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: Guanco Yonathan Jesus

1. ¿Defina que es una clase y atributo en OOP(Object Oriented Programming)?

- Una clase es una plantilla o modelo que define la estructura y el comportamiento de los objetos en la programación orientada a objetos
- una clase especifica qué datos (atributos) y qué operaciones (métodos) tendrán los objetos creados a partir de ella

Ejempló:

Clase: Automóvil

Atributos:

- Marca
- Modelo
- Año
- Color
- Número de matrícula

Métodos:

- Arrancar()
- Frenar()
- Acelerar()
- Apagar()

2. Clases y Atributos

1. Avión

- Código; Identificador único del avión
- Tipo; Descripción del modelo del avión (ej BOEING-747)
- Base ;Ubicación donde se realizan las revisiones periódicas de mantenimiento

2. Piloto

- Código :Identificador único del piloto
- Nombre :Nombre del piloto
- Horas de vuelo :Total de horas que ha volado el piloto

3. Miembro de Tripulación

- Código :Identificador único del miembro de la tripulación
- Nombre: nombre del miembro de la tripulación

4. Vuelo

- Número de vuelo: Identificador del vuelo (ej IB-8830)
- Origen: Aeropuerto de salida
- Destino: Aeropuerto de llegada
- Hora: Hora programada para el vuelo
- Avión: Referencia al avión asignado para el vuelo
- Piloto :Referencia al piloto que realiza el vuelo
- Miembros de tripulación: Referencias a los miembros de la tripulación asignados al vuelo

5. Base

- Código :Identificador único de la base
- Nombre: Nombre de la base

¿Cuáles son las características de una clase?

Las características de una clase son:

- Atributos: Propiedades o datos que definen el estado de la clase
- Métodos: Funciones que definen el comportamiento de la clase
- Encapsulamiento: Protección de los datos mediante modificadores de acceso
- Herencia: Capacidad de una clase para heredar atributos y métodos de otra
- Polimorfismo: Posibilidad de que diferentes clases implementen métodos con el mismo nombre de manera diferente
- Abstracción: Proceso de ocultar detalles complejos y mostrar solo la información esencial

¿Qué es una interface?

una interfaz es un contrato que define métodos y propiedades que una clase debe implementar, sin proporcionar la implementación. Facilita la reutilización y el intercambio de comportamientos entre diferentes clases

¿Cual es la diferencia entre una clase y una interface?

Implementación:

Clase: Puede contener tanto la definición como la implementación de métodos

Interfaz: Solo define métodos (sin implementación) y propiedades

Herencia:

Clase: Puede heredar de otra clase (una sola clase base en la mayoría de los lenguajes)

Interfaz: Puede ser implementada por múltiples clases, permitiendo la herencia múltiple

Instanciación:

Clase: Se puede instanciar directamente para crear objetos

Interfaz: No se puede instanciar; solo se puede implementar en clases

Acceso a miembros:

Clase: Puede tener modificadores de acceso (público, privado, protegido)

Interfaz: Todos los métodos son implícitamente públicos y no pueden tener modificadores de acceso

Uso:

Clase: Se utiliza para crear objetos con estado y comportamiento

Interfaz: Se utiliza para definir un comportamiento que puede ser compartido entre diferentes clases

¿Cuáles son las relaciones que existen entre clases?

- Herencia: Una clase toma prestadas las características de otra clase
- Composición: Una clase tiene otras clases como parte de su estructura
- Agregación: Una clase incluye otras clases, pero estas pueden existir de forma independiente
- Asociación: Una clase está conectada a otra, usando sus métodos o propiedades
- Dependencia: Una clase usa los servicios o funcionalidades de otra clase, pero solo temporalmente
- Implementación: Una clase sigue un contrato definido por una interfaz

Cual es el concepto de cada una de esas relaciones?

