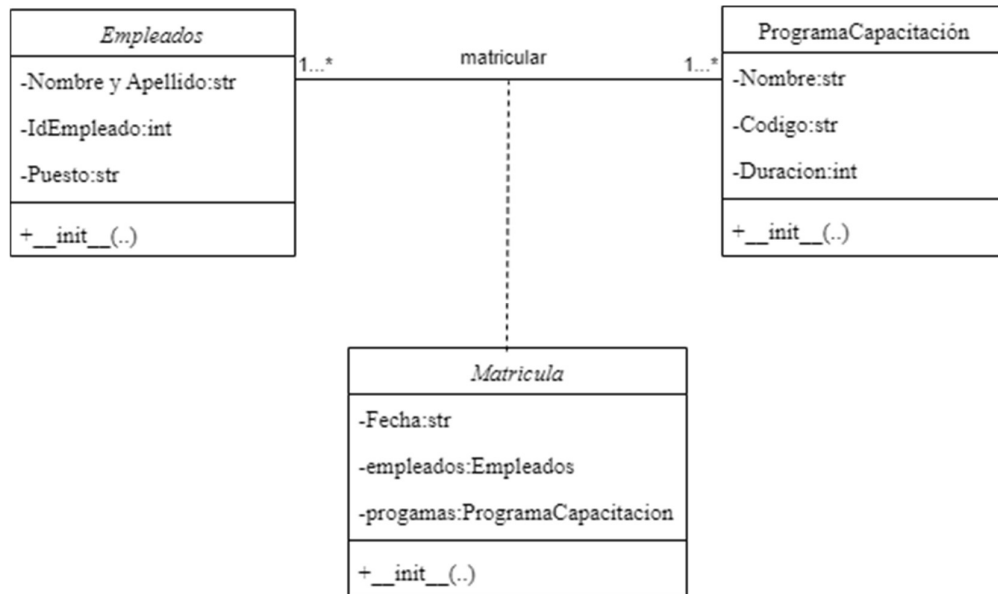
- Definir las clases involucradas en el diagrama, con sus atributos y métodos correspondientes.
- Definir una GestorMaterial, que permita almacenar los materiales. El gestor debe basarse en una lista del lenguaje Python.
- Definir una GestorLadrillo, que permita almacenar los ladrillos. El gestor debe basarse en una lista del lenguaje Python.
- Implementar un programa con un menú de opciones que permita:
  - Para un identificador de ladrillo ingresado por teclado: Detallar costo y característica del material solicitado.
  - Mostrar para cada ladrillo el costo total de fabricación del pedido.
  - Para cada uno de los ladrillos fabricados mostrar el detalle asociado con el siguiente formato:

N° identificador	Material	Costo asociado
xxxxxxxxx	xxxxxxxxx	xxxxxxxxxxxxx
xxxxxxxxx	xxxxxxxxx	xxxxxxxxxxxxx
xxxxxxxxx	xxxxxxxxx	xxxxxxxxxxxxx

### Ejercicio Nº 3

#### Asociación

Dado el siguiente diagrama:



- Definir las clases *Empleados*, *ProgramaCapacitacion* y *Matricula* con sus atributos y métodos necesarios.
- Definir una clase *GestorEmpleados*, que permita registrar los empleados a través de una lista.
- Definir una clase *GestorProgramaCapacitacion*, que registre y gestione a través de una lista los programas a tomar.
- Definir una clase *GestorMatricula*, que permita manejar las matrículas.
- Implementar un programa con un menú de opciones que:
  - Dado el Id del empleado, informe la duración de todos los programas de capacitación en los que está matriculado.
  - Dado el nombre de un programa de capacitación, muestre el/los empleados matriculados en el mismo.
  - Informar aquellos Empleados que no han sido matriculados en ningún programa de capacitación.

## **Ejercicio Nº4:**

### **Herencia y métodos con ligadura dinámica**

#### **Descripción del Sistema**

Una compañía editorial produce publicaciones para la venta, estas pueden ser, libros impresos y audio-libros en discos compactos.

De todas las publicaciones se conoce: su título, categoría (comedia, drama, autoayuda etc.), y su precio base.

De un libro impreso se conoce, además: nombre del autor, fecha de edición, y cantidad de páginas.

De un CD se conoce, además: el tiempo de reproducción en minutos, y el nombre del narrador.

El importe de venta de cada publicación se calcula en función del precio base y de sus características. Para ello se deben considerar las siguientes reglas de negocio:

Importe de venta de un libro impreso es el precio base, menos el 1% por cada año de antigüedad (año actual – año de edición), este porcentaje se calcula sobre el precio base.

Importe de venta de un audio-libro es el precio base, más 10% por gastos de regalía, porcentaje se calculan sobre el precio base.

