UNIVERSIDAD NACIONAL DE INGENIERÍA

Unidad I:
El Paradigma Orientado a Objetos

Nombre del docente(s):

Ing. Pablo E. Argeñal

Fecha de elaboración:

Marzo 2021

Contenido

1. COMPARACIÓN Y EVALUACIÓN DE LAS METODOLOGÍAS DE ANÁLISIS Y DISEÑO ORIENTADAS A OBJETOS (METODOLOGÍA OMT POR RUMBAUGH, ANÁLISIS Y DISEÑO ORIENTADO A OBJETOS POR GRADDY BOOCH, ETC)

La tecnología orientada a objeto es una vieja y madura tecnología que se remonta a los años sesenta. Uno de los lenguajes orientados a objetos más antiguos es Simula, desarrollado en 1967, utilizado para hacer simulaciones, creado por Ole-Johan Dahl y Kristen Nygaard del Centro de Cómputo Noruego en Oslo. Este centro trabajaba en simulaciones de naves. La idea surgió al agrupar los diversos tipos de naves en diversas clases de objetos, siendo responsable cada clase de objetos de definir sus *propios* datos y comportamientos.

El diseño OO difiere considerablemente del diseño estructurado. En diseño OO no se realiza un problema en términos de tareas o procesos (subrutinas) ni en término de datos, sino que se analiza el problema como un sistema de objetos que interactúan entre si.

Un problema desarrollado con técnicas orientadas a objetos requiere en primer lugar, responder a una pregunta **¿Cuáles son los objetos del programa?** Como los objetos son instancias de clases, en realidad la primera etapa en un desarrollo orientado a objetos exige la identificación de clases y posteriormente son atributos y comportamiento, así como las relaciones entre clases y su posterior implementación en un lenguaje de programación.

Existen numerosos métodos de desarrollo orientado a objetos: Booch, Yourdon/Coad, Martin, Shlaer & Mellor, Rumbaugh, etc.

En general un proyecto orientado a objeto se compone de las siguientes etapas:

- Análisis orientado a objetos (**AOO)**
- Diseño orientado a objetos (**DOO)**
- Programación orientada a objetos (**POO)**

Al no estar delimitadas las etapas o fases de análisis y diseño en orientación a objetos, sintetizamos las ideas claves de las diferentes metodologías en una única secuencia de pasos que encuadraremos dentro de la denominación *diseño o desarrollo orientado a objetos (**DOOO).***

Análisis Orientado a Objetos

El método de Booch´93 considera que las etapas del denominado microproceso en un desarrollo orientado a objetos son:

- Identificar las clases y objetos en un nivel dado de abstracción
- Identificar la semántica de estas clases y objetos
- Identificar las relaciones entre clases y objetos
- Especificar el interfaz y la implementación de estas clases y objetos.

Estas etapas suelen seguirse por la mayoría de los métodos de diseño OO. Así otros métodos de diseño OO contienen las etapas siguientes:

- Identificar clases y objetos.
- Asignar atributos y comportamiento.
- Encontrar relaciones entre clases
- Organizar las clases en jerarquías
- Implementar las especificaciones de diseño.

Identificar clases y objetos

Los objetos son las entidades fundamentales de un programa orientado a objetos, y las clases son conjuntos de objetos, por esta razón, la identificación de ambas entidades son la primera etapa en cualquier desarrollo OO que se realice.

El mejor sistema para identificar objetos es leer la definición o descripción de especificaciones del problema y localizar nombres o frases con nombre y verbos o frases con verbos. Los ***nombres*** son buenos indicadores de la existencia de objetos en el modelo OO y los ***verbos*** son candidatos a métodos o funciones miembro.

Sin embargo no todos los nombre identificados son buenos candidatos a clases. Después de encontrar todos los nombres se examina la lista y se decide que nombres son realmente clases en nuestro sistema.

Por ejemplo: un nombre que no puede ser una clase es un número de la Seguridad Social. Contiene datos, pero ninguna acción sobre el cálculo.