- **Herencia**: Permite que una clase (subclase) herede atributos y métodos de otra clase (superclase), facilitando la reutilización y extensión de código.
- **Composición**: Una clase contiene instancias de otras clases como parte de su estructura, gestionando su ciclo de vida y definiendo una relación "tiene un".
- **Agregación**: Una clase contiene instancias de otras clases, pero la relación es menos estricta; las instancias contenidas pueden existir independientemente de la clase contenedora.
- **Dependencia**: Una clase utiliza otra para realizar una función, sin necesidad de mantener una referencia permanente a ella.
- **Interfaz**: Define un contrato de métodos que una clase debe implementar, permitiendo la intercambiabilidad de distintas implementaciones.
- **Asociación**: Relación entre dos clases donde una clase puede conocer o interactuar con la otra, pudiendo ser unidireccional o bidireccional.

¿Cuáles son las diferencias entre Agregación y Composición?

- Definición
- Agregación: Relación donde una clase es parte de otra, pero los componentes pueden existir independientemente. Ejemplo: una escuela y sus estudiantes.
- Composición: Relación más fuerte donde los componentes no pueden existir sin el todo. Ejemplo: un coche y sus ruedas
  Dependencia
- Agregación: Los objetos pueden existir sin el otro
- Composición: La existencia de un objeto depende del otro
-
  Representación en UML
- Agregación: Línea con punta de flecha vacía
- Composición: Línea con punta de flecha llena
- La agregación es una relación débil; la composición es una relación fuerte. La elección entre ambas depende de cómo se gestionan las dependencias entre los objetos en un sistema

¿Qué es un diagrama de clases?

Un diagrama de clases es una representación gráfica que muestra las clases de un sistema de software, sus atributos, métodos y las relaciones entre ellas, con el objetivo de ilustrar la estructura estática del sistema

Elabore un diagrama de clases para una agencia de alquiler de autos

2. **Clase: Auto**
    - **Atributos:**
        - `id`
        - `marca`
        - `modelo`
        - `año`
        - `color`
        - `disponible`
    - **Métodos:**
        - `marcarComoDisponible()`
        - `marcarComoNoDisponible()`
3. **Clase: Cliente**
    - **Atributos:**
        - `id`
        - `nombre`
        - `direccion`
        - `telefono`
        - `email`
    - **Métodos:**
        - `actualizarInformacion()`
4. **Clase: Alquiler**
    - **Atributos:**
        - `id`
        - `fechaInicio`
        - `fechaFin`
        - `costoTotal`
    - **Métodos:**
        - `calcularCosto()`
        - `cerrarAlquiler()`
5. **Clase: Reserva**
    - **Atributos:**
        - `id`
        - `fechaReserva`
        - `fechaInicio`
        - `fechaFin`
    - **Métodos:**
        - `confirmarReserva()`

        - `cancelarReserva()`

6. **Clase: Factura**
    - **Atributos:**
        - `id`
        - `fecha`
        - `monto`
    - **Métodos:**
        - `generarFactura()`

Relaciones entre las clases:

- **Auto** y **Alquiler**:
    - Relación de asociación (un auto puede ser alquilado en múltiples alquileres, y un alquiler está asociado a un solo auto).
    - Multiplicidad: 1..* (Un auto puede estar asociado con muchos alquileres).
- **Cliente** y **Alquiler**:
    - Relación de asociación (un cliente puede tener múltiples alquileres, y un alquiler está asociado a un solo cliente).
    - Multiplicidad: 1..* (Un cliente puede tener muchos alquileres).
- **Reserva** y **Auto**:
    - Relación de asociación (una reserva se refiere a un auto específico).
    - Multiplicidad: 1..1 (Cada reserva se refiere a un único auto).
- **Reserva** y **Cliente**:
    - Relación de asociación (un cliente puede hacer múltiples reservas).
    - Multiplicidad: 1..* (Un cliente puede hacer muchas reservas).
- **Alquiler** y **Factura**:
    - Relación de asociación (un alquiler genera una factura).
    - Multiplicidad: 1..1 (Cada alquiler genera una única factura)