UNIVERSIDAD RAFAEL LANDÍVAR
LICENCIATURA EN INGENIERÍA EN INFOMÁTICA Y SISMTEAS
PROGRAMACIÓN ORIENTADA A OBJETOS
SEGUNDO SEMESTRE
ERNESTO TAY 1532625
JACKELIN VASQUEZ 1503025
JORGE RIVERA 1511425
PABLO QUIJIVIX 1578125

PSEUDOCODIGO DE PROYECTO DE PROGRAMACIÓN

Quetzaltenango 18 de septiembre del 202

FUNCIÓN menu

- **FUNCIÓN** menu (recibe una lista de opciones y un titulo por defecto "MENÚ")
    - Inicializar seleccion en 0.
    - **BUCLE** mientras sea verdadero
        - Imprimir el titulo con una animación de caracteres.
        - Imprimir una línea separadora.
        - **PARA CADA** opcion con su indice en la lista opciones
            - **SI** indice es igual a seleccion
                - Imprimir la opción con un prefijo de flecha en color magenta claro.
            - **SI NO**
                - Imprimir la opción sin formato especial.
        - Imprimir las instrucciones de navegación (usar ↑/↓ y ENTER).
        - Esperar una pulsación de tecla.
        - **SI** la tecla es flecha arriba (b'H') y la selección no está al inicio, decrementar la selección.
        - **SI NO SI** la tecla es flecha abajo (b'P') y la selección no está al final, incrementar la selección.
        - **SI NO SI** la tecla es ENTER (b'\r')
            - Hacer una pausa de 1.5 segundos.
            - **RETORNAR** la seleccion.
        - Limpiar la pantalla de la consola.
- **FIN DE FUNCIÓN**

CLASE User

- **CLASE** User (clase base para estudiantes y profesores)
    - **MÉTODO** __init__ (Constructor)
        - Inicializar atributos públicos: name, address, dob.
        - Inicializar atributos privados (__): __dpi, __phone, __password.
    - **MÉTODO** @property documento_personal
        - **RETORNAR** el valor de __dpi.
    - **MÉTODO** @property phone_u
        - **RETORNAR** el valor de __phone.
    - **MÉTODO** @phone_u.setter phone_u
        - Recibe un nuevo número de teléfono.
        - **SI** la longitud del número no es 8, imprimir un mensaje de error.
        - **SI NO**, actualizar el valor de __phone.
    - **MÉTODO** @property pass_ward
        - **RETORNAR** el valor de __password.
    - **MÉTODO** display_info (aún no implementado).

- **FIN DE CLASE**

CLASE Student

- **CLASE** Student (hereda de User)
    - **MÉTODO** __init__ (Constructor)
        - Llamar al constructor de la clase padre con super().
        - Inicializar atributos específicos de estudiante: __id_s, gen (año de ingreso), assigned_c (diccionario de cursos asignados) y reports (lista de reportes).
    - **MÉTODO** @property carnet
        - **RETORNAR** el valor de __id_s.
    - **MÉTODO** entregar_tarea
        - Recibe un objeto curso_seleccionado.
        - **SI** no hay actividades en el curso, imprimir un mensaje.
        - Listar las actividades disponibles.
        - Pedir al usuario el número de la actividad a entregar.
        - **INTENTAR**
            - Validar la entrada y obtener la actividad.
            - Marcar la actividad como "Entregado" para el estudiante actual.
            - Imprimir mensaje de éxito.
        - **EXCEPTO** ValueError (si la entrada no es un número), imprimir un mensaje de error.
    - **MÉTODO** display_info
        - **RETORNAR** una cadena de texto formateada con la información del estudiante.
    - **MÉTODO** inscription
        - Recibe un objeto faculty.
        - **SI** no hay cursos disponibles, imprimir un mensaje.
        - Listar los cursos y sus docentes.
        - **BUCLE** mientras sea verdadero
            - Pedir al usuario que ingrese el número del curso.
            - **SI** la opción es válida, obtener el curso y salir del bucle.
            - **SI NO**, imprimir un mensaje de error.
        - **SI** el estudiante ya está inscrito en ese curso, imprimir un mensaje y salir.
        - Agregar el curso a self.assigned_c y al roster_alumnos del curso.
        - Guardar los datos actualizados en los archivos estudiantes.txt y Cursos.txt.
        - Imprimir mensaje de confirmación.
    - **MÉTODO** promedio_general

