UNIDAD I - PROBLEMAS CON TIPOS DE DATOS SIMPLES

OBJETIVOS DE LA PRIMER SEMANA

Clase teórica
- Presentación de la materia, condiciones de cursado, regularización, examen, etc.
- Objetivos de la materia.
- Resolución de problemas (Análisis del problema, Diseño del algoritmo y resolución del problema por computadora).
- Análisis y programación orientada a objetos.

Clase práctica
- Ejercicios de identificación de entidades.
- Repaso de matemática discreta (diagramas de flujo).

OBJETIVOS DE LA SEGUNDA SEMANA

Clase teórica
- Algoritmos (Concepto, representación gráfica, escritura de algoritmos, estructuras de control)
- Programas (Concepto, Organización, Programas tipos, Elementos básicos y Esquema general).
- Lenguajes de programación.

Clase práctica
- Ejercicios con clases (en diagramas) y diagramas de flujo.

OBJETIVOS DE LA TERCER SEMANA

Clase teórica
- Lenguaje C++.
- Elementos del lenguaje.
- Clases en C++ (Estructura general, declaración y definición, cuerpo de una clase. Instancias de una clase (objetos) (acceso a los miembros, la vida de un objeto), declaración de miembros de clases (modificadores de acceso a miembros de clases) y atributos.

Clase práctica
- Ejercicios de clases con código y diagramas de flujo.
  - Mínimo y Máximo en una secuencia de números
  - Pares, Impares en una secuencia de números

OBJETIVOS DE LA CUARTA SEMANA

Clase teórica
- Clases en C++ (continuación métodos: Declaración y definición, el cuerpo, llamadas a métodos el objeto actual (this), métodos especiales: sobrecargados, constructores y destructores).
- Problemas tipos.
- Funciones básicas de caracteres.

Clase práctica
- Ejercicios de clases con código y diagramas de flujo.
  - Secuencia ascendente de caracteres
  - Términos en serie aritmética
  - Tratamiento de caracteres

Material disponible en el sitio labsys.

Autor: Guzmán, Analía - Tymoschuk, Jorge
Compaginación: Guzmán, Analía

SEMANA Nº 1

OBJETIVOS DE LA PRIMER SEMANA

Clase teórica

- Presentación de la materia, condiciones de cursado, regularización, examen, etc.
- Objetivos de la materia.
- Resolución de problemas (Análisis del problema, Diseño del algoritmo y resolución del problema por computadora).
- Análisis y programación orientada a objetos.

Clase práctica

- Ejercicios de identificación de entidades.
- Repaso de matemática discreta (diagramas de flujo).

CLASE TEORICA

OBJETIVOS DE LA MATERIA

- Desarrollar la capacidad de razonamiento y lógica necesarias para Programar.
- Abordar problemas reales, analizarlos, definir la estrategia de su resolución, explicitar los algoritmos necesarios y codificarlos en un lenguaje de programación.
- Lograr entrenamiento en:
    1. Interpretar el problema (ANALISIS)
    2. Reducirlo a lo esencial (ABSTRACCION)
    3. Formular la estrategia de resolución (DIAGRAMA)
    4. Verificar dicha estrategia (PRUEBA DE ESCRITORIO)
    5. Codificar en un lenguaje de programación.
    6. Depurar errores de codificación.

Pruebas, correcciones,...

1. COMPRENSION DEL PROBLEMA (ANALISIS)

El análisis del problema va a depender de la visión y del método que se utilice para interpretarlo. Estos métodos o modelos básicos son los que se llaman paradigmas de programación y le indican al programador como debe encarar la visión del mundo real. Un paradigma de programación es un modelo básico de análisis, diseño e implementación de programas.

2. ABSTRACCION

Describir lo esencial de un proceso haciendo abstracción de los detalles, para poder asimilarlo a otro y hacer uso de técnicas y conocimientos ya adquiridos. Una vez que contamos con todos los elementos necesarios claramente definidos podemos proceder a desarrollar la **estrategia** que nos llevará a solucionar el problema.

