Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2233 — Programación Avanzada 11111100100-1

1011 de junio de 11111100100
**Actividad Sumativa**

# Actividad Sumativa 04

## I/O + Serialización

## Entrega

- **Lugar:** En su repositorio privado de GitHub, en la **carpeta** `Actividades/AS04/`
- **Hora del *push*:** 16:50

**Importante:** Antes de comenzar, comprueba que Git este funcionando correctamente en tu repositorio privado. Para esto, **sube los archivos base de la actividad de inmediato** (*add*, *commit*, *push*). Se espera que en esta actividad (así como en las demás actividades y tareas) utilices Git a lo largo de **todo tu desarrollo** como una herramienta, no sólo como un método de entrega. Es por esto que recomendamos enfáticamente que vayas subiendo tus cambios constantemente (***push***), ya que **problemas de último minuto** relacionados con la entrega y Git **no serán considerados**.

## Introducción


Estando ya en los últimos meses de clases, hay algo que aún te preocupa: ¡No conoces los intereses de los ayudantes jefes de IIC2233! Luego de ~~ver muchos memes~~ pensar en cómo solucionar esta situación, antes de que acabe el semestre, decides jugar al famoso juego **Avanzada Quién?** y así conocer un poco más a tus ayudantes jefes.

El juego de **Avanzada Quién?** se divide en diferentes partes que se encuentran en carpetas distintas. Al completar cada parte obtendrás el usuario y contraseña de un jefe, información que podrás ingresar en un programa con interfaz gráfica ya implementada (en la carpeta `Avanzada Quién`), y así descubrir quiénes son tus ayudantes jefes.

## Flujo del programa

La actividad está compuesta de tres partes diferentes, más el *bonus*. **Cada una de ellas es independiente de las demás**, por lo que puedes comenzar y terminar por la parte que prefieras. En cada sección obtendrás un **usuario** y una **clave** las cuales pueden ser ingresadas en la interfaz de **Avanzada Quién?**.

Para desarrollar cada parte de la actividad, deberás ingresar a cada una de las cuatro carpetas, donde cada carpeta representa una parte diferente de la actividad:

- **Parte *Bytes***: En esta parte deberás reparar un archivo cuyos *bytes* fueron manipulados, con el fin de obtener un archivo de imagen con los datos que necesitas del ayudante jefe coordinador **Enzo Tamburini**. Aplicando tus conocimientos de manipulación de *bytes* encontrarás el usuario y clave necesarios para conocer a Enzo.
- **Parte JSON**: En esta parte deberás extraer información de los ayudantes de la sección de **Tareas**, donde se encuentra la ~~líder suprema~~ jefa de tareas **Daniela Concha**. Aplicando tus conocimientos en serialización de **JSON** encontrarás el usuario y clave necesario para conocer a Dani.
- **Parte Pickle**: En esta parte deberás extraer información de los ayudantes de la sección de **Docencia**, donde se encuentran los datos que necesitas del ayudante jefe de docencia **Dante Pinto**. Aplicando tus conocimientos de serialización mediante `pickle` encontrarás el usuario y clave necesario para conocer a ~~Dr. Pinto~~ Dante.
- ***Bonus*:** En esta parte deberás reparar un conjunto de archivos cuyos *bytes* fueron manipulados, con el fin de obtener un módulo de Python que al ejecutarlo te dará los datos que necesitas del ayudante jefe híbrido **Benjamín Martínez**. Aplicando tus conocimientos de manipulación de *bytes* encontrarás el usuario y clave necesarios para conocer a Benja.

El uso de la interfaz gráfica **no será evaluado**. Al ejecutar `main.py` en la carpeta `Adivina Quien` aparece una interfaz donde se puede ingresar la información descubierta, pero no es obligatorio usarla. Su intención es hacer más dinámica y entretenida la actividad. **No debes modificar la interfaz gráfica**.

Si usas VS Code para ejecutar la actividad, como está dividida por carpetas con diferentes archivos, recuerda abrir la carpeta correspondiente por parte de la actividad, de otra forma tendrás problemas para abrir y cerrar archivos.

## Parte *Bytes*

La información del ayudante jefe coordinador se encuentra corrompida en un archivo cuyos *bytes* fueron manipulados. Tendrás que reparar dicho archivo para recuperar los datos del ayudante, completando la función `reparar_usuario` que se encuentra dentro del archivo `reparar_bytes.py`.

- `def reparar_usuario(ruta):` Esta función recibe la ruta del archivo a reparar, lee el archivo como *bytes* y mediante un algoritmo los modifica para escribir los *bytes* originales en el archivo `user_info.bmp`. El algoritmo recorre **segmentos de 32 *bytes* contiguos** y para cada uno sabes que:
  1. El primer *byte* corresponde a un entero que puede ser 1 o 0, los siguientes 16 *bytes* pertenecen al archivo original y los últimos 15 **NO** pertenecen al archivo original.
  2. Si el primer *byte* es un 1, significa que los siguientes 16 *bytes* han sido invertidos, mientras que si es un 0, entonces están en el orden original. Por invertido, se refiere a que si la secuencia es 1 2 3 4, entonces la original es 4 3 2 1.

