Rúbrica Evaluación: Sumativa N°1
Programación Orientada a Objeto Seguro.
(ponderación de evaluación 20%)

Área Académica	Tecnologías de Información y Ciberseguridad	**Carrera**	Ingeniería Informática
Sede	Puente Alto	**Código**	TI3V21
Docente	Michael Alexis Arjel Mayerovich	**Fecha**	04/11/2025
Sección	114-2B-F1	**Duración**	7 días

Nombre Estudiante:

Apellido Paterno Apellido Materno Nombres

Rut: ☐☐☐☐☐☐☐☐ - ☐

Puntaje Máximo		**Nota:**	Firma Conforme
Puntaje Obtenido			
Solicita Re-Corrección	**Sí** / **No**	**Motivo:**	

INSTRUCCIONES GENERALES:
1. La nota 4.0 se obtiene logrando un 60% del puntaje total.
2. Preocúpese de la redacción, ortografía y legibilidad de sus respuestas.

Aprendizaje esperado:

1.1.- Informe propuesta de solución con diagrama de clases.

Criterios de evaluación:

1.1.1.- Distingue los conceptos asociados a la programación orientada a objetos, considerando las clases, objetos, atributos y métodos, polimorfismo y herencia.
1.1.2.- Reconoce los símbolos de los diagramas de clases, en base al estándar UML.
1.1.3.- Identifica las relaciones entre las distintas clases, de acuerdo con el tipo de problema analizado.
1.1.4.- Desarrolla diagrama de clases, considerando la simbología y las interacciones de la solución.

I. Presentación de la actividad

La presente actividad corresponde a una evaluación sumativa individual del módulo Programación Orientada a Objetos en Python, específicamente vinculada a los aprendizajes esperados de la Unidad: Fundamentos de la POO y control de versiones con Git.

Los estudiantes deberán desarrollar una serie de ejercicios prácticos en Python que reflejen el uso de clases, constructores, atributos y métodos, además de aplicar un flujo de trabajo básico con Git para gestionar y publicar una rama propia en un repositorio remoto.

La actividad contempla:

- Trabajo individual (cada estudiante debe desarrollar y entregar su propio código).
- Uso de Git: creación de una rama siguiendo el formato indicado y publicación en el repositorio remoto.
- Resolución de ejercicios de programación (7 en total), aplicando POO en Python con instancias y métodos.
- Participación autónoma: se evaluará tanto la correcta implementación de los ejercicios como la claridad y organización del proyecto.

Esta evaluación busca que los estudiantes apliquen conocimientos prácticos de programación y control de versiones en un contexto estructurado, fortaleciendo la capacidad de modelar problemas simples mediante objetos y de manejar un flujo básico de trabajo colaborativo con Git.

II. Actividades

A continuación, se presentan las actividades que cada estudiante deberá desarrollar para abordar la evaluación. El desarrollo debe basarse lo visto en las clases previas del módulo.

GIT.

1. En su repositorio personal deberá crear un proyecto, crea una nueva rama con el siguiente formato:
    - evaluacion1/nombre_apellido.
    - Desarrolla los ejercicios entregados a continuación.
    - Publica tu rama en el repositorio.
    - Fusiona los cambios con tu rama main
    - Verifica que tu rama y código están publicados en el repositorio remoto.

Ejercicios.

Ejercicio 1 — Libro y Biblioteca

Diseña un sistema sencillo para gestionar una biblioteca. Cada libro debe tener identificados su título, autor y la cantidad de copias disponibles. La biblioteca debe permitir registrar nuevos libros, entregar libros en préstamo a los usuarios, recibir devoluciones y mostrar en cualquier momento el estado completo del catálogo (qué libros hay y cuántas copias quedan disponibles).

Requerimientos funcionales

- El sistema debe permitir registrar un nuevo libro, indicando su título, autor y número de copias disponibles.
- El sistema debe permitir consultar el catálogo completo, mostrando todos los libros registrados junto con sus copias disponibles.
- El sistema debe permitir buscar un libro por título para verificar si existe en la biblioteca y cuántas copias quedan.
- El sistema debe permitir registrar el préstamo de un libro, disminuyendo en una unidad la cantidad de copias disponibles.
- Si se intenta prestar un libro sin copias disponibles, el sistema debe informarlo claramente y no realizar el préstamo.
- El sistema debe permitir registrar la devolución de un libro, aumentando en una unidad la cantidad de copias disponibles.
- El sistema debe ofrecer una forma de visualizar el estado actualizado de un libro específico (título, autor, copias disponibles) después de préstamos y devoluciones.