3. ESTRATEGIA

Es un plan cuidadosamente elaborado que describe en líneas generales, el conjunto de pasos a seguir para solucionar la situación planteada. Desglosa el problema inicial en problemas más simples, (acciones elementales) que pueden ser solucionadas individualmente.

4. VERIFICAR DICHA ESTRATEGIA (PRUEBA DE ESCRITORIO)

5. CODIFICAR EN UN LENGUAJE DE PROGRAMACION.

6. DEPURAR ERRORES DE CODIFICACION.

7. PRUEBAS, CORRECCIONES,...

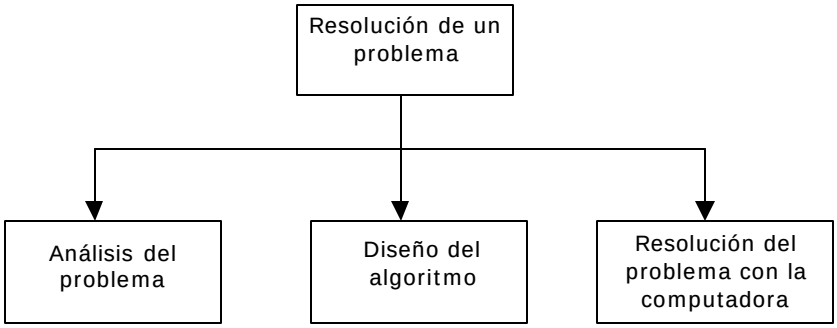
RESOLUCION DE PROBLEMAS

La principal razón para que las personas aprendan a programar en general y los lenguajes de programación en particular es utilizar la computadora como una herramienta para la resolución de un problema se puede dividir en tres partes importantes:

1. Análisis del problema
2. Diseño o desarrollo del algoritmo
3. Resolución del algoritmo en la computadora.

El primer paso, análisis del problema, requiere que el problema sea definido y comprendido claramente para que pueda ser analizado con todo detalle. Una vez analizado el problema, se debe

desarrollar el algoritmo, procedimiento paso a paso para solucionar el problema dado. Por último, para resolver el algoritmo mediante una computadora se necesita codificar el algoritmo en un lenguaje de programación, convertir el algoritmo en programa, ejecutarlo y comprobar que el programa soluciona verdaderamente el problema. Las partes del proceso de resolución de un problema mediante computadora se ve en el gráfico.



1. ANALISIS DEL PROBLEMA

El propósito del análisis de un problema es ayudar al programador para llegar a una cierta comprensión de la naturaleza del problema. El problema debe estar bien definido si se desea llegar a una solución satisfactoria.

Para poder definir con precisión el problema se requiere que las especificaciones de entrada y salida sean descritas con detalle. Una buena definición del problema, junto con una descripción detallada de las especificaciones de entrada y salida, son los requisitos más importantes para llegar a una solución eficaz.

El análisis del problema exige:

1. Elección de la forma de encarar el problema y su resolución a través de un método adecuado, como por ejemplo: estructurado, orientado a objetos, etc.
2. Una vez elegida la forma de trabajo realizar una lectura previa del problema a fin de obtener una idea general de lo que se solicita.
3. La segunda lectura deberá servir para responder a las preguntas: ¿Qué información debe proporcionar la resolución del problema? ¿Qué datos se necesitan para resolver el problema?

El paso 1 es obligado ya que todos los precedentes dependen de esta elección. En nuestra materia veremos el análisis y la programación desde el punto de vista orientado a objetos.

La respuesta a la primera pregunta indicará los resultados deseados o las *salidas del problema.* La respuesta a la segunda indicará qué datos se proporcionan o las *entradas del problema.*

Por ejemplo:

PROBLEMA

Leer el radio de un circulo y calcular e imprimir su superficie y longitud.

