Facultad de Ingeniería – TDSC - UNSTA

Materia: Desarrollo Back End – 1° Trabajo Práctico.

Profesor: Ing. Tulio Ruesjas Martín.

Fecha de Entrega : 05 de setiembre de 2023 Nombre

y Apellido del Alumno:

Temario:

1. ¿Defina que es una clase y atributo en OOP(Object Oriented Programming)?
2. Se desea hacer el modelado de clases de una compañía aérea en una base de datos relacional. La compañía aérea tiene tres recursos principales: aviones, pilotos y miembros de tripulación. De cada piloto se desea conocer su código, nombre y horas de vuelo. De los miembros de tripulación sólo mantendremos su código y nombre. Todos ellos (pilotos y miembros) tienen una base a la que regresan después de los vuelos de una jornada. Un vuelo que va desde un origen a un destino y a una hora determinada, tiene un número de vuelo (por ejemplo, el vuelo de Tucumán a Buenos Aires de las 13:50 es el vuelo IB-8830). De cada vuelo que se va a realizar durante los próximos tres meses, así como de los vuelos que ya se han realizado, se desea saber el avión en que se va a hacer o en el que se ha hecho, el piloto y cada uno de los miembros de la tripulación. Cada avión tiene un código, es de un tipo (por ejemplo, BOEING-747) y tiene una base donde es sometido a las revisiones periódicas de mantenimiento.
   Solo se deben identificar las clases con sus atributos.

3. ¿Cuáles son las características de una clase?
4. ¿Qué es una interface? ¿Cual es la diferencia entre una clase y una interface?
5. ¿Cuáles son las relaciones que existen entre clases?¿Cual es el concepto de cada una de esas relaciones?
6. ¿Cuáles son las diferencias entre Agregación y Composición?
7. ¿Qué es un diagrama de clases?
8. Elabore un diagrama de clases para una agencia de alquiler de autos.

Formato del Trabajo:

1. El trabajo deberá ser entregado en formato PDF.
2. Utiliza una estructura clara y coherente para cada sección del trabajo.
3. Cita adecuadamente las fuentes utilizadas para respaldar tus respuestas.

Facultad de Ingeniería – TDSC - UNSTA

4. El trabajo debe ser enviado a: tulio.ruesjas@unsta.edu.ar hasta las 14 horas del día 05 de setiembre del 2024. TP enviado después de esa hora será rechazado.

Firma Alumno Cantidad de Hojas(Incluida Enunciado)

Nota	Firma Profesor

Desarrollo

1) Una clase es un elemento de la programación orientada a objetos que actúa como una plantilla y va a definir las características y comportamientos de una entidad. La clase va a ser como un molde a partir del cual vamos a poder definir entidades. Una clase va a definir las características y los comportamientos de una entidad.
   Un atributo es una característica o propiedad que se asigna a un objeto, variable o elemento de código. Estos atributos se utilizan para describir y definir las características de un elemento en particular

2) Avion: código*, tipo, estadoMantenimiento, base
   Piloto: código*, nombre
   Vuelo: código, origen, destino, horaSalida, horaLlegada, estadoVuelo, fechaVuelo, codAvion, codPiloto1, codPiloto2, List<Miembro tripulación>
   MiembroTripulacion: código*, nombre

   3)

1. Atributos (o propiedades):

- Son las variables que almacenan el estado o la información de los objetos creados a partir de una clase.
- Representan las características de un objeto.

2. Métodos:

- Son las funciones definidas dentro de la clase que describen el comportamiento de los objetos.
- Pueden modificar los atributos o realizar otras operaciones relacionadas con la clase.

3. Encapsulamiento:

- Es el concepto de **ocultar** los detalles internos de los objetos y **restringir el acceso** a ciertos atributos o métodos desde fuera de la clase.
- Se logra usando modificadores de acceso como public, private y protected, que definen qué partes del código pueden acceder a los atributos o métodos.

4. Herencia:

- Permite crear una **nueva clase (subclase)** a partir de una existente (superclase), lo que permite **reutilizar el código** y extender las funcionalidades.
- Una subclase hereda atributos y métodos de la superclase, y puede agregar o modificar comportamiento.

5. Polimorfismo:

- Permite que una **misma operación** se comporte de diferentes maneras en diferentes clases.
- Se puede lograr mediante **sobrecarga** (varias implementaciones del mismo método con diferentes parámetros) o **sobreescritura** (modificación de un método heredado en la subclase).

6. Abstracción:

- Es el proceso de **ocultar detalles complejos** y exponer solo la interfaz necesaria para interactuar con el objeto.

7. Constructores:

- Son **métodos especiales** que se utilizan para inicializar los objetos de una clase.
- Generalmente, permiten asignar valores iniciales a los atributos cuando se crea un objeto.

4) Una **interfaz** en programación orientada a objetos es una estructura que define un conjunto de **métodos** que una clase debe implementar, sin proporcionar su implementación concreta. Es como un contrato que establece **qué** métodos deben existir en una clase, pero no **cómo** deben comportarse esos métodos.

La diferencia entre una **clase** y una **interfaz** en programación orientada a objetos radica principalmente en su propósito y en cómo se utilizan para definir y estructurar el comportamiento de los objetos. Aquí te explico sus diferencias más relevantes:

1. Propósito:

- **Clase**: Es una plantilla o molde para crear objetos. Define tanto **atributos** (estado) como **métodos** (comportamiento) que pueden tener los objetos.
- **Interfaz**: Define solo el **comportamiento** (métodos) que las clases que la implementen deben cumplir, sin proporcionar implementación alguna. No tiene atributos de instancia, solo métodos abstractos.