Ejercicio 2 — Alumno y Curso

Diseñe un sistema sencillo para gestionar la inscripción de alumnos en un curso. Cada alumno debe estar identificado por su nombre. El curso debe tener un nombre y mantener el listado de alumnos inscritos, permitiendo agregar nuevos alumnos, retirar alumnos existentes y consultar en cualquier momento quiénes forman parte del curso.

- El sistema debe permitir definir un nuevo curso, indicando su nombre.
- El sistema debe permitir registrar un nuevo alumno, indicando su nombre, y inscribirlo en el curso.
- El sistema debe permitir remover a un alumno del curso, de modo que ya no aparezca en el listado de inscritos.
- El sistema debe permitir listar todos los alumnos inscritos en el curso, mostrando al menos su nombre.
- Si se intenta remover a un alumno que no está inscrito, el sistema debe informarlo claramente.
- El sistema debe ofrecer una forma de consultar el estado actualizado del curso, es decir, su nombre y el listado actual de alumnos inscritos.

Ejercicio 3 — Pedido e Ítem

Diseñe un sistema sencillo para gestionar un pedido de compra. Cada ítem del pedido debe tener un nombre, un precio y una cantidad, y debe ser capaz de calcular su propio subtotal (precio por cantidad). El pedido debe permitir registrar varios ítems y ofrecer una forma de calcular el monto total a pagar sumando los subtotales de todos los ítems incluidos.

- El sistema debe permitir registrar un nuevo ítem indicando su nombre, precio y cantidad.
- El sistema debe permitir calcular el subtotal de un ítem, multiplicando su precio por la cantidad.
- El sistema debe permitir agregar ítems a un pedido, de modo que un mismo pedido pueda contener varios ítems distintos.
- El sistema debe permitir consultar el listado de ítems de un pedido, mostrando al menos el nombre, el precio, la cantidad y el subtotal de cada uno.
- El sistema debe permitir calcular el total del pedido, sumando los subtotales de todos los ítems registrados.
- El sistema debe ofrecer una forma de visualizar el total final a pagar junto con el detalle de los ítems que lo componen.

Ejercicio 4 — Sensor y Mediciones

Diseñe un sistema sencillo para registrar y analizar las mediciones de un sensor. Cada sensor debe estar identificado por un nombre y almacenar los valores de sus mediciones a lo largo del tiempo. El sistema debe permitir registrar nuevos valores para el sensor y consultar en cualquier momento el promedio de las mediciones realizadas, así como el valor máximo y el valor mínimo registrados.

Requerimientos funcionales

- El sistema debe permitir definir un sensor, indicando su nombre.
- El sistema debe permitir registrar nuevas mediciones asociadas a ese sensor.
- El sistema debe permitir consultar el promedio de todas las mediciones registradas para el sensor.
- El sistema debe permitir consultar el valor máximo registrado por el sensor.
- El sistema debe permitir consultar el valor mínimo registrado por el sensor.
- El sistema debe ofrecer una forma de visualizar el nombre del sensor y un resumen de sus mediciones, incluyendo promedio, máximo y mínimo.

Ejercicio 5 — Película y Catálogo

Diseñe un sistema sencillo para gestionar un catálogo de películas. Cada película debe estar identificada por un título, un género y un año de lanzamiento. El catálogo debe permitir registrar nuevas películas, buscar una película específica por su título, filtrar el listado para ver solo las películas de un determinado género y consultar en cualquier momento el listado completo de todas las películas registradas.

Requerimientos funcionales

- El sistema debe permitir registrar una nueva película, indicando su título, género y año de lanzamiento.
- El sistema debe permitir consultar el catálogo completo, mostrando todas las películas registradas con su información básica.
- El sistema debe permitir buscar una película por su título, indicando claramente si se encuentra en el catálogo y mostrando sus datos.
- El sistema debe permitir filtrar las películas por género, mostrando solo aquellas que correspondan al género solicitado.
- Si se realiza una búsqueda o filtro y no se encuentran películas, el sistema debe informarlo claramente.
- El sistema debe ofrecer una forma de visualizar el catálogo actualizado después de agregar nuevas películas o realizar búsquedas y filtros.

Ejercicio 6 — Usuario y Autenticación simple

Diseñe un sistema sencillo de autenticación de usuarios. Cada usuario debe estar identificado por un nombre de usuario y una contraseña. El sistema debe permitir registrar nuevos usuarios y, posteriormente, validar sus credenciales cuando intenten iniciar sesión, indicando claramente si el acceso es aceptado o rechazado.