El analista de la editorial le solicita a usted que desarrolle una aplicación con las siguientes restricciones:

- a- Definir la jerarquía de clases con los métodos correspondientes a cada clase de la narrativa dada.
- b- Almacenar en una colección tipo Lista definida por el programador, las publicaciones de la editorial.
- c- Implementar un programa principal con un menú de opciones que permita testear las siguientes acciones:
  - 1. Agregar publicaciones a la colección
  - 2. Dada una posición de la lista: Mostrar por pantalla qué tipo de publicación se encuentra almacenada en dicha posición (usar la función `isinstance()`).
  - 3. Mostrar la cantidad de publicaciones de cada tipo.
  - 4. Recorrer la colección y mostrar para todas las publicaciones Título, categoría e importe de venta.

## Ejercicio N°5:

### Interfaces - Excepciones

Defina una interface con los siguientes métodos:

- a- insertarElemento: para insertar un objeto en una posición determinada en una colección, teniendo en cuenta el manejo de excepciones cuando la posición donde se vaya a insertar no sea válida.
- b- agregarElemento: para agregar un elemento al final de una colección.
- c- mostrarElemento: dada una posición de la colección, mostrar los datos del elemento almacenado en dicha posición si esa posición es válida, en caso de que no sea válida lanzar una excepción que controle el error.

## Ejercicio N°6:

### Herencia y archivos JSON

#### **Descripción del Sistema**

Una Empresa dedicada a la venta de equipos de calefacción ofrece dos tipos de calefactores: eléctricos y a gas natural. Para ello cuenta con un archivo, "calefactores.json", con la estructura necesaria para almacenar calefactores eléctricos y calefactores a gas natural.

Como estrategia de venta, a los equipos menos vendidos, los promociona con un porcentaje de descuento si la venta se realiza de contado.

De todos los calefactores se registra: marca, modelo, país de fabricación, precio de lista, forma de pago (contado o en cuotas), cantidad de cuotas (será =1 si el pago es de contado) y promoción (Si/No).

**De un calefactor eléctrico** se registra, además: la potencia máxima por ej.:(500 watts).

**De un calefactor a gas natural** se registra, además: matricula por ej.:(GN01-00001-06-057), y calorías por ej.:(4000kilocalorias/m3).

El importe de venta de cada equipo se calcula en función del precio de lista, de sus características, la forma de pago, y si está o no en promoción.

Para ello se deben considerar las siguientes reglas de negocio:

**Para todos los calefactores que estén en promoción el Importe de venta:** precio de lista - 15%.

**Importe de venta de los Calefactores a gas natural:** precio de lista más: el 1% si las calorías superan los 3000kilocalorias/m3, más el 40% si el pago se realiza en cuotas.

Todos los porcentajes se calculan sobre el precio de lista.

**Importe de venta de los Calefactores eléctricos es el precio de lista, más:** el 1% si la potencia máxima supera los 1000watts+ 30% si el pago se realiza en cuotas.

Todos los porcentajes se calculan sobre el precio de lista.

El analista le solicita a usted que desarrolle una aplicación con las siguientes restricciones.

La colección de **calefactores** debe implementarse usando una lista definida por el programador, donde los nodos deberán almacenar calefactores.

La lista deberá proveer los métodos y atributos necesarios para que sea un iterable, e implementar la interface del **ejercicio 5**.

Para cumplir con lo solicitado por el analista, usted deberá:

- a- Definir la jerarquía de clases correspondiente a la problemática planteada.
- b- Leer y procesar el archivo “calefactores.json” para almacenar en la lista los calefactores de la empresa.
- c- Implemente un programa que a través de un menú de opciones permita:
  - 1. Insertar un calefactor en la colección en una posición determinada.
  - 2. Agregar un calefactor a la colección (solicitar el tipo de calefactor, y luego los datos que correspondan).
  - 3. Dada una posición de la Lista: Mostrar por pantalla qué tipo de objeto se encuentra almacenado en dicha posición.
  - 4. Mostrar marca, modelo y kilocalorías del calefactor a gas natural de menor precio.
  - 5. Dada una marca de calefactor eléctrico, mostrar modelo, potencia y precio de lista de todos los calefactores de esa marca.
  - 6. Mostrar marca, modelo, país de fabricación e importe de venta de todos los calefactores que están en promoción
  - 7. Almacenar los objetos de la colección Lista en el archivo “calefactores.json”.

### **Ejercicio N°7:**

#### **Herencia Múltiple, polimorfismo**

En el ámbito universitario los agentes que ahí trabajan se pueden clasificar de la siguiente manera: Docentes, investigadores, docente investigador y personal de apoyo.

La información del personal universitario se encuentra almacenada en un archivo denominado “personal.json”.

El Centro de cómputos de la UNSJ, lo contrata a usted como programador, para que desarrolle una aplicación que permita procesar el archivo “json” donde guarda la información del personal que trabaja en la misma. Para ello le provee la siguiente narrativa, que ha sido elaborada por el analista funcional:

## **Narrativa**

De todo el personal se conoce la siguiente información: cuil, apellido, nombre, sueldo básico y antigüedad.

Si es un docente se registra además carrera en la que dicta clases, cargo y cátedra.

