Programación Avanzada (POO) Depto. Ing. Computación – FI - UMAG

Guía de Ejercicios POO

Mayo 2025

Prof. Carlos Arias Méndez

Problema 1: Sistema de Biblioteca y Gestión de Préstamos de Libros

Una biblioteca desea implementar un sistema básico y orientado a objetos, para gestionar su colección de libros y permitir préstamos y devoluciones de los mismos. Para ello, se deben definir las clases necesarias para representar los libros individuales y la biblioteca que los contiene.

1) Para los libros, se debe considerar su título (cadena), autor (cadena), y una marca de disponible (booleano, por defecto True). De tal forma que al hacer la acción de prestar dicho libro, si está disponible, lo marca como no disponible y devuelve True. En caso contrario, devuelve False. Y, cuando se haga la acción de devolver el libro se marca como disponible.

2) Para la Biblioteca, que tenga una lista de objetos libros. Y, que permita:
   a) agregar un libro a la colección
   b) buscar un libro por título y, si está disponible, lo presta.
   c) buscar un libro por título.

Objetivo del problema:

Implementar correctamente las clases siguiendo los principios de la programación orientada a objetos, utilizando adecuadamente los métodos y atributos, para modelar un sistema funcional de préstamos y devoluciones.

Sección de Autoevaluación

Responda brevemente las siguientes preguntas para reflexionar sobre su comprensión del problema:

- ¿Qué representa cada clase en este problema y cuál es su responsabilidad?
- ¿Cómo se relacionan los objetos de dichas clases?
- ¿Qué sucede si se intenta prestar un libro que no está disponible?
- ¿Qué ocurriría si se devolviera un libro que no está registrado en la biblioteca?
- ¿Qué ventaja ofrece usar POO para modelar este tipo de problemas?
- ¿El código está bien organizado y documentado?

Programación Avanzada (POO) Depto. Ing. Computación – FI - UMAG

Problema 2: Sistema de Reserva de Hotel

Una cadena hotelera desea automatizar el proceso de reserva de habitaciones para sus clientes. Para ello, se requiere desarrollar un sistema básico de reservas utilizando los conceptos de clases, atributos, y métodos, respetando el paradigma de programación orientada a objetos.

Se deben modelar los siguientes elementos:
1. Cada habitación del hotel está identificada por un número y puede estar ocupada o disponible.
2. Cada cliente tiene un nombre y, eventualmente, puede tener una reserva de habitación asociada.

El sistema debe permitir:
- Reservar una habitación solo si se encuentra disponible.
- Asociar correctamente una habitación reservada al cliente.
- Liberar una habitación (marcarla como disponible nuevamente).

Requerimientos específicos:
1. En relación a la habitación, ésta contempla:
- Un número que la identifica.
- Una marca que indique si la habitación está actualmente ocupada (inicialmente debe estar libre).
- Al reservarla, que:
  - Se marque la habitación como ocupada si estaba libre.
  - Devuelva True si la reserva fue exitosa y False si ya estaba ocupada.
- Al liberar la habitación, se marque como disponible nuevamente.

2. En relación al Cliente:
- Que se identifique por su nombre.
- Vincularle la habitación que se le ha asignado, o None si aún no tiene ninguna.
- Que al hacer la reserva de la habitación, que:
  - Intente reservar la habitación indicada por la persona que administra la aplicación.
  - Si la reserva es exitosa, asignar la habitación al cliente y retorne un mensaje del tipo:
    "Habitación 101 reservada para Juan"
  - Si no es posible realizar la reserva, retorne:
    "No se pudo reservar".

3. Simula el siguiente escenario:
- Crear una habitación con número 101.
- Crear un cliente llamado "Juan".

Programación Avanzada (POO) Depto. Ing. Computación – FI - UMAG

Problema 3: Sistema de Gestión de Inscripciones en una Universidad

Una universidad desea implementar un sistema básico que le permita registrar estudiantes e inscribirlos en distintos cursos. Para ello, se requiere modelar el comportamiento de los estudiantes y los cursos utilizando programación orientada a objetos.

Requisitos:

1. En relación a los cursos:

- - Deben tener el nombre del curso(str), y los estudiantes inscritos en dicho curso (lista de objetos de estudiantes).
- - Inscribir estudiantes, agregando el estudiante a la lista de inscritos.

2. En relación a los estudiantes:

- - Se deben identificar por su nombre, y los cursos en los que está inscrito (lista de objetos curso).
- - Permitir la acción de inscribirse en un curso, agregando el curso a la lista del estudiante.