Requerimientos funcionales

- El sistema debe permitir registrar un nuevo usuario, indicando nombre de usuario y contraseña.
- El sistema debe impedir registrar un nombre de usuario que ya exista, informando claramente la situación.
- El sistema debe permitir iniciar sesión (login) solicitando nombre de usuario y contraseña.
- Al intentar iniciar sesión, el sistema debe validar las credenciales y:
    - Autorizar el acceso si el usuario existe y la contraseña coincide.
    - Rechazar el acceso si el usuario no existe o la contraseña es incorrecta, informando el motivo de forma clara.
- El sistema debe permitir consultar si un usuario está registrado, usando su nombre de usuario.
- El sistema debe ofrecer una forma de mostrar un mensaje de resultado después de cada intento de registro o login, indicando si la operación fue exitosa o no.

Ejercicio 7 — Agenda y Contacto

Diseñe un sistema sencillo para gestionar una agenda de contactos. Cada contacto debe estar identificado por un nombre, un número de teléfono y un correo electrónico. La agenda debe permitir registrar nuevos contactos, buscar un contacto específico, eliminar contactos existentes y consultar en cualquier momento el listado completo de las personas almacenadas.

Requerimientos funcionales

- El sistema debe permitir agregar un nuevo contacto, indicando nombre, teléfono y correo electrónico.
- El sistema debe permitir consultar el listado completo de contactos, mostrando los datos principales de cada uno.
- El sistema debe permitir buscar un contacto por su nombre, mostrando su información si existe en la agenda.
- El sistema debe permitir eliminar un contacto de la agenda, usando su nombre u otro dato identificador.
- Si se intenta buscar o eliminar un contacto que no existe, el sistema debe informarlo claramente.
- El sistema debe ofrecer una forma de visualizar el estado actualizado de la agenda después de agregar, eliminar o buscar contactos

III. Requerimiento de presentación y entrega

Para la entrega de esta evaluación, cada alumno deberá subir al espacio habilitado en el Ambiente de Aprendizaje un bloc de notas con EL NOMBRE DE SU RAMA.

Requisitos del código:

- Cada alumno deberá subir el código a su repositorio personal, creando el proyecto según lo visto en clases.
- Las clases asociadas a un ejercicio deberán estar en una carpeta (ejercicio1/clases/nombreclase.py).
- La instancia de cada ejercicio deberá llamarse desde la clase main.py.

Medio de entrega:

- Subida al Aula Virtual correspondiente, dentro del plazo establecido por el/la docente.

Fecha de entrega:

- 11/11/2025 a las 20:29

Criterio	Dimensión /Indicadores	Niveles de desempeño.					Observaciones
		Destacado (7 puntos)	Habilitado (5 puntos)	En desarrollo (3 puntos)	No logrado (1 punto)	Puntaje obtenido	
Creación rama GIT	Crea y publica correctamente una rama en el repositorio remoto	Crea la rama con el formato solicitado y la publica en remoto sin errores.	Crea y publica la rama, pero no respeta totalmente el formato o requiere mínima ayuda.	Crea la rama pero no logra publicarla en remoto sin asistencia	No crea ni publica la rama		
Ejercicio 1 — Libro y Biblioteca	El estudiante desarrolla correctamente la clase solicitada en el ejercicio, utilizando constructor __init__, atributos y métodos de instancia, asegurando que el código sea funcional y muestre ejemplos de uso.	Clase completa y funcional, con constructor, atributos y métodos correctos. Incluye validaciones básicas se usa en el método main	Clase presente pero incompleta o con fallas importantes en constructor o métodos. Lógica parcial o errores frecuentes al ejecutar.	Clase ausente, incorrecta o con errores graves que impiden su ejecución. No cumple lo solicitado en el enunciado.	Ejercicio no desarrollado.		