Los problemas potenciales que se pueden encontrar en un sistema son:

1. Encontrar demasiados objetos.
2. Alguno nombres son variables, no objetos. (Numero de seguro social)

Algunos criterios (Shlaer y Mellor) que se pueden seguir para clasificar objetos son:

- Cosas tangibles (mesa despacho, carro, base de datos)
- Interacciones (matrimonio, ventas, compras)
- Papeles-roles (director, propietario, cliente)
- Incidentes (viaje, transacción, llegada)
- Unidades organizacionales (departamentos, divisiones)
- Lugares, posiciones geográficas o físicas (ciudad, camino)
- Especificaciones (descripciones o estándares).

Consideremos algunos casos concretos:

1. Identificar clases para un programa que modele objetos físicos. Ej.: un programa que maneje reservas de líneas aéreas, posiblemente se necesitará una clase **Avión** y una clase **Pasajero**. Si el programa es un sistema operativo, se necesitará una clase **Dispositivo** que representa unidades de disco e impresoras.

2. Para los programas que no modelan entidades físicas, se deben identificar las entidades conceptuales que manipulan al programa. En programa de dibujo gráfico se utilizará las clases **Rectángulo y Circulo**. Un compilador necesitara una clase **ArbolSintáctico** y un sistema operativo una clase **Proceso**.

3. Los sucesos (cosas que suceden a un objeto) e iteraciones (cosas que suceden entre objetos). Ej.: Una clase **Transacción** puede representar cosas como depósitos, préstamos, transferencias en un programa bancario. Una clase **Orden** puede representar acciones realizadas por el usuario.

4. Estructuras de datos, tales como pilas, colas listas, etc.

Asignaciones de atributos y comportamiento

Una vez que se ha identificado una clase, la siguiente tarea es determinar qué responsabilidades tiene. Las cuales caen dentro de dos categorías:

- La información que un objeto de una clase ha de mantener (los atributos: que conoce un objeto de esta clase?).

- Las operaciones que un objeto puede realizar o se pueden realizar sobre el mismo (el comportamiento: que puede hacer este objeto?).

2. SIMBOLOGIA Y NOTACION DEL DISEÑO ORIENTADO A OBJETOS

El análisis y diseño OO de cualquier problema y su posterior implementación requiere de buenas notaciones gráficas.

Una buena notación OO debe contener como mínimo símbolos que representen:

- Clases (datos y operaciones: atributos y métodos.
- Objetos (Instancias de clases).
- Herencia
- Agregación
- Comunicación y relaciones entre objetos y clases
- Relaciones de asociación
- Uso.
- Instanciación.

Notación de BOOCH'93

Incorpora dos tipos distintos de diagramas:

1. Diagrama de clases
2. Diagrama de objetos

Cada diagrama tiene su propia notación y un conjunto de iconos (Figura 1.)

(a) (b) (c)

Los diagramas de clases documentan la estructura de clases, sus relaciones con otras y las relaciones generales de sus instancias. Los diagramas de objetos muestran la interacción entre objetos.

Relaciones entre clases

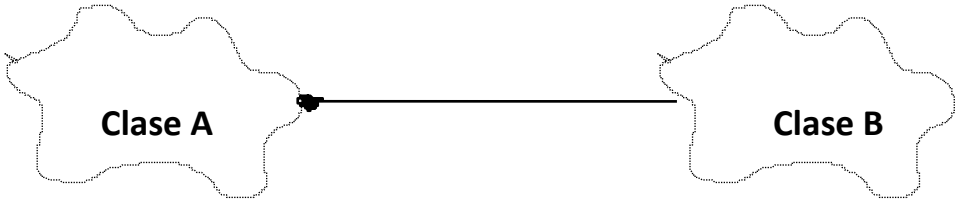
Las relaciones entre clases que contempla la notación de Booch son:

1. Generalización/especialización (hereda).
2. Agregación.
3. Asociación.
4. Uso.

La relación de herencia entre clases muestra que la subclase comparte la estructura o comportamiento definido en una o más superclases. La relación de herencia muestra una relación **es-un** entre clases. La flecha apunta hacia la clase base.