        - Calcular el promedio general del estudiante sumando las notas obtenidas en todos los cursos y dividiéndolas por el total de puntos posibles.
        - Manejar la excepción ZeroDivisionError si no hay puntos posibles, retornando 0.
        - **RETORNAR** el promedio.
    - **MÉTODO** ver_nota
        - Recibe curso_id y faculty.
        - Buscar el curso.
        - **SI** el curso o las actividades no existen, imprimir un mensaje.
        - Recorrer las actividades, sumar las notas y mostrar el punteo obtenido y total.
    - **MÉTODO** ver_nota_actividad
        - Recibe un objeto curso_seleccionado.
        - Listar cada actividad, mostrando el punteo obtenido por el estudiante y el estado de entrega.
    - **MÉTODO** ver_reportes
        - **SI** la lista de reportes está vacía, imprimir un mensaje.
        - **SI NO**, listar cada reporte con sus detalles (curso, profesor, fecha y descripción).
    - **MÉTODO** deploy_s_menu
        - Recibe un objeto faculty.
        - **BUCLE** mientras sea verdadero
            - Mostrar el menú principal de estudiante y obtener la selección.
            - **EVALUAR** la opción:
                - "1": Ver cursos asignados. Mostrar un submenú para "Entregar Tareas", "Ver nota de curso" o "Ver nota de actividad".
                - "2": Inscribirse a un curso.
                - "3": Ver perfil personal.
                - "4": Ver trayectoria académica.
                - "5": Ver notas de todos los cursos.
                - "6": Ver reportes.
                - "7": Cerrar sesión.
                - Cualquier otro caso: Opción inválida.
- **FIN DE CLASE**

CLASE Teacher

- **CLASE** Teacher (hereda de User)
    - **MÉTODO** __init__ (Constructor)
        - Llamar al constructor de la clase padre.
        - Inicializar el atributo __id_cat y la lista de assigned_courses.
    - **MÉTODO** @property id_cat

        - **RETORNAR** el valor de __id_cat.
    - **MÉTODO** subir_notas
        - Recibe curso y faculty.
        - Mostrar un submenú para actualizar notas de **todas** las actividades o de **una** en particular.
        - **EVALUAR** la opción:
            - "1": Recorrer todas las actividades y todos los estudiantes, pidiendo una nota para cada uno.
            - "2": Pedir el ID de una actividad específica y luego recorrer los estudiantes para pedir sus notas.
    - **MÉTODO** crear_asignacion
        - Recibe un objeto curso.
        - **SI** el profesor no tiene cursos asignados, imprimir un mensaje.
        - **SI NO**, solicitar los datos de la nueva asignación (nombre, valor, fechas, etc.).
        - Validar que el valor de la asignación esté entre 1 y 100.
        - Validar que la suma de todos los valores de las asignaciones no exceda 100 puntos.
        - Validar el formato de fecha y hora (dd-mm-aaaa y hh:mm).
        - **INTENTAR**
            - Crear un objeto Actividad y agregarlo a la lista de asignaciones del curso.
            - Guardar los cambios en el archivo Cursos.txt.
            - Imprimir mensaje de éxito.
        - **EXCEPTO** excepciones personalizadas (ValueError, nameDupeError, fechaFormatError, horaFormatError, courseError), imprimir el mensaje de error correspondiente.
    - **MÉTODO** crear_reporte
        - Recibe un objeto curso.
        - **SI** no hay alumnos en el curso, imprimir un mensaje.
        - Listar a los alumnos y pedir el ID del estudiante para el reporte.
        - Validar que la descripción del reporte tenga al menos 10 caracteres.
        - Crear un diccionario de reporte con los detalles y agregarlo a la lista de reports del estudiante.
    - **MÉTODO** deploy_t_menu
        - Recibe un objeto faculty.
        - **BUCLE** mientras sea verdadero
            - Mostrar el menú del docente y obtener la selección.
            - **EVALUAR** la opción:
                - "1": Ver cursos asignados. Mostrar un submenú para "Crear Asignación", "Subir Notas" o "Generar reporte".
                - "2": Cerrar sesión.
                - Cualquier otro caso: Opción inválida.
- **FIN DE CLASE**

CLASE Curso

- **CLASE** Curso
  - **MÉTODO** __init__ (Constructor)
    - Inicializar id_course, name, teacher_assigned, roster_alumnos (lista), y asignaciones (lista).
  - **MÉTODO** to_dict
    - Convierte el objeto Curso y sus atributos a un diccionario para poder guardarlo en un archivo JSON.
  - **MÉTODO** @staticmethod from_dict
    - Invierte el proceso de to_dict, reconstruyendo un objeto Curso a partir de un diccionario.
  - **MÉTODO** mostrar_datos
    - Recibe faculty.
    - Imprimir los detalles del curso, la lista de alumnos asignados y las asignaciones.
  - **MÉTODO** calcular_nota
    - Recibe carnet.
    - Calcular la nota total obtenida y la nota total posible para un estudiante específico en el curso.
    - **RETORNAR** ambos valores.
- **FIN DE CLASE**

CLASE Actividad

- **CLASE** Actividad
  - **MÉTODO** __init__ (Constructor)
    - Inicializar atributos de la actividad, incluyendo __act_id, name, valor_n, date, h_apertura, h_cierre, type_a, y el diccionario submission para las notas.
    - Llamar a set_status() para determinar si la actividad está abierta o cerrada.
  - **MÉTODO** to_dict y @staticmethod from_dict
    - Métodos para serializar y deserializar el objeto Actividad a y desde un diccionario.
  - **MÉTODO** set_status
    - Compara la fecha y hora actuales con la fecha y hora de cierre para determinar si la actividad está abierta (True) o cerrada (False).
  - **MÉTODO** mostrar_datos
    - Imprimir los detalles de la actividad.
- **FIN DE CLASE**

Funciones Globales y Ejecución Principal

- **FUNCIÓN** guardar
  - Recibe un objeto faculty.
  - **INTENTAR** guardar los datos de estudiantes, profesores y cursos en sus respectivos archivos (.txt).
  - **EXCEPTO** cualquier error, imprimir un mensaje.
- **FUNCIÓN** deploy_admin_menu
  - Recibe un objeto faculty.
  - **BUCLE** mientras el administrador esté autenticado
    - Mostrar el menú de administrador y obtener la selección.
    - **EVALUAR** la opción: "Crear Curso", "Crear Usuario", "Ver cursos", "Ver alumnos", "Ver maestros", "Asignar Maestros", "Guardar" o "Salir".
- **CLASE** Database
  - **MÉTODO** __init__ (Constructor)
    - Inicializar diccionarios para students_db, teachers_db y courses_db.
  - **MÉTODO** cargar_estudiantes, cargar_profesores, cargar_cursos
    - Métodos para leer los datos desde los archivos de texto correspondientes y reconstruir los objetos.
    - Manejar la excepción FileNotFoundError si los archivos no existen.
- **FUNCIONES Auxiliares**
  - **FUNCIÓN** id_creation
    - Genera un ID único basado en un prefijo (C, S, T, A) y números aleatorios.
  - **FUNCIÓN** b_day_check
    - Valida que el formato de la fecha de nacimiento sea correcto (dd/mm/aaaa). Si no, vuelve a solicitar la entrada.
  - **FUNCIÓN** doc_check
    - Valida que el DPI tenga 13 dígitos, sea numérico y no exista ya en la base de datos.
- **EJECUCIÓN PRINCIPAL**
  - Crear una instancia de Database.
  - Cargar todos los datos desde los archivos.
  - **BUCLE** mientras el programa esté en ejecución
    - Solicitar nombre de usuario y contraseña.
    - **SI** las credenciales son de administrador, entrar al menú de administración.
    - **SI NO SI** son de estudiante, entrar al menú de estudiante.
    - **SI NO SI** son de docente, entrar al menú de docente.
    - **SI NO SI** la entrada es 0 y 0, salir del programa.
    - **SI NO**, mostrar un mensaje de error de credenciales.