Ejercicio 2 — Alumno y Curso	El estudiante desarrolla correctamente la clase solicitada en el ejercicio, utilizando constructor __init__, atributos y métodos de instancia, asegurando que el código sea funcional y muestre ejemplos de uso.	Clase completa y funcional, con constructor, atributos y métodos correctos. Incluye validaciones básicas se usa en el método main	Clase presente pero incompleta o con fallas importantes en constructor o métodos. Lógica parcial o errores frecuentes al ejecutar.	Clase ausente, incorrecta o con errores graves que impiden su ejecución. No cumple lo solicitado en el enunciado.			
Ejercicio 3 — Pedido e Ítem	El estudiante desarrolla correctamente la clase solicitada en el ejercicio, utilizando constructor __init__, atributos y métodos de instancia, asegurando que el código sea funcional y muestre ejemplos de uso.	Clase completa y funcional, con constructor, atributos y métodos correctos. Incluye validaciones básicas se usa en el método main	Clase presente pero incompleta o con fallas importantes en constructor o métodos. Lógica parcial o errores frecuentes al ejecutar.	Clase ausente, incorrecta o con errores graves que impiden su ejecución. No cumple lo solicitado en el enunciado.			

Ejercicio 4 — Sensor y Mediciones	El estudiante desarrolla correctamente la clase solicitada en el ejercicio, utilizando constructor __init__, atributos y métodos de instancia, asegurando que el código sea funcional y muestre ejemplos de uso.	Clase completa y funcional, con constructor, atributos y métodos correctos. Incluye validaciones básicas se usa en el método main	Clase presente pero incompleta o con fallas importantes en constructor o métodos. Lógica parcial o errores frecuentes al ejecutar.	Clase ausente, incorrecta o con errores graves que impiden su ejecución. No cumple lo solicitado en el enunciado.			

Ejercicio 5 — Película y Catálogo	El estudiante desarrolla correctamente la clase solicitada en el ejercicio, utilizando constructor __init__, atributos y métodos de instancia, asegurando que el código sea funcional y muestre ejemplos de uso.	Clase completa y funcional, con constructor, atributos y métodos correctos. Incluye validaciones básicas se usa en el método main	Clase presente pero incompleta o con fallas importantes en constructor o métodos. Lógica parcial o errores frecuentes al ejecutar.	Clase ausente, incorrecta o con errores graves que impiden su ejecución. No cumple lo solicitado en el enunciado.			
Ejercicio 6 — Usuario y Autenticación simple	El estudiante desarrolla correctame	Clase completa y funcional, con	Clase presente pero incomplet	Clase ausente, incorrecta o con			

	nte la clase solicitada en el ejercicio, utilizando constructor __init__, atributos y métodos de instancia, asegurando que el código sea funcional y muestre ejemplos de uso.	constructor, atributos y métodos correctos. Incluye validaciones básicas se usa en el método main	a o con fallas importantes en constructor o métodos. Lógica parcial o errores frecuentes al ejecutar.	errores graves que impiden su ejecución. No cumple lo solicitado en el enunciado.			
Ejercicio 7 — Agenda y Contacto	El estudiante desarrolla correctamente la clase solicitada en el ejercicio, utilizando constructor __init__, atributos y métodos de instancia, asegurando que el código sea funcional y muestre ejemplos de uso.	Clase completa y funcional, con constructor, atributos y métodos correctos. Incluye validaciones básicas se usa en el método main	Clase presente pero incompleta o con fallas importantes en constructor o métodos. Lógica parcial o errores frecuentes al ejecutar.	Clase ausente, incorrecta o con errores graves que impiden su ejecución. No cumple lo solicitado en el enunciado.			

	Puntaje total		

IV

Tabla de puntaje y nota:

Pueden usar este formato o capturar la escala del siguiente link:

https://escaladenotas.cl/

Puntaje	Nota	Puntaje	Nota	Puntaje	Nota	Puntaje	Nota	Puntaje	Nota
0.0	1.0	10.0	2.0	20.0	3.0	30.0	4.1	40.0	5.6
1.0	1.1	11.0	2.1	21.0	3.1	31.0	4.2	41.0	5.8
2.0	1.2	12.0	2.2	22.0	3.2	32.0	4.4	42.0	5.9
3.0	1.3	13.0	2.3	23.0	3.3	33.0	4.6	43.0	6.1
4.0	1.4	14.0	2.4	24.0	3.4	34.0	4.7	44.0	6.2
5.0	1.5	15.0	2.5	25.0	3.6	35.0	4.9	45.0	6.4
6.0	1.6	16.0	2.6	26.0	3.7	36.0	5.0	46.0	6.5
7.0	1.7	17.0	2.7	27.0	3.8	37.0	5.2	47.0	6.7
8.0	1.8	18.0	2.8	28.0	3.9	38.0	5.3	48.0	6.8
9.0	1.9	19.0	2.9	29.0	4.0	39.0	5.5	49.0	7.0