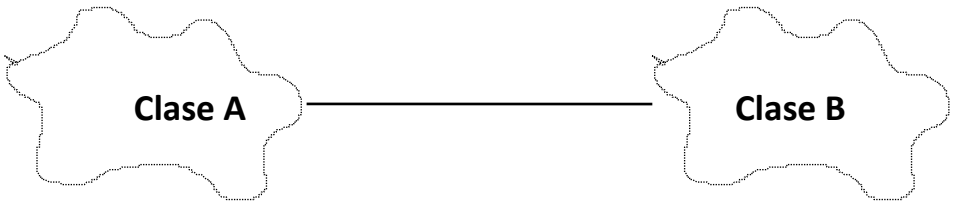
Relación hereda

La relación de agregación muestra una relación entre clase **todo-y-partes**(parte-de). Se conoce también como relación **tiene-un** (has-a).La clase en el extremo fuente de la relación **tiene** se llama a veces la clase agregada. El objeto agregado está constituido físicamente a partir de otros objetos.



Relación agregación

La relación de asociación representa una conexión semántica entre dos clases. Las asociaciones son bidirecionales; son las relaciones más generales de todas y las más débiles semánticamente.



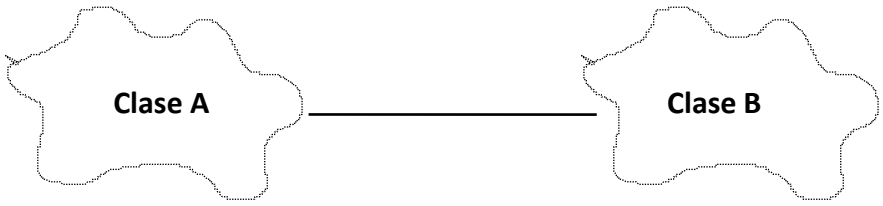
Relación hereda de asociación

La relación de uso entre dos clases muestra que la clase fuente depende de la clase destino para proporcionar servicios, tales como:

1. La clase fuente accede a un valor (constante o variable) definido en la clase destino.

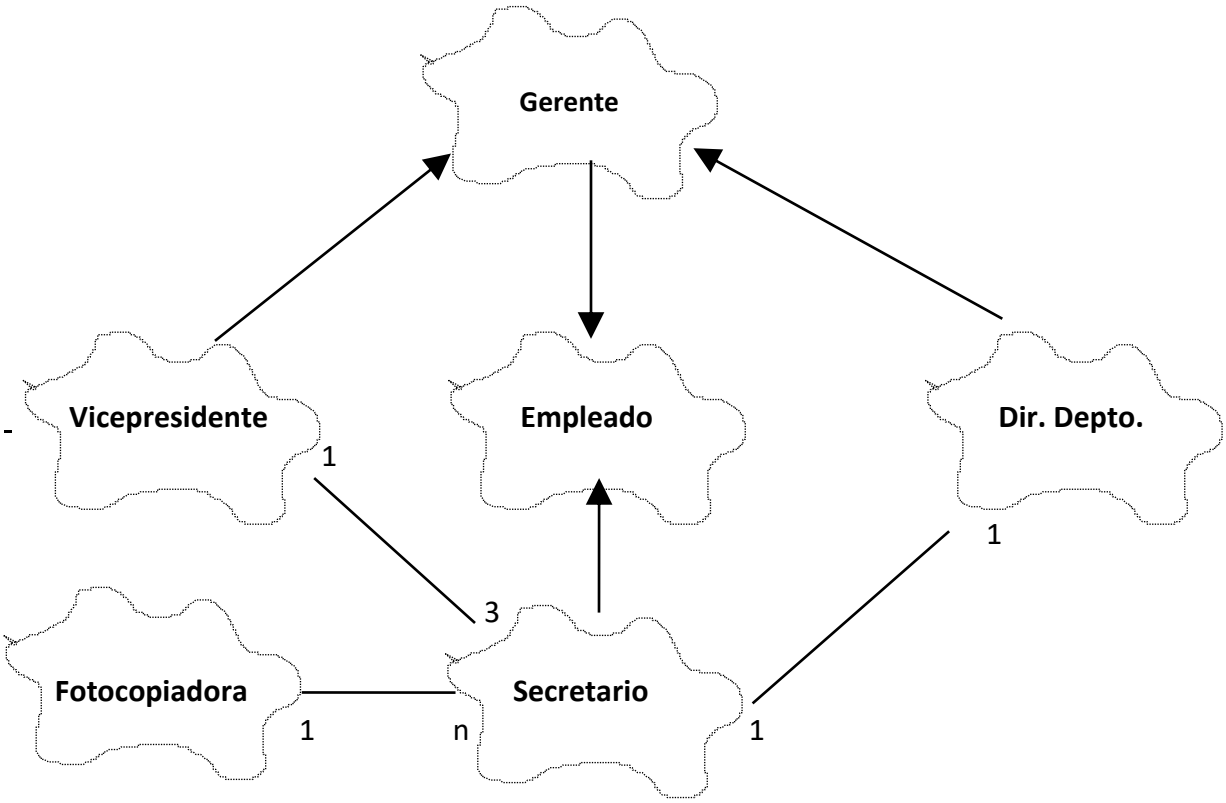
2. Operaciones de la clase fuente invocan operaciones de la clase destino.