ANALISIS

Para resolver este problema utilizando la filosofía de objetos podríamos seguir los siguientes pasos:

1. Identificar el objeto: identificar la entidad que engloba al problema, en nuestro ejemplo es el círculo.

2. Identificar sus atributos: podemos decir que todo círculo tiene como datos esenciales el radio, además, nuestro problema en particular necesitará el valor de la superficie y de la longitud, que son datos calculables a partir del radio y que pueden figurar como atributos o no dependiendo si se quiere guardar su estado en el objeto. Como nuestro enunciado no nos restringe la forma de implementarlo vamos a definir como atributos el radio, la superficie y la longitud.

3. Identificar sus métodos: los métodos a elegir siempre dependerán de las operaciones que podemos realizar con los atributos, entonces podemos enumerar:

Inicializar el radio: Inicializar en 0, por ejemplo, el radio.
Inicializar la superficie: Inicializar en 0, por ejemplo, la superficie.
Inicializar la longitud: Inicializar en 0, por ejemplo, la longitud.
Ingresar el radio: Se ingresa solo el valor del radio ya que el resto de los atributos se calculan a partir de este.
Mostrar el radio: Mostrar el valor del radio.
Mostrar la superficie: Mostrar el valor de la superficie.
Mostrar la longitud: Mostrar el valor de la longitud.
Calcular la superficie: formula que devuelve la superficie
Calcular la longitud: formula que devuelve la longitud.

2. DISEÑO DEL ALGORITMO

El análisis del problema nos resuelve el problema de identificar las entidades (objetos), junto con sus datos (atríbutos) y sus procedimientos (métodos), pero a partir de allí hay que comenzar a desarrollar el cuerpo de los métodos, para ello se necesitará un algoritmo.

El significado de algoritmo es similar al de receta, método, técnica, procedimiento o rutina, y se define como "Un conjunto finito de reglas diseñadas para crear una secuencia de operaciones para resolver un tipo específico de problemas.

También se define algoritmo como una estrategia que exige precisión en las instrucciones (descripción de los pasos a seguir). Donde cada instrucción debe ser:

- Clara y precisa para que no surja ninguna duda en cuanto a su ejecución.
- Sencilla para que no surja ninguna duda en su comprensión y pueda ser ejecutada sin dificultad.

Una computadora no tiene capacidad para solucionar problemas más que cuando se le proporcionan los sucesivos pasos a realizar. Estos pasos sucesivos que indican las instrucciones a ejecutar por la máquina constituyen el *algoritmo*

Los algoritmos se pueden representar a través de herramientas tales como: diagramas de flujo, pseudocódigos o diagramas N.-S.

3. RESOLUCION DEL PROBLEMA POR COMPUTADORA

Una vez que las entidades y el algoritmo están diseñados y representados gráficamente se debe pasar a la fase de resolución práctica del problema con la computadora.

Esta fase se descompone a su vez en las siguientes subfases:

1. Codificación del algoritmo en un programa.
2. Ejecución del programa.
3. Comprobación del programa.

En el diseño del algoritmo, éste se describe en una herramienta de programación tal como un diagrama de flujo o pseudo código. Sin embargo, el programa que implementa el algoritmo debe ser escrito en un lenguaje de programación y siguiendo las reglas gramaticales o sintaxis del mismo. La fase de conversión del algoritmo en un lenguaje de programación se denomina *codificación,* ya que el algoritmo en lenguaje de programación se denomina código.

ANALISIS Y PROGRAMACION ORIENTADA A OBJETOS

1. INTRODUCCIÓN A LA PROGRAMACIÓN CENTRADA EN EL CONCEPTO (ORIENTADA A OBJETOS)

La orientación a objetos es una forma natural de pensar en relación con el mundo y de escribir programas de computación. Mire a su alrededor. Por todas partes: *¡objetos!* Personas, animales, plantas, automóviles, aviones, edificios, cortadoras de pastos, computadoras y demás. Cada una de ellas tiene ciertas características y se comporta de una manera determinada. Si las conocemos, es porque tenemos el *concepto de lo que son.* Conceptos persona, objetos persona. Los seres humanos **pensamos en términos de objetos**. Tenemos la capacidad maravillosa de la *abstracción,* que nos permite ver una imagen en pantalla como personas, aviones, árboles y montañas, en vez de puntos individuales de color.