3. Es necesario extraer los *bytes* originales al archivo en el orden que corresponda. Luego se avanza al siguiente segmento de 32 *bytes*.

Si se concatenan las porciones de *bytes* extraídas, respetando el orden de los segmentos de 32 *bytes*, se obtiene el contenido completo que se escribe en el archivo de imagen `user_info.bmp`. Al abrir la imagen, si se aplicó el algoritmo correctamente, se revelarán los datos del ayudante jefe.

## Parte JSON

En esta sección obtendrás el usuario y clave de la jefa de tareas a partir de los datos de los ayudantes de esta área. La información de los ayudantes se encuentra encriptada en el archivo `tareas.json` y es tu trabajo **corregirla** e **instanciarla**. Cada ayudante posee un **usuario**, un **cargo**, un **pokemon** favorito, una **pizza** favorita y una **serie** favorita, y los posibles cargos para cada uno son: Full Tarea, Híbrido Tarea y Jefe Tareas.

Debes completar el archivo `tareas.py`, el cual contiene lo siguiente:

- Clases:
  - `class AyudanteTareo`: Esta clase modela a los ayudantes de tareas.
    - `def __init__(self, usuario, cargo, pokemon, pizza, serie)`: Recibe información de los ayudantes y la almacena como atributos para la instancia. **Viene implementado**
    - `def __repr__(self)`: Permite una representación útil de la clase y **viene implementado**
- Funciones:
  - `def desencriptar(string)`: Decodifica un *string* encriptado y **viene implementada**. Debe usarse para desencriptar el contenido de `tareas.json`.
  - `def cargar_ayudantes(ruta)`: Recibe la ruta al archivo `tareos.json` y **debes completarla**. Debe retornar una **lista** de **instancias** de `AyudanteTareo` cargadas desde el archivo y desencriptadas. Los detalles de esto se explican a continuación.

El archivo `tareas.json` contiene un diccionario donde las llaves corresponden a los **usuarios** de los ayudantes, y el valor corresponde a una lista que contiene el resto de la información del ayudante correspondiente: su **cargo**, **pokemon favorito**, **pizza favorita** y **serie favorita**. Si el ayudante no tiene preferencia en alguna categoría aparece un `str` vacío. El diccionario se verá de la siguiente forma:

```
diccionario[usuario] = [cargo, pokemon, pizza, serie]
```

El objetivo es extraer las entidades y cargarlas como instancias de `AyudanteTareo` para poder leerlos. Sin embargo, los valores de los atributos se encuentran **encriptados** y debes utilizar la función `desencriptar` para obtener su valor correcto.

Una vez cargadas las instancias, estas se imprimirán en pantalla. Con esto podrás buscar la información de la legendaria **Daniela Concha**, cuyo usuario es `lily416`. Su **clave** se obtiene combinando en un *string* su **pokemon favorito** y su **serie favorita**, de la forma:

```
clave = f"{pokemon}#{serie}"
```

## Parte Pickle

Para encontrar los datos del ayudante jefe de docencia, deberás completar las funciones en `decodificador_docencia.py` utilizando la biblioteca de serialización `pickle`. Para probar los resultados, puedes ejecutar el archivo `docencia.py` que **viene implementado**.

Esta etapa está sub-dividida en **carga** y **guardado**. En la **carga** se encuentra la clase `EquipoDocencia` y la función `cargar_instancia`. Lo que debes hacer en carga es la personalización de la deserialización de una instancia de `EquipoDocencia` y cargar el archivo completando la función `cargar_instancia`.

Por otro lado, el **guardado** incluye la clase `Ayudante`, **que no debes modificar**; la clase `AyudanteJefe`, que hereda de `Ayudante`; y la función `guardar_instancia`. En **guardado**, debes personalizar la serialización de la clase `AyudanteJefe` y completar la función `guardar_instancia` para guardar la nueva instancia de `EquipoDocencia`. Recomendamos que desarrolles esta parte de la actividad en el orden que se presenta lo que debes realizar. A continuación se detalla lo que debes completar:

- Etapa de carga:
  - `class EquipoDocencia`: Almacena todos los ayudantes de IIC2233 relacionados con docencia y deberás instanciarla con la función `def cargar_instancia`. **Debes modificar uno de sus métodos**.
    - `def __init__(self)`: Los atributos son: `ayudantes_normales` (un objeto de tipo `list` que contiene instancias de `Ayudante` y/o `AyudanteJefe`) y `ayudante_jefe` (que corresponde a una instancia de `AyudanteJefe` o `None`). **No debes modificar este método.**
    - `def __setstate__(self, estado)`: En este método debes personalizar la deserialización que hará `pickle`. Primero debes recordar que el argumento `estado` de este método es un diccionario cuyas llaves son los nombres de cada atributo y el contenido es el valor del atributo correspondiente.