2. Atributos:

- Clase: Puede tener atributos (variables) que representan el estado de los objetos, además de constantes.
- Interfaz: No puede tener atributos de instancia. En algunos lenguajes, puede tener constantes (valores que no cambian), pero no atributos variables.

3. Métodos:

- Clase: Los métodos en una clase pueden tener implementación (es decir, incluyen el código que define su comportamiento). También puede contener constructores, que se utilizan para crear objetos de la clase.
- Interfaz: Los métodos en una interfaz son abstractos (sin implementación). Solo definen la firma (nombre, tipo de retorno, parámetros) de los métodos, dejando la implementación a las clases que implementan la interfaz.

4. Herencia:

- Clase: Puede heredar de otra clase (en muchos lenguajes, solo puede heredar de una clase a la vez). Hereda tanto atributos como métodos.
- Interfaz: Una clase puede implementar múltiples interfaces a la vez. En algunos lenguajes (como Java), una interfaz puede heredar de otras interfaces, pero no puede heredar de una clase.

5. Polimorfismo:

- Clase: Una clase puede ser utilizada en polimorfismo, pero solo a través de su jerarquía de herencia.
- Interfaz: Es una herramienta fundamental para el polimorfismo, ya que diferentes clases pueden implementar la misma interfaz y ser tratadas de manera uniforme, independientemente de su implementación específica.

6. Modificadores de acceso:

- Clase: Los métodos y atributos de una clase pueden tener modificadores de acceso como public, private, protected, lo que define el nivel de visibilidad de esos elementos.
- Interfaz: Los métodos de una interfaz son, por defecto, públicos (aunque esto puede variar según el lenguaje), ya que se supone que deben ser implementados por otras clases. No hay atributos privados o protegidos en una interfaz.

7. Constructores:

- Clase: Puede tener uno o más constructores, que se utilizan para inicializar objetos cuando se crean.
- Interfaz: No tiene constructores, ya que no se pueden crear instancias directamente de una interfaz.

8. Implementación múltiple:

- **Clase**: En muchos lenguajes (como Java), una clase puede heredar de solo **una clase** (herencia simple).
- **Interfaz**: Una clase puede **implementar múltiples interfaces** al mismo tiempo, lo que le permite cumplir con varios contratos o comportamientos.

9. Uso típico:

- **Clase**: Se utiliza para crear objetos con una estructura y comportamiento definido, y puede ser parte de una jerarquía de herencia.
- **Interfaz**: Se usa para definir un conjunto de métodos que varias clases pueden implementar de maneras diferentes, proporcionando flexibilidad en el diseño del sistema.

5) Las clases, al igual que los objetos, no existen de modo aislado. La Orientación a Objetos (POO) intenta modelar aplicaciones del mundo real tan fielmente como sea posible y por lo tanto debe reflejar estas relaciones entre clases y objetos.
Existen tres clases básicas de relaciones entre los objetos:

- Agregación / Composición
- Asociación
- Generalización / Especialización. HERENCIA
    - Herencia Simple.
    - Herencia Múltiple.

Composición:

- **Descripción**: Es una relación en la que una clase está compuesta de una o más instancias de otras clases, implicando una relación de **todo/parte**. Si la clase "contenedora" es destruida, las clases contenidas también lo serán. Existe una **dependencia fuerte** entre las clases.

Agregación:

- **Descripción**: Es una relación similar a la composición, pero más **débil**. En la agregación, las clases "contenidas" no dependen totalmente de la "contenedora". Si la clase contenedora es destruida, las clases que están dentro de ella pueden seguir existiendo independientemente.
-

Asociación:

- **Descripción**: Es una relación más general entre dos clases, donde ambas interactúan entre sí, pero ninguna de ellas depende estrictamente de la otra. La asociación se utiliza para reflejar que un objeto necesita interactuar con otro para cumplir con una tarea.

Herencia (Generalización/Especialización):

- Descripción: Es la relación en la que una clase (subclase) hereda propiedades y métodos de otra clase (superclase). La subclase es una especialización de la superclase, lo que significa que hereda su comportamiento y atributos, y puede agregar o modificar funcionalidades.

6)

Agregación	Composición
Un subconjunto de la asociación	Un subconjunto de la agregación.
Un tipo de asociación débil.	Un tipo de asociación fuerte
Los objetos enlazados son independientes entre sí.	Los objetos enlazados son altamente dependientes entre sí.
Se representa con una línea sólida y una punta de flecha vacía.	Representado por una línea sólida con una punta de flecha rellena.
La agregación se define como una relación " tiene-un ".	La composición se define como una relación "parte-de".

7)

Un diagrama de clases es un tipo de diagrama de modelado estático utilizado en la Programación Orientada a Objetos (POO) para representar la estructura de un sistema. Es parte del lenguaje de modelado UML (Unified Modeling Language), que se utiliza para diseñar y visualizar los diferentes aspectos de un sistema de software.

En un diagrama de clases se muestran las clases, los atributos, los métodos y las relaciones entre las clases (como herencia, agregación, composición, entre otras). El objetivo es proporcionar una vista clara y organizada de cómo se estructuran y se relacionan los objetos dentro de un sistema.

8)

Pago: código*, método, fechaPago

Venta: código*, codigoAuto, codigoVendedor, codigoCliente, codigoPago, monto, fechaVenta

Auto: código*, marca, descripción, precio

Vendedor: código* ,nombre, email, teléfono, nroOficina, cantidadVendida

Cliente: código*, nombre, apellido, email, teléfono