Todos estos objetos tienen algunas cosas en común. **Todos tienen atributos**, como tamaño, forma, color, peso y demás. Todos ellos exhiben **algún comportamiento.** Un automóvil acelera, frena, gira, etcétera. El objeto persona habla, ríe, estudia, baila, canta ...

Los seres humanos aprenden lo relacionado con los objetos estudiando sus atributos y observando su comportamiento. Objetos diferentes pueden tener muchos atributos iguales y mostrar comportamientos similares. Se pueden hacer comparaciones, por ejemplo, entre bebes y adultos, entre personas y chimpancés. Automóviles, camiones, pequeños carros rojos y patines tienen mucho en común.

La programación orientada a objetos (POO) **hace modelos de los objetos del mundo real** mediante sus contrapartes en software. Aprovecha las relaciones de clase, donde objetos de una cierta clase, como la clase de vehículos, tienen las mismas características. Aprovecha las relaciones de *herencia,* donde clases recién creadas de objetos se derivan heredando características de clases existentes, pero también poseyendo características propias de ellos mismos. Los bebes tienen muchas características de sus padres, pero ocasionalmente padres de baja estatura tienen hijos altos.

La programación orientada a objetos nos proporciona una forma más **natural e intuitiva** de observar el proceso de programación, es decir *haciendo modelos* de objetos del mundo real, de sus atributos y de sus comportamientos. POO también hace modelos de la comunicación entre los objetos. De la misma forma que las personas se envían *mensajes* uno al otro los objetos también se comunican mediante mensajes.

La **POO** *encapsula* **datos (atributos) y funciones (comportamiento)** en paquetes llamados *objetos;* los datos y las funciones de un objeto están muy unidos. Los objetos tienen la propiedad de *ocultar* la *información.* Esto significa que aunque los objetos puedan saber cómo comunicarse unos con otros mediante *interfaces* bien definidas, a los objetos por lo regular no se les está permitido saber cómo funcionan otros objetos. Los detalles de puesta en práctica quedan ocultos dentro de los objetos mismos. (*Casi estamos diciendo que existe entre ellos el respeto a la intimidad ...*) A esto se le llama *Encapsulamiento.*

2. QUE ES LA PROGRAMACION ORIENTADA A OBJETOS

Es una técnica o estilo de programación que utiliza objetos como bloque esencial de construcción. Los programas se organizan como colecciones de **objetos** que colaboran entre sí enviándose mensajes. Solo se dispone de "*objetos que colaboran entre sí*". Por lo tanto un programa orientado a objetos viene definido por la ecuación:

Objetos + Mensajes = Programa

3. COMPONENTES BÁSICOS

Objetos

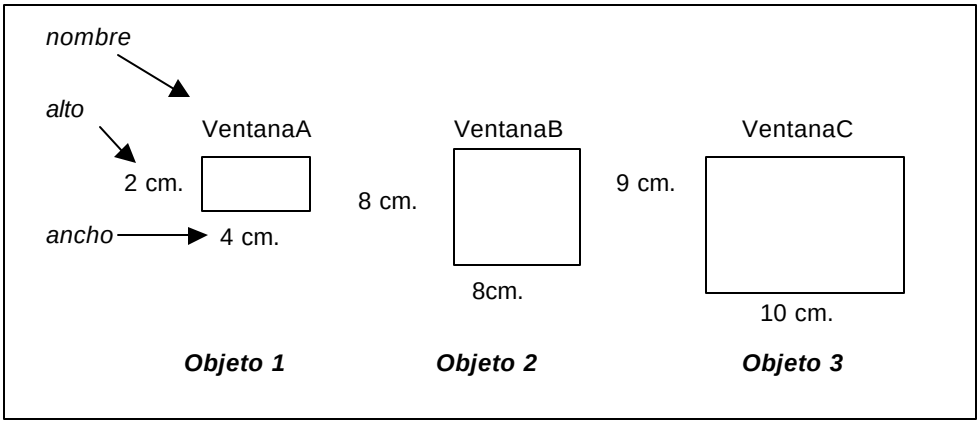
Es el elemento fundamental de la programación orientada a objetos. Es una entidad que posee atributos y métodos, los atributos definen el estado de mismo y los métodos definen su comportamiento.