El circulo abierto designa la clase fuente y el otro extremo es la clase destino. La relación de uso es, desde el punto de vista práctico, una relación *cliente - servidor* en la que una clase cliente utiliza los servicios de otra clase servidor.



Relación de uso

La siguiente figura muestra un diagrama de clases Empleado con relaciones de herencia entre clase (clase derivadas Secretario y Gerente) y relaciones de agregación o composición (cada Director_de_Departamento tiene una Secretaria y cada Vicepresidente tiene tres Secretarias, cada objeto de la clase Secretaria puede acceder a una fotocopiadora y cada fotocopiadora puede servir para muchos secretarios).



NOTACION DE RUMBAUGH (OMT)

Es una de las más ricas existentes en la actualidad. Los iconos de clases y objetos, como instancias de la clase, se representan en la siguiente figura:

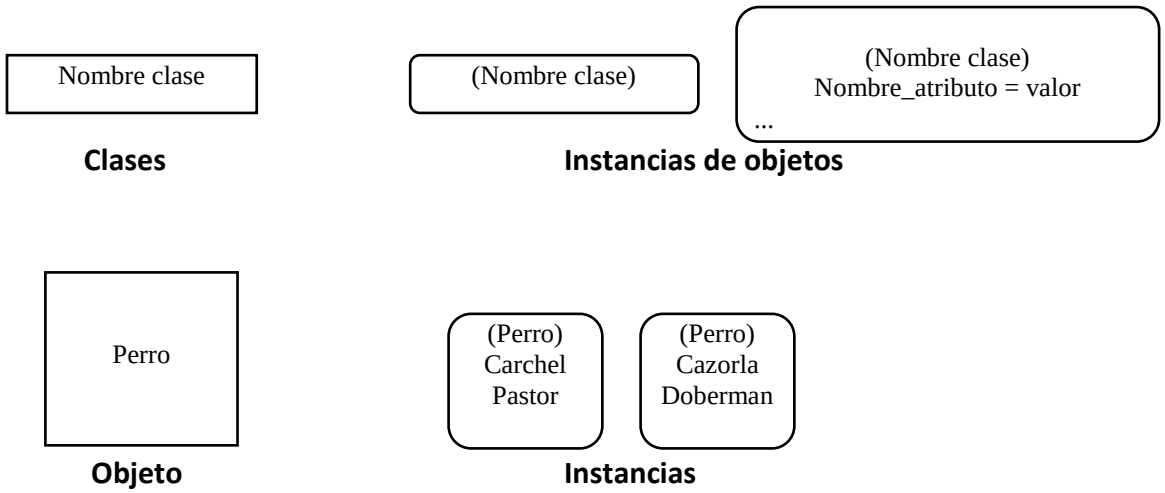


Figura Clases y objetos

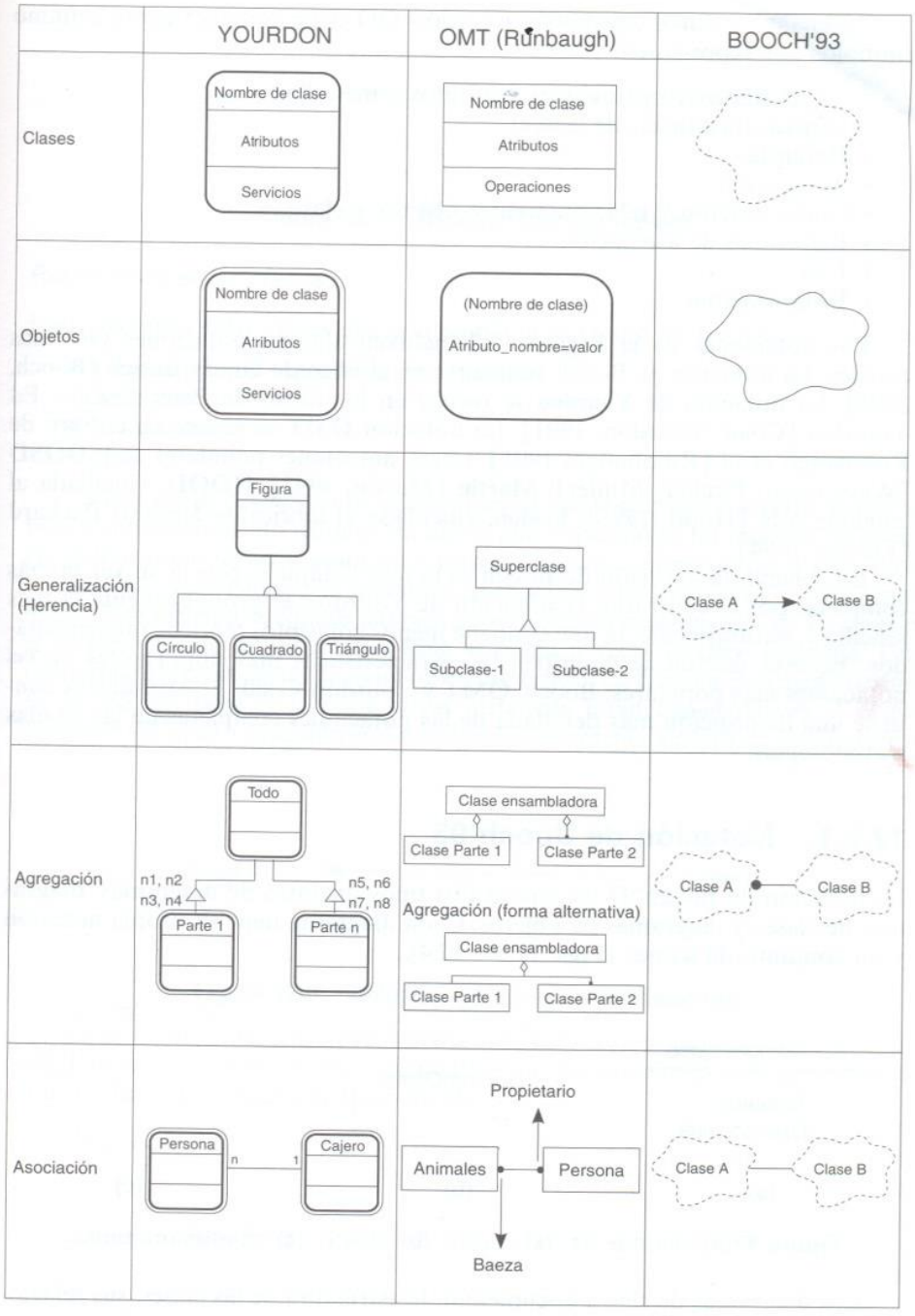
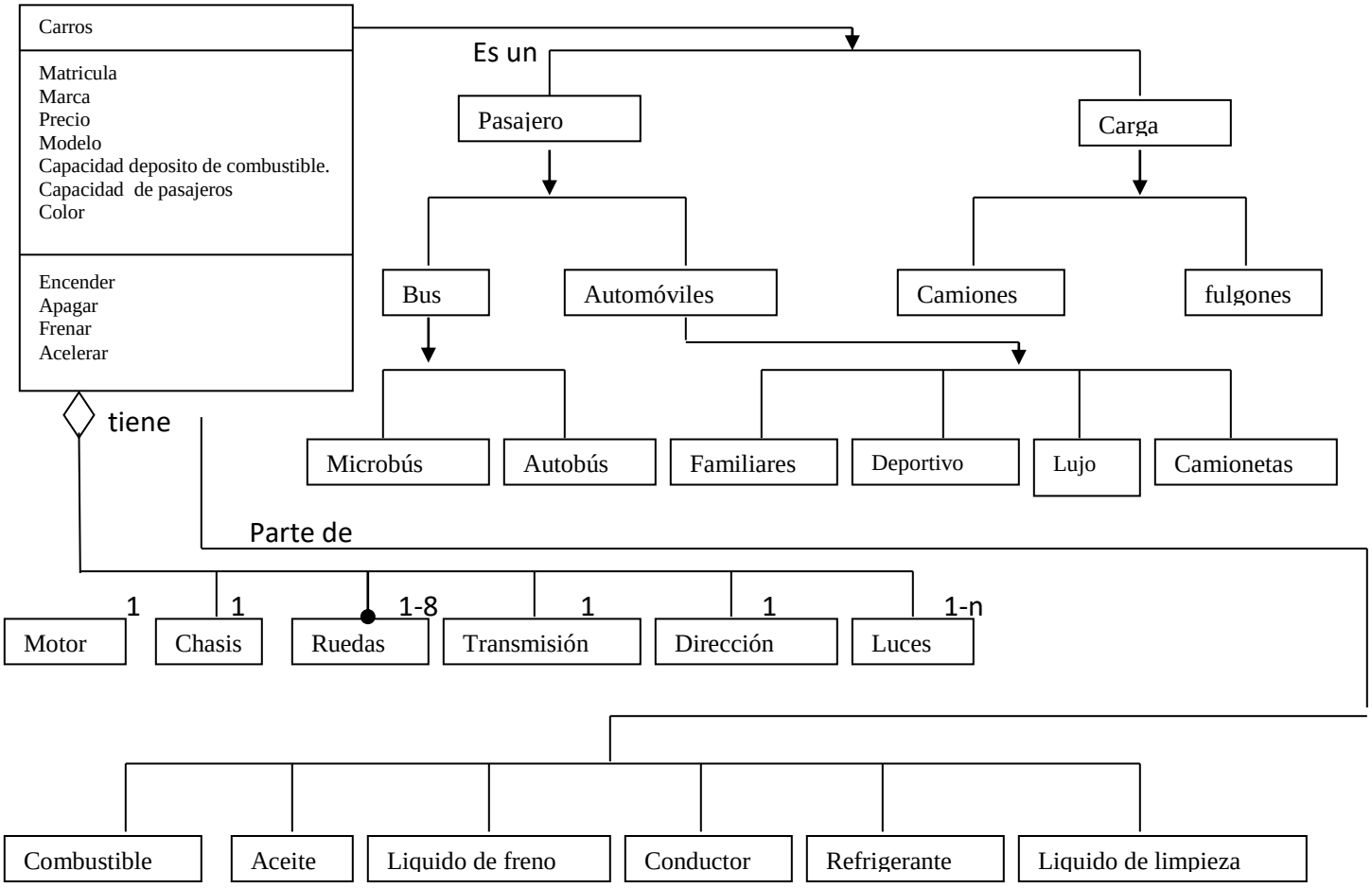


Figura 12.2. Notaciones orientadas a objetos.

3. Ejemplos

1. Utilizando la notación indicada, cree un modelo extendido para clasificar los carros, de acuerdo al siguiente ejemplo mostrado.



2. Tomando en cuenta los conceptos abordados sobre OO y la notación descrita para crear modelos OO, desarrolle el modelo OO de un sistema de Biblioteca.
3. Desarrollar el modelo para un AutoLote, con ayuda de la notación descrita.
4. Desarrollar el modelo de un Restaurante.
5. Desarrollar el modelo de PaintBrush de Windows.
6. De acuerdo al siguiente enunciado construya el modelo OO y las clases: En una Empresa trabajan diferentes tipos de empleados, incluyendo oficinistas y técnicos. Para todos los empleados se guarda su nombre, dirección, edad y salario. Los oficinistas tienen teléfono asignados, mientras que los técnicos tienen seguro de vida, además de un turno.

4. Bibliografía

BIBLIOGRAPHY Aguilar, L. J. (1996). *Programación Orientada a Objetos Conceptos, Modelado, Diseño y Codificación en C++*. Madrid: McGraw-Hill.

Graham, I. (1996). *Métodos Orientados a Objetos.* Addison - Wesley/Díaz de Santos.