**4) Una empresa de servicios guarda es una lista las tareas que debe realizar cada empleado. La estructura es la siguiente:**

EmpresaLista:

* Código Empleado (Clave ordenamiento)
* Cantidad de Tareas
* Cola de Tareas

ColaTarea:

* Área Solicitante
* Descripción

Realizar procedimientos para este TDA que:

* Permita Ingresar una nueva tarea en el empleado que tenga menos tareas.
* Permita Procesar las tareas de los empleados 15 a 30, sacándolas de la cola.

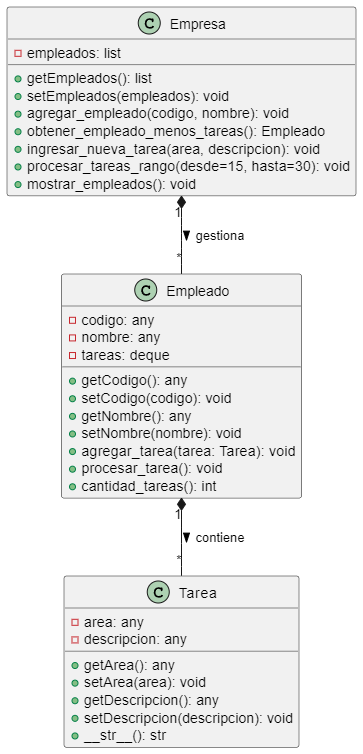
Aclaración: Cuando las estructuras están ordenadas por algún campo no se deben recorrer de más.

# Desarrollo

Se procedió a realizar el *servicios.puml* donde tenemos las clases de: Tarea, Empleado y Empresa. Adicional a esto, se les asigno sus atributos y sus métodos que relacionamos de la siguiente manera:

* **Empleado** contiene una colección de **Tarea.**
* **Empresa** administra una lista de **Empleado.**

Como podemos visualizar en la siguiente imagen:

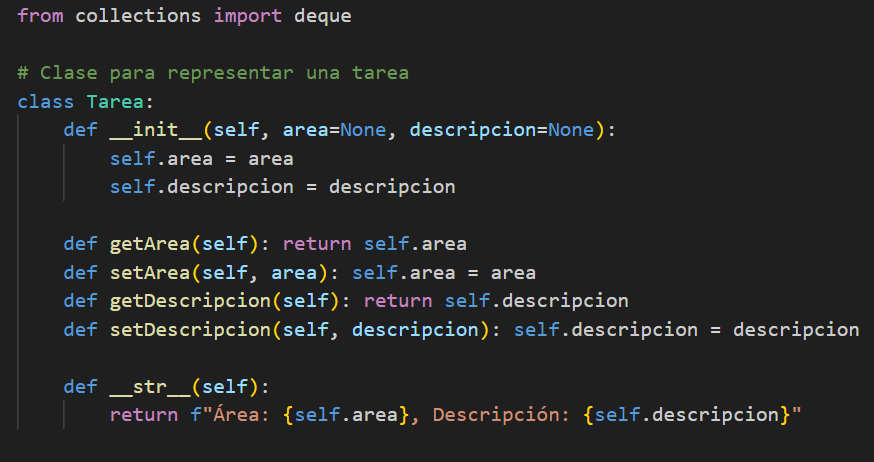


Para el código de Python, utilizamos la Clase Tarea:  
Esta clase representa una tarea asignable a un empleado. Cada tarea contiene dos atributos:

* area: área de la empresa responsable (como Sistemas, RRHH, etc.)
* descripcion: detalle específico de la tarea

Además, incluye:

* Constructor por defecto y personalizado.
* Métodos get y set para acceder y modificar atributos.
* Método \_\_str\_\_para representar la tarea como una cadena legible.