Si es personal de apoyo se registra además categoría.

Si es un investigador se registra además área de investigación y tipo de investigación

Si es un docente investigador se registra además carrera en la que dicta clases, cargo, cátedra, área de investigación y tipo de investigación, categoría en el programa de incentivos de investigación, importe extra por docencia e investigación.

Para cumplir con las tareas solicitadas por el analista, usted deberá:

- a- Definir la jerarquía de clases correspondiente a la narrativa dada.
- b- Almacenar en una colección tipo Lista definida por el programador, los agentes de la Universidad, obteniendo los datos del archivo "personal.json", la misma deberá implementar la interfaz definida en el **ejercicio 5**. Los nodos de la lista, serán referencias a objetos que representen objetos de la clase base de la jerarquía.
- c- Implementar un programa principal con un menú de opciones que permita testear las siguientes acciones:
  - 1. Insertar a agentes a la colección.
  - 2. Agregar agentes a la colección.
  - 3. Dada una posición de la lista: Mostrar por pantalla que tipo de agente se encuentra almacenado en dicha posición.
  - 4. Ingresar por teclado el nombre de una carrera y generar un listado ordenado por nombre con todos los datos de los agentes que se desempeñan como docentes investigadores.
  - 5. Dada un área de investigación, contar la cantidad de agentes que son docente investigador, y la cantidad de investigadores que trabajan en esa área.
  - 6. Recorrer la colección y generar un listado que muestre nombre y apellido, tipo de Agente y sueldo de todos los agentes, ordenado por apellido.
  - 7. Dada una categoría de investigación (I, II, III, IV o V), leída desde teclado, listar apellido, nombre e importe extra por docencia e investigación, de todos los docentes investigadores que poseen esa categoría, al final del listado deberá mostrar el total de dinero que la Secretaría de Investigación debe solicitar al Ministerio en concepto de importe extra que cobran los docentes investigadores de la categoría solicitada.
  - 8. Almacenar los datos de todos los agentes en el archivo "personal.json"

**Reglas de negocio para el cálculo del sueldo:**

Por cada año de antigüedad el sueldo se incrementa en un porcentaje sobre el sueldo básico, por ejemplo: si tienen 5 años de antigüedad el sueldo sería sueldo básico + 5%(básico).

Porcentaje por cargo: 10 % si el cargo es simple, 20% si el cargo es semiexclusivo, 50% si el cargo es exclusivo.

Porcentaje por categoría: 10% si la categoría es de 1 a 10, 20 % si la categoría es de 11 a 20, 30% si la categoría es de 21 a 22.

Todos los porcentajes se calculan sobre el sueldo básico.

$\text{sueldoPersonalDeApoyo} = \text{Sueldo Básico} + \% \text{antigüedad} + \% \text{por categorías}$

$\text{sueldoDocente} = \text{Sueldo Básico} + \% \text{antigüedad} + \% \text{por cargo}$

$\text{sueldoInvestigador} = \text{Sueldo básico} + \% \text{antigüedad}$

$\text{sueldoDocenteInvestigador} = \text{sueldoDocente}() + \text{importe extra por docencia e investigación.}$

**Ejercicio N°8****Usar la narrativa del Ejercicio N°7**

Se ha agregado nuevos requerimientos al sistema, y usted como programador, debe darles solución:

El sistema prevé que los agentes serán administrados a través de una colección. Existen dos interfaces distintas para operar con agentes, la interfaz del Tesorero, que solamente puede acceder a los gastos que la universidad tiene en concepto de sueldos, para ello, dado un número de documento, podrá consultar el gasto de sueldos para el agente al que pertenece dicho número de documento.

La interfaz del Director, permite modificar el sueldo básico de todos los agentes, el porcentaje que se paga por cargo a un docente, el porcentaje que se paga por categoría a un personal de apoyo, y el porcentaje extra que se paga a un docente investigador; para ello debe proveerse el número de documento del agente, y el valor que corresponda según lo que se quiera modificar.

Usted deberá crear las interfaces mencionadas con las funcionalidades solicitadas.

**class ITesorero (Interface)**

```
def gastosSueldoPorEmpleado ( dni):  
    pass
```

**class IDirector (Interface)**

```
def modificarBasico(dni, nuevoBasico):  
    pass
```



```
def modificarPorcentajeporcargo(dni, nuevoPorcentaje):  
    pass
```

```
def modificarPorcentajeporcategoría(dni, nuevoPorcentaje):  
    pass
```

```
def modificarImporteExtra(dni, nuevoImporteExtra):  
    pass
```

En el programa principal agregue una opción para el Tesorero y un menú de opciones para Director. Para ello deberá autenticar al usuario que quiere acceder, si es tesorero, accederá a las funcionalidades de tesorero y si es director, accederá a las funciones de director. La autenticación del usuario se hace pidiendo nombre de usuario y contraseña.

Usuario/contraseña para Tesorero: uTesoreso/ag@74ck

Usuario/contraseña para Director: uDirector/ufC77#!1