      En este método se debe encontrar una instancia de `AyudanteJefe` al interior de la lista de ayudantes (almacenada bajo la llave `ayudantes_normales`). Para encontrarla, debes recordar que esta lista contiene instancias de la clase `Ayudante` y `AyudanteJefe` y ambas cuentan con un atributo llamado `cargo` que será igual a `"Jefe"` sólo si la instancia es de la clase `AyudanteJefe`.

      Una vez encontrado el `AyudanteJefe`, debes sacar la instancia de la lista y asignarlo al atributo `ayudante_jefe` de forma que la lista final en `ayudantes_normales` contenga las instancias de todos los ayudantes que no son jefes.
  - `def cargar_instancia(ruta)`: Esta función recibe la ruta al archivo `equipo_docencia.bin`, que corresponde a una instancia serializada de la clase `EquipoDocencia` y se encarga de deserializar el contenido del archivo utilizando `pickle` y luego **retornarlo**.
- Etapa de guardado:
  - `def guardar_instancia(ruta, objeto_lista_ayudantes)`: Recibe la ruta que corresponde a la ubicación de un archivo vacío llamado `equipo_corregido.bin` y una instancia de `EquipoDocencia`. La función debe serializar la instancia en la ruta del archivo nuevo, mediante `pickle`, y luego retornar `True`.
  - `class Ayudante`: Corresponde a todos los ayudantes que tiene cargo de docencia pero no son jefes. **No debes modificarla.**
    - `def __init__(self, cargo, github_user, pokemon_favorito, pizza_favorita):`

Tiene como atributos a `cargo` (`str`), `github_user` (`str`), `pokemon_favorito` (`str`) y `pizza_favorita`(`str`), los cuales recibe como argumentos. **No debes modificar este método.**

- `def __repr__(self):` Se encarga de cambiar la representación de la clase ayudante. **No debes modificar este método.**

- `class AyudanteJefe(Ayudante):` Corresponde al ayudante jefe de docencia. Hereda de `Ayudante`. **Debes modificar uno de sus métodos**.
  - `def __init__(self, cargo, github_user, pokemon_favorito, pizza_favorita, trabajo_restante, experto):` Además de los atributos heredados de `Ayudante`, posee los atributos: `trabajo_restante` (`str`), `experto` (`str`), `carrera` (`str`). Recibe como argumentos todos los atributos de la clase de la que hereda, además de sus propios atributos. **No debes modificar este método.**
  - `def __getstate__(self):` En este método debes personalizar la serialización, modificando el estado de los atributos que se guardarán de la clase. Específicamente, debes asegurarte de que `pizza_favorita` sea `None`, `trabajo_restante` sea igual a `"Nada"` y `experto` sea igual a `"TortugaNinja"`.

## ***Bonus*: Más manejo de *Bytes***

¡Solo te falta un jefe por descubrir! El ~~innombrable~~ misterioso **Benjamín Martinez**. Para lograr descubrirlo, en la carpeta `Bonus` se te entregan **nueve** archivos con el nombre de `secuencia_i`, donde `i` va del 1 al 9, los cuales entregan parte de un mensaje. Tu objetivo es limpiar la información de estos archivos y combinarlos usando tus conocimientos de *bytes*.

Para lograr esto, en la carpeta `Bonus` se te entrega el archivo `bonus.py`, el cual **debes completar**. Aquí, cada archivo `secuencia_i` contiene una secuencia de *bytes*. Debes concatenar dicho contenido respetando el orden ascendente (1, 2, ..., 9) de los archivos. Con ese contenido completo, luego debes filtrar y no considerar aquellos *bytes* cuyo valor numérico es menor estricto a `10` o mayor estricto a `195`.

La cadena de *bytes* restante y que mantiene el orden original, corresponderá al contenido de un módulo de Python, por lo que deberás crear el archivo `clave.py` y escribir los datos ahí. Una vez realizado esto, puedes ejecutar `clave.py` y obtendrás el usuario y clave de **Benjamín**.

## Requerimientos

- (2.00 pts) Parte *Bytes*
  - (0.2 pts) Lee correctamente el contenido de *bytes* en `reparar_usuario`.
  - (0.4 pts) Identifica y separa los segmentos de *bytes*.
  - (1.0 pts) Procesa correctamente los distintos segmentos.
  - (0.2 pts) Concatena correctamente los segmentos procesados.
  - (0.2 pts) Escribe el archivo resultante.
- (2.00 pts) Parte JSON:
  - (0.8 pts) Carga correctamente el contenido del archivo en `cargar_ayudantes`.
  - (0.6 pts) Desencripta correctamente el usuario y los atributos de cada ayudante.

    - (0.6 pts) Retorna la lista con todas las instancias de `AyudanteTareo`.

- (2.00 pts) Parte Pickle:
    - (0.6 pts) Completa la clase `EquipoDocencia`.
    - (0.4 pts) Carga correctamente el contenido del archivo en `cargar_instancia`.
    - (0.4 pts) Serializa correctamente una instancia en `guardar_instancia`.
    - (0.6 pts) Completa la clase `AyudanteJefe`.
- (1.00 pt) *Bonus:* Más manejo de *Bytes*
    - (1.00 pt) Completar el archivo `bonus.py` correctamente.