Cada objeto forma parte de una organización, no es un ente aislado, sino que forma parte de una organización jerárquica o de otro tipo.

¿Qué son objetos en POO? La respuesta es cualquier entidad del mundo real que se pueda imaginar:

- *Objetos físicos*
  - Automóviles en una simulación de tráfico.
  - Aviones en un sistema de control de tráfico aéreo.
  - Componentes electrónicos en un programa de diseño de circuitos.
  - Animales mamíferos.
- *Elementos de interfaces gráficos de usuarios*
  - Ventanas.
  - Objetos gráficos (líneas, rectángulos, círculos).
  - Menús.
- *Estructuras de datos*
  - Vectores.
  - Listas.
  - Arboles binarios.
- *Tipos de datos definidos por el usuario*
  - Números complejos.
  - Hora del día.
  - Puntos de un plano.

Como ejemplo de objeto, podemos decir que una ventana es un objeto que puede tener como atributos: **nombre, alto, ancho, etc.** y como métodos: **crear la ventana, abrir la ventana, cerrar la ventana, mover la ventana, etc.**



Métodos

Podemos definir un método como un programa procedimental escrito en cualquier lenguaje, que está asociado a un objeto determinado y cuya ejecución sólo puede desencadenarse a través de un mensaje recibido por éste o por sus descendientes.

Mensajes

Un mensaje es simplemente una petición de un objeto a otro para que éste se comporte de una determinada manera, ejecutando uno de sus métodos. La técnica de enviar mensajes se conoce como *paso de mensajes*.

Los mensajes relacionan unos objetos con otros y con el mundo exterior.

La figura 1. Representa un objeto A (emisor) que envía un mensaje Test al objeto B (receptor).

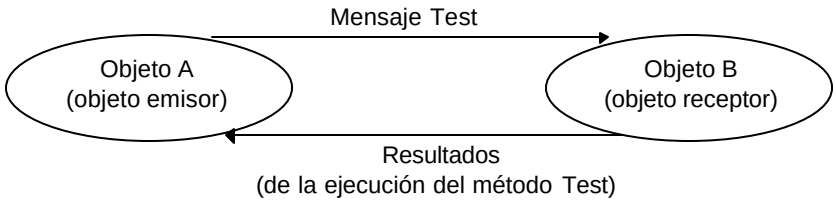


Figura 1. Paso de mensajes entre objetos.

Usando el ejemplo anterior de ventana, podríamos decir que algún objeto de Windows encargado de ejecutar aplicaciones, por ejemplo, envía un mensaje a nuestra ventana para que se abra. En este ejemplo el objeto emisor es un objeto de Windows y el objeto receptor es nuestra ventana, el mensaje es la petición de abrir la ventana y la respuesta es la ejecución del método abrir ventana.

Clases

Una clase es simplemente un modelo que se utiliza para describir uno o más objetos el mismo tipo.

Cada vez que se construye un objeto de una clase, se crea una instancia de esa clase. Por consiguiente, los objetos son instancias de clases. En general, los términos objetos e instancias de una clase se pueden utilizar indistintamente.

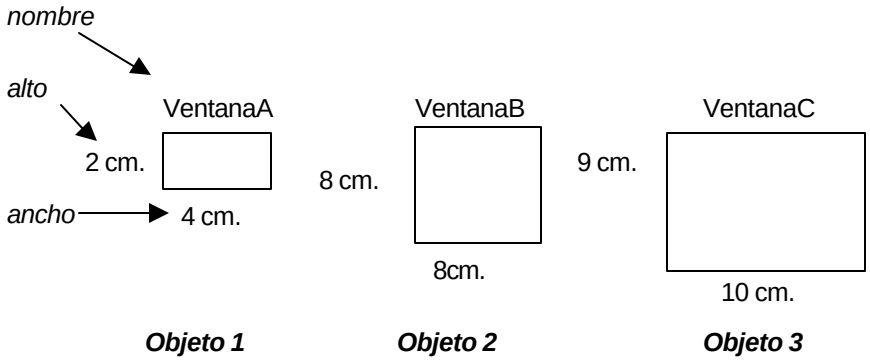
Una clase puede tener muchas instancias y cada una es un objeto independiente.