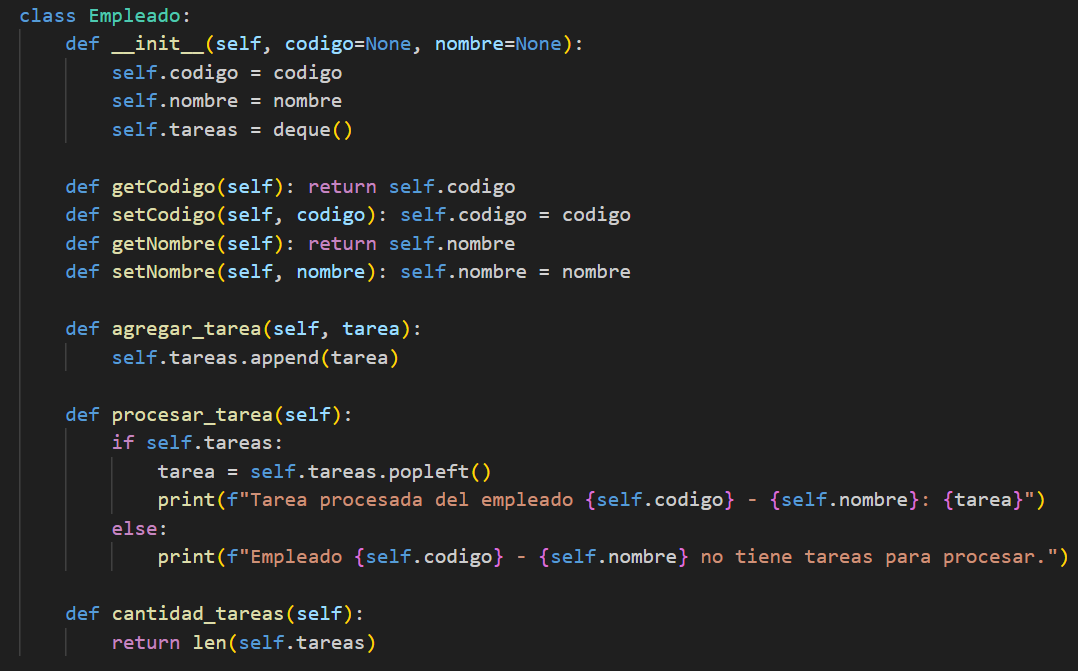
**Clase Empleado**

Esta clase modelo a cada empleado con:

* codigo: identificador numérico único.
* nombre: nombre del empleado.
* tareas: una cola (deque) que almacena las tareas asignadas, permitiendo procesarlas en orden FIFO (primero en entrar, primero en salir).

Incluye los siguientes métodos:

* agregar\_tarea(tarea): añade una tarea al empleado.
* procesar\_tarea(): elimine la tarea más antigua de la cola y la muestra como procesada.
* cantidad\_tareas(): retorna la cantidad de tareas actuales.
* Métodos get y set para sus atributos.



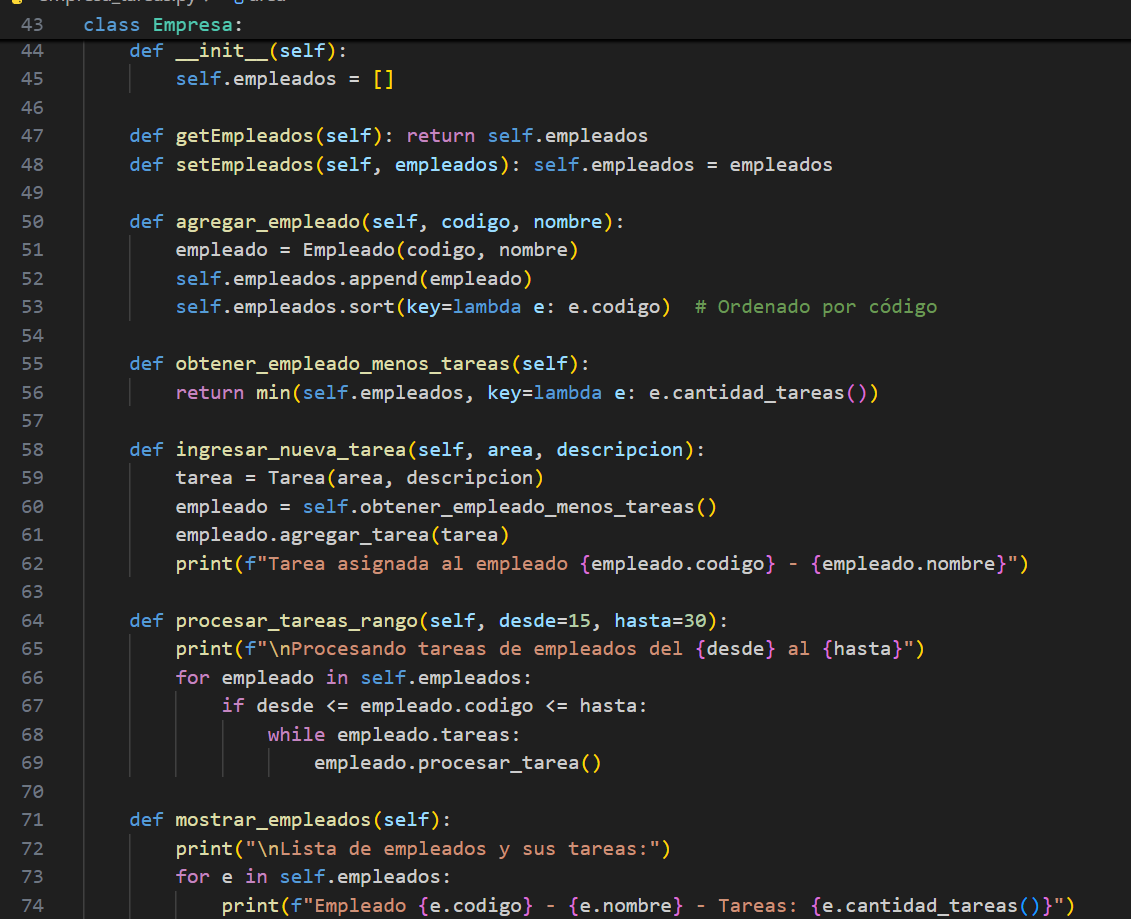
**Clase Empresa**

Contiene la lógica general para administrar empleados y tareas. Sus principales responsabilidades son:

* Mantener una lista de empleados.
* Agregar nuevos empleados, ordenados por código.
* Asignar tareas automáticamente al empleado con menos tareas.
* Procesar todas las tareas de los empleados en un rango de códigos dado.
* Mostrar un resumen de los empleados y sus tareas actuales.

Métodos destacados:

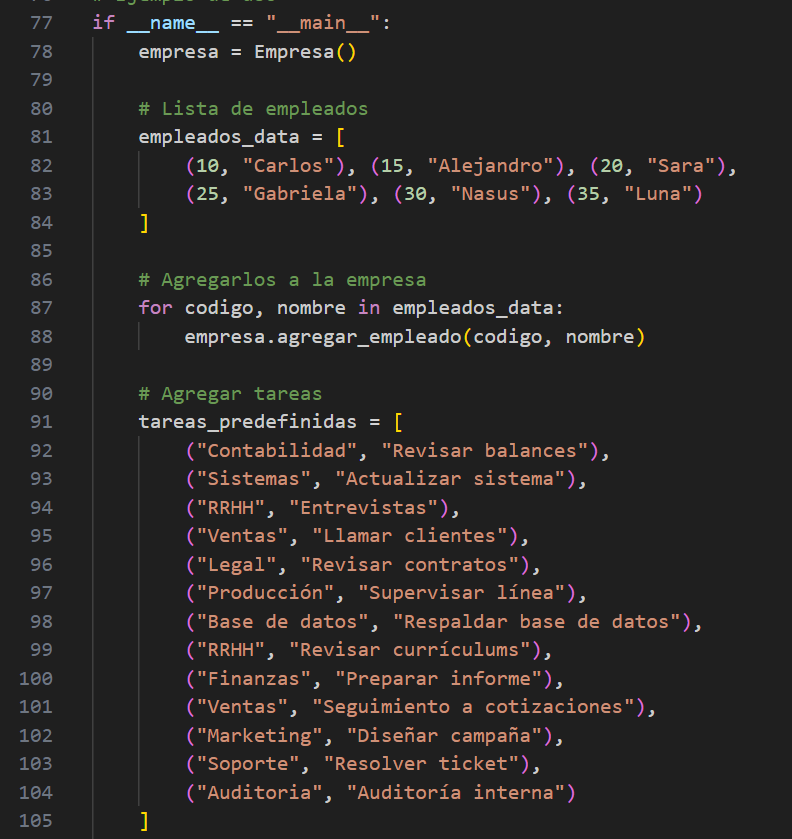
* agregar\_empleado(codigo, nombre)
* ingresar\_nueva\_tarea(area, descripcion)
* procesar\_tareas\_rango(desde, hasta)
* mostrar\_empleados()

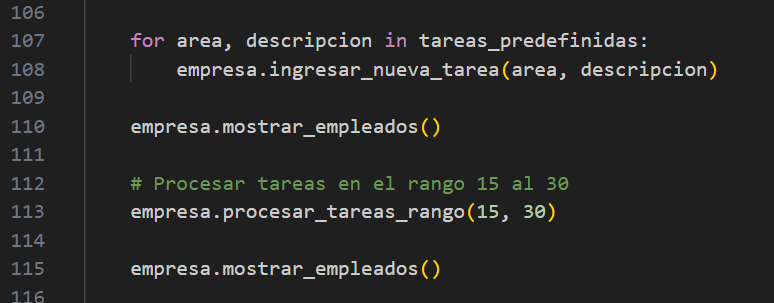


El programa comienza creando una instancia de la clase Empresa y agregando empleados con sus respectivos nombres y códigos únicos.

Asignamos Tareas, una lista predefinida de tareas está distribuida equitativamente entre los empleados. La lógica garantiza que cada nueva tarea sea asignada al empleado con la menor carga de trabajo en ese momento.

Después realizamos el Procesamiento de tareas, las tareas de los empleados cuyo código se encuentre dentro de un rango específico (por ejemplo, del 15 al 30) son procesadas automáticamente. Esto simula que esos empleados completan sus tareas.





Al final de todo tenemos una interacción donde el programa permite que el usuario continúe ingresando tareas manualmente por consola, las cuales también serán asignadas al empleado con menos tareas.