Actividades requeridas:

a) Implementar las clases correspondientes respetando las relaciones bidireccionales entre estudiantes y cursos.

b) Crear al menos tres cursos y cuatro estudiantes, e inscribir a los estudiantes en distintos cursos.

c) Imprimir por pantalla:

- - Los cursos a los que está inscrito cada estudiante.
- - Los nombres de los estudiantes inscritos en cada curso.

Consideraciones adicionales:

- - Asegurarse de mantener la coherencia entre las clases.
- - Evitar inscripciones duplicadas (opcional).
- - Utilizar comprensiones de listas o bucles for para imprimir la información.

Autoevaluación sugerida:

1. ¿Qué relación existe entre las clases creadas? ¿Cómo se refleja en el código?

2. ¿Cómo se garantiza la consistencia entre la lista de cursos de un estudiante y la lista de inscritos de un curso?

3. ¿Qué modificaciones haría al sistema si un estudiante quisiera darse de baja de un curso?

4. ¿Cómo evitaría inscripciones duplicadas de un estudiante en un mismo curso?

5. ¿Podría identificar si este sistema permite fácilmente agregar más atributos, como un ID de curso o una nota final? Justifique su respuesta.

6. ¿El código está bien organizado y documentado?

Programación Avanzada (POO) Depto. Ing. Computación – FI - UMAG

Problema 4: Simulación de un Cajero Automático

Una entidad bancaria desea simular el funcionamiento básico de un cajero automático, centrado en la operación de retiros de dinero desde cuentas bancarias. Para ello, se requiere el diseño de un sistema basado en Programación Orientada a Objetos que incluya:

- Clientes que poseen cuentas bancarias, identificadas por un número de cuenta y un saldo disponible.
- Un cajero automático, que realiza la operación de retiro de dinero desde una cuenta, siempre y cuando existan fondos suficientes.

Especificaciones del sistema

1. Sobre la cuenta bancaria:
   - Que tenga un número identificador de la cuenta (tipo cadena) y un saldo actual disponible en la cuenta (tipo numérico).
   - Que al retirar una cantidad de dinero verifique si el saldo disponible es suficiente. Si el retiro es exitoso, descontar la cantidad del saldo y devuelve True. Si no hay fondos suficientes, devuelve False.

2. En relación al cajero:
   - Que permita hacer retiros basado en la cuenta y monto. Utiliza la cuenta bancaria para intentar realizar el retiro. Devolver un mensaje indicando si el retiro fue exitoso y cuál es el nuevo saldo, o si hubo fondos insuficientes.

Objetivo del estudiante

Implementar correctamente las clases, garantizando la interacción entre ellas y simulando una operación de retiro desde una cuenta bancaria usando un cajero automático.

Requisitos adicionales para la evaluación

1. Agregar al menos dos cuentas bancarias diferentes con saldos distintos.
2. Simular al menos tres operaciones de retiro, incluyendo:
   - Un retiro exitoso.
   - Un intento de retiro con monto mayor al saldo disponible.
   - Un retiro que reduzca el saldo a cero.
3. Mostrar por pantalla los mensajes generados por cada operación de retiro.
4. Agregar una función mostrar_saldo(cuenta) que imprima el número de cuenta y su saldo actual.
5. (Opcional) Extender la funcionalidad de la cuenta bancaria para registrar un historial de retiros realizados.

Actividades de autoevaluación

1. ¿Qué sucede si intenta retirar más dinero del que hay en la cuenta?
2. ¿Cómo se produce la colaboración entre el cajero y la cuenta bancaria?

3. ¿Qué responsabilidad tiene cada clase en el diseño orientado a objetos?
4. ¿Sería posible adaptar este sistema para incluir depósitos? ¿Qué cambios haría?

5. ¿El código está bien organizado y documentado?

Programación Avanzada (POO) Depto. Ing. Computación – FI - UMAG

Problema 5: Sistema de Gestión de Evaluación Académica de Estudiantes

Un profesor desea implementar un sistema simple de evaluación académica para registrar y calcular el desempeño de sus estudiantes en distintos exámenes.

Se requiere modelar los siguientes elementos utilizando programación orientada a objetos:

- - Cada examen debe abarcar:
  - El tema evaluado (por ejemplo, 'POO', 'Bases de Datos', etc.).
  - La nota obtenida, representada como un número entre 1.0 y 7.0.
- - Para cada estudiante se debe considerar:
  - Un nombre que lo identifique.
  - Una lista de exámenes rendidos.
  - La posibilidad de:
    - Agregar un examen a su historial académico.
    - Calcular su promedio general de notas a partir de los exámenes rendidos.