Siguiendo con el ejemplo de ventana, el modelo o clase para todas las ventanas sería:

Clase ventana

Atributos: nombre, alto, ancho.
Métodos: crear, abrir, cerrar, cambiar tamaño.

Objetos de tipo ventana



4. CARACTERISTICAS

Abstracción

La abstracción se define como la *"extracción de las propiedades esenciales de un concepto"*. Permite no preocuparse de los detalles no esenciales. Implica la identificación de los atributos y métodos de un objeto.

Es la capacidad para encapsular y aislar la información de diseño y ejecución.

Encapsulamiento

Toda la información relacionada con un objeto determinado está agrupada de alguna manera, pero el objeto en sí es como una caja negra cuya estructura interna permanece oculta, tanto para el usuario como para otros objetos diferentes, aunque formen parte de la misma jerarquía. La información contenida en el objeto será accesible sólo a través de la ejecución de los métodos adecuados.

Herencia

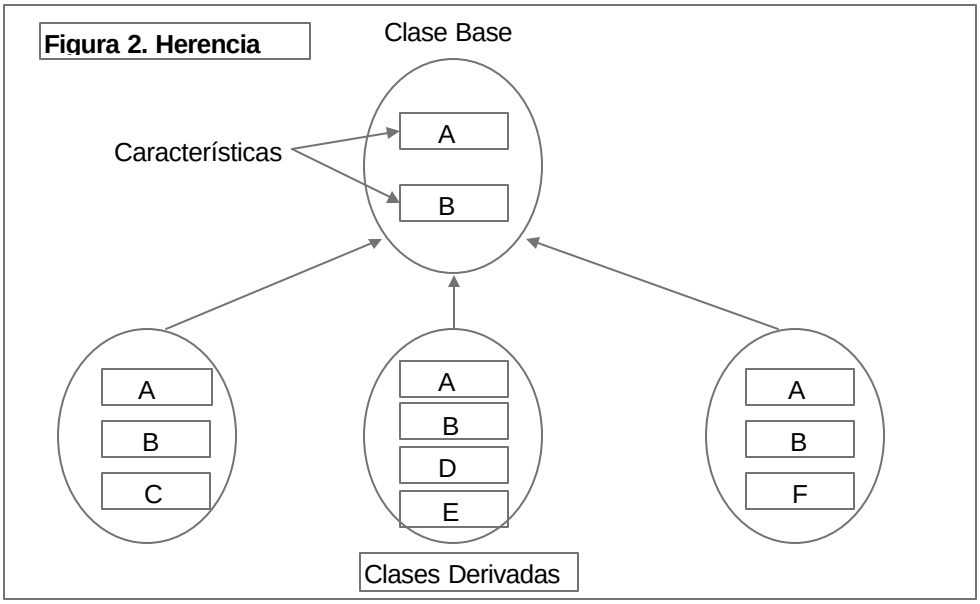
La herencia es la propiedad que permite a los objetos construirse a partir de otros objetos. El concepto de herencia está presente en nuestras vidas diarias donde las clases se dividen en subclases. Así por ejemplo, las clases de animales se dividen en mamíferos, anfibios, insectos, pájaros, etc. La clase de vehículos se divide en automóviles, autobuses, camiones, motocicletas, etc.

El principio de este tipo de división es que cada subclase comparte características comunes con la clase de la que se deriva. Los automóviles, camiones autobuses y motocicletas -que pertenecen a la clase vehículo- tienen ruedas y un motor; son las características de vehículos. Además de las características compartidas con otros miembros de la clase, cada subclase tiene sus propias características particulares: autobuses, por ejemplo, tienen un gran número de asientos, un aparato de televisión para los viajeros, mientras que las motocicletas tienen dos ruedas, un manillar y un asiento doble.

