PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

**IIC2233 - Programación Avanzada**
**$2^o$ semestre 2016**

# Actividad 07

## Metaclases

### Problema

Tim Peters, gran exponente en la programación, y desarrollador principal de CPython dice:"Metaclasses are deeper magic that 99 % of users should never worry about. If you wonder whether you need them, you don't (the people who actually need them know with certainty that they need them, and don't need an explanation about why)."

Claramente, esto significa que el concepto de metaclases no es para nada trivial, por lo que decidiste estudiar el tema con mayor profundidad. Tras muchas horas de esfuerzo lo lograste, pero para asimilar tanta materia nueva tuviste que eliminar muchos recuerdos de tu mente, entre ellos, como crear correctamente las clases (y otras cosas, como recuerdos familiares, amigos, etc. pero al menos entendiste la materia :D).

Con tus grandes conocimientos de metaclases, lograste impresionar a una famosa empresa llamada *Feisbuk*, que decidió contratarte. Esta te solicitó el modelamiento de su empresa a nivel corportativo según sus integrantes y te entregó las clases que quería que tuviese tu programa. Por suerte, ahora que dominas Metaclases ¡puedes modelar todo a partir de ellas!... A pesar de que se te haya olvidado todo lo demás.

### Instrucciones

Dentro del archivo *main.py* descargado junto a este enunciado encontrará tres clases; **Boss**, **Worker** y **Organization**. Dichas clases se encuentran funcionalmente incompletas y es deber suyo corregirlas, **solo mediante el uso de metaclases**.

Deberá crear las metaclases **MetaOrganization** y **MetaPerson** para posteriormente poder aplicarlas sobre cualquier clase, en particular, MetaOrganization sobre Organization y MetaPerson sobre Boss y Worker para completar el funcionamiento del programa.

A partir de ahora el enunciado llamará Organización a cualquier clase con metaclase MetaOrganization y Persona a cualquier clase con metaclase MetaPerson.

### Requerimientos

Cualquier clase Organización debe cumplir:

- Poseer un método para listar a sus miembros.
- Poseer un método para reemplazar al jefe actual por otro.
- Cuando se llama a una de sus instancias, se imprime la información de la instancia en cuestión; **nombre**, **jefe** y **cantidad de miembros**.

- Si al intentar instanciar una organización de clase `Organizaciony` el nombre se encuentra ocupado por alguna otra instancia de **dicha** clase `Organizacion`, la instanciación se verá denegada, retornando **None** en vez de la instancia esperada.

Cualquier clase Persona debe cumplir:

- Cuando se instancia una persona a partir de una clase Persona esta recibe un nombre, apellido y edad aleatorios.
- El atributo `organizacion` de la clase persona debe guardar el nombre de la organización, **no la instancia organización**
- Si el nombre de la clase Persona **contiene** la palabra *Boss* se le considerará como una clase Jefe. Si **contiene** la palabra *Worker* se le considerará como una clase Trabajador.
- Al momento de intentar instanciar una persona a partir de una clase Persona se debe cumplir:
  - Si la organización **no posee** un jefe y la persona **no es un jefe**, no se verá instanciada, retornando **None**.
  - Si la organización **no posee** un jefe y la persona **es un jefe**, se verá instanciada. Además, la organización le adoptará como jefe, otorgándole el método `add_members` para añadir personas a la organización.
  - Si la organización **posee** un jefe y la persona **no es un jefe**, será instanciada, pero no añadida a la organización necesariamente. Aceptar a dicha persona es deber del jefe.
  - Si la organización **posee** un jefe y la persona **es un jefe**, se reemplazará al jefe actual de la organización por el nuevo jefe mediante el método de la organización correspondiente.
- Cualquier clase Persona considerada Jefe deberá poseer un método para dar órdenes **to_order**. Puede ser simulado mediante una impresión en pantalla
- Cualquier clase Persona considerada Trabajador deberá poseer un método para trabajar **to_work**. Puede ser simulado mediante una impresión en pantalla.
- El método `add_members` recibe como parámetro un miembro a agregar a la organización. Este método debe agregarlos a la misma organización que pertenece el jefe.

## Notas

- Puede asumir que cualquier clase Persona su constructor solicitará el parámetro **organization**.
- *main.py* ofrece listas de nombres y apellidos para hacer uso de ellos cuando sea necesario.
- **IMPORTANTE**: No puede escribir sobre las clases Organization, Boss ni Worker, a excepción del código necesario para aplicar una metaclase en ellos. Cambiar algo de esas clase significará que tendrá un 1.0 (uno) en la actividad.

## To - DO

- (2.00 pts) Clases Organización e instancias de estas cumplen con los requerimientos.
- (1.00 pts) Clases Persona e instancias de estas reciben atributos nombre, apellido y edad aleatoriamente.
- (1.50 pts) Clases Persona de tipo Trabajador e instancias de estas cumplen con los requerimientos.
- (1.50 pts) Clases Persona de tipo Jefe e instancias de estas cumplen con los requerimientos.

## Entrega

- **Lugar:** GIT - Carpeta: Actividades/AC07
- **Hora:** 16:55