Requerimientos

1. Crea una clase `Examen` que reciba el tema y la nota.

2. Por cada estudiante considerar que:
   - Reciba el nombre al momento de su creación.
   - Mantenga internamente una lista de exámenes rendidos.
   - Permita agregar exámenes.
   - Calcule el promedio general de notas basado en todos los exámenes rendidos.

3. Escribir un ejemplo para probar el programa, donde:
   - Se cree un estudiante con nombre.
   - Se agreguen al menos dos exámenes con distintas notas.
   - Se imprima el promedio final del estudiante.

Consideraciones opcionales

- Manejar el caso en que el estudiante no tenga exámenes al calcular el promedio (evitar división por cero).

- Validar que la nota esté dentro del rango permitido (por ejemplo, entre 1.0 y 7.0).

- Agregar un método `mostrar_examenes()` que liste todos los exámenes rendidos con su tema y nota.

Autoevaluación sugerida

1. ¿Pudo implementar correctamente las clases con los métodos requeridos?

2. ¿La implementación considera posibles errores como listas vacías o notas inválidas?

3. ¿Probó el programa con varios estudiantes y exámenes distintos?

4. ¿Documentó adecuadamente el código con comentarios claros?

Programación Avanzada (POO) Depto. Ing. Computación – FI - UMAG

Problema 6: Agenda de Contactos

Un desarrollador está implementando un sistema de agenda digital para gestionar los contactos personales de un usuario. Esta agenda permite almacenar contactos con su nombre y número telefónico, y ofrece la funcionalidad de búsqueda por nombre.

Se pide implementar una solución utilizando programación orientada a objetos (POO), considerando los siguientes requerimientos:

Requisitos del Modelo

1. Que el contacto considere:
   - nombre: cadena de texto
   - teléfono: cadena de texto
2. Que la agenda considere una lista de contactos y permita:
   - Agregar un contacto
   - Buscar un contacto por su nombre, de tal forma que retorne una lista con los teléfonos de los contactos cuyo nombre coincide exactamente con el buscado.

Parte Obligatoria del Ejercicio

1. Implementar las clases y los métodos correspondientes.
2. La búsqueda debe ser exacta (sensible a mayúsculas y minúsculas).
3. Demostrar el funcionamiento con al menos tres contactos, de los cuales dos deben tener el mismo nombre.

Autoevaluación Sugerida

- ¿Implementó correctamente las clases y métodos requeridos?
- ¿Probó el programa con contactos repetidos?
- ¿La búsqueda funciona exactamente como se espera?
- ¿El código está bien organizado y documentado?

Programación Avanzada (POO) Depto. Ing. Computación – FI - UMAG

Problema 7: Sistema de Pedidos de una Tienda

Una tienda desea implementar un sistema básico para registrar los pedidos que realizan los clientes. Cada pedido puede contener uno o más productos. La tienda necesita llevar un registro de los productos agregados a cada pedido y calcular el total a pagar.
Para ello, se desea modelar el sistema utilizando programación orientada a objetos. El sistema debe incluir al menos lo siguiente:

Que el producto cuando sea creado, se incluyan los valores para el nombre del producto (cadena de texto) y su precio (número real positivo).

Que el pedido considere una lista que almacene objetos productos que forman parte del pedido y que permita agrega un producto a la lista de productos del pedido. Además, que permita conocer la suma total de los precios de todos los productos del pedido.

Requerimientos

1. Implemente las clases respetando las especificaciones anteriores.
2. Cree un pedido nuevo y agregue al menos tres productos distintos con precios variados.
3. Muestre en pantalla el total del pedido utilizando el método correspondiente.
4. Asegúrese de aplicar correctamente los principios de encapsulamiento y uso de objetos en la solución.
5. Opcional: permita agregar productos utilizando datos ingresados por el usuario (input()), y muestra un resumen del pedido indicando el nombre y precio de cada producto, seguido del total general.

Autoevaluación

- ¿He utilizado correctamente las clases y los objetos en mi solución?
- ¿He separado correctamente los datos (atributos) del comportamiento (métodos)?
- ¿He probado mi código con diferentes productos y verificado que el total es correcto?
- ¿Mi código es legible y está bien comentado?

Programación Avanzada (POO) Depto. Ing. Computación – FI - UMAG

Problema 8: Sistema de Mensajería entre Usuarios

En esta actividad, se debe diseñar un sistema orientado a objetos que simule el envío y la recepción de mensajes entre distintos usuarios. Cada usuario tendrá un nombre y un buzón de mensajes, donde se almacenarán los mensajes recibidos con el nombre del remitente y el contenido del mensaje.

Requisitos del Programa

1. Crear usuarios que tengan el nombre del usuario (string) y un buzón con la lista de mensajes recibidos. Además, que permita enviar un mensaje a otro usuario y leer todos los mensajes recibidos.

Ejemplo de Uso

A continuación se muestra un ejemplo del comportamiento esperado del programa:

```
u1 = Usuario("Alicia")
u2 = Usuario("Juan")
u3 = Usuario("Clara")

u1.enviar_mensaje(u2, "Hola Carlos, ¿cómo estás?")
u2.enviar_mensaje(u1, "Hola Alicia, todo bien. ¿Y tú?")
u3.enviar_mensaje(u1, "Alicia, ¿tienes un momento?")
u1.enviar_mensaje(u3, "Claro Clara, dime.")

print("Mensajes de Alicia:", u1.leer_mensajes())
print("Mensajes de Juan:", u2.leer_mensajes())
print("Mensajes de Clara:", u3.leer_mensajes())
```

Autoevaluación

Responda las siguientes preguntas para reflexionar sobre su solución:

1. ¿Cómo se relacionan los objetos entre sí? ¿Qué tipo de asociación existe?
2. ¿Qué ventajas tiene separar la funcionalidad en métodos como enviar_mensaje y leer_mensajes?
3. ¿Sería posible extender este sistema para incluir respuestas a mensajes o mensajes con fecha? ¿Cómo lo harías?

Programación Avanzada (POO) Depto. Ing. Computación – FI - UMAG

Problema 9: Sistema de Inventario de Productos

Se desea desarrollar un sistema básico de inventario para una tienda o almacén. El sistema debe permitir registrar productos con su respectivo nombre y cantidad, gestionar el stock de cada producto cuando se agregan nuevas unidades, y consultar cuántas unidades de un producto hay disponibles actualmente.

Para ello, se debe considerar lo siguiente:
1. Que el producto se identifique por su nombre (tipo str) y la cantidad disponible del producto (tipo int), que se determinan cuando el producto es creado para ser ingresado al inventario.

2. Que el inventario contemple un diccionario cuyas claves son los nombres de los productos y cuyos valores son los objetos correspondientes a los productos creados. Permitiendo agregar un producto, recibiendo un objeto correspondiente a un producto; si el producto ya existe en el inventario (mismo nombre), debe sumar la cantidad; de lo contrario, debe agregar el nuevo producto al diccionario. Y, que permita consultar el stock, recibiendo un nombre de producto y devolviendo la cantidad disponible; si el producto no existe, debe devolver 0.

Requisitos para la evaluación:

1. Implementar correctamente las clases según lo indicado.
2. Usar encapsulamiento apropiado si se considera necesario.
3. Comprobar el funcionamiento del método agregar_producto para casos repetidos y nuevos.
4. El método consultar_stock debe manejar correctamente productos inexistentes.

Autoevaluación sugerida:

- ¿Fui capaz de reutilizar una misma clase para representar varios productos?
- ¿Logré evitar duplicados en el inventario usando el nombre como clave?
- ¿Qué ocurriría si se agrega un producto con cantidad cero?
- ¿Qué mejoras propondría al diseño del sistema?

Programación Avanzada (POO) Depto. Ing. Computación – FI - UMAG

Problema 10: Sistema de Empleados y Nómina

Una empresa desea automatizar el cálculo de la nómina mensual de sus empleados. Cada empleado recibe un salario fijo mensual.

Requerimientos:

1. Para el empleado considerar el nombre del empleado, el salario base mensual del empleado. Además, que permita calcular el pago, retornando el salario base mensual del empleado.

2. Para la gestión de la nómina considere las siguientes responsabilidades:

- Lista interna para almacenar empleados.
- Agregar un empleado a la nómina.
- Calcular y retorna la suma total de todos los salarios del mes.

Finalizada la solución del problema anterior, extenderlo para considerar bonos:

Implemente ahora la alternativa de considerar un bono como posible ingreso adicional, que se sume al salario. Para ello, utilice el concepto de herencia.

Evaluación sugerida:

- ¿Pude implementar correctamente las clases con sus atributos y métodos?

- ¿Entendí la importancia del uso de herencia para extender funcionalidad?

- ¿Fui capaz de calcular correctamente la nómina total de empleados con y sin bono?