La idea de herencia se muestra en la figura 2. Observen en la figura que las características A y B que pertenecen a la clase base, también son comunes a todas las clases derivadas, y a su vez estas clases derivadas tienen sus propias características.

La herencia impone una relación jerárquica entre clases en la cual una clase *hija* hereda de su clase *padre*. Si una clase sólo puede recibir características de otra clase base, la herencia se denomina *herencia simple*.

Si una clase recibe propiedades de más de una clase base, la herencia se denomina *herencia múltiple*.



Como ejemplo, supongamos que tenemos que registrar los datos de alumnos y profesores, para ello vamos a analizar el problema y refinar la solución en varios pasos:

Primer paso: Identificamos los atributos esenciales para cada entidad.

Alumno	**Profesor**
Nombre	Nombre
Domicilio	Domicilio
Telefono	Telefono
Legajo	Legajo
Carrera	Titulos
Materias	Cursos

Segundo paso: Si observamos, los dos tienen atributos en común: Nombre, Domicilio y Telefono que se corresponden con los datos personales de cualquier persona, y datos particulares para el alumno y el docente. Entonces podríamos escribir los atributos de la siguiente forma:

Alumno	**Profesor**
Datos personales	Datos personales
Datos del alumno	Datos del profesor

Tercer paso: Si agrupamos los datos personales en una clase llamada Persona con los datos comunes a las dos entidades, formaremos lo siguiente:

Persona
Datos Personales

Alumno	**Profesor**
Es una persona	Es una persona
(porque tiene datos de persona)	(porque tiene datos de persona)
Tiene datos del alumno	Tiene datos del profesor

Cuarto paso: Si completamos los atributos, vamos a obervar que hemos ahorrado atributos, ya que en el primer paso había atributos que se repetían en las dos entidades y ahora hay tres entidades en donde cada una tiene los atributos que le corresponden sin repetición, pero alumno y profesor van a heredar de persona los atributos que le correspondan.

Persona
Nombre
Domicilio
Telefono

Alumno	**Profesor**
Hereda los datos de persona	Hereda los datos de persona
Legajo	Legajo
Carrera	Titulos
Materias	Cursos

Polimorfismo

En un sentido literal, *polimorfismo* significa cualidad de tener más de una forma. En el contexto de POO, el polimorfismo se refiere al hecho que una misma operación puede tener diferente comportamiento en diferentes objetos. En otras palabras, diferentes objetos reaccionan al mismo mensaje de modo diferente.

Por ejemplo, consideremos la operación sumar. En un lenguaje de programación el operador + representa la suma de dos números (x + y) de diferentes tipos: enteros, coma flotante, ... Además se puede definir la operación de sumar dos cadenas : concatenación mediante el operador suma.

De modo similar, supongamos un número de figuras geométricas que responden todas al mensaje, **dibujar**. Cada objeto reacciona a este mensaje visualizando su figura en la pantalla de visualización. Obviamente, el mecanismo real para visualizar los objetos difiere de una figura a otra, pero todas las figuras realizan esta tarea en respuesta al mismo mensaje.

Otro ejemplo podría ser el siguiente: un usuario de un sistema tiene que imprimir un listado de sus clientes, el puede elegir imprimirlo en el monitor, en la impresora, en un archivo en disco o en una terminal remota. De esta forma el mensaje es imprimir el listado pero la respuesta la dará el objeto que el usuario desee, el monitor, la impresora, el archivo o la terminal remota. Cada uno responderá con métodos (comportamientos) diferentes.

5. Resumen

Como vimos anteriormente, el análisis de un problema es el paso más importante para resolver un problema, existen varias formas de plantear las soluciones, pero hay dos grandes formas que se están utilizando hoy en día, que son el análisis estructurado y el orientado a objetos, el primero se está dejando de usar, sin embargo aún quedan muchos sistemas resueltos bajo esta óptica que hay que mantener o actualizar. Y la segunda forma es la que nosotros vamos a usar ya que es la tendencia del mercado, es la que mejor rendimiento tiene, se está usando desde hace varios años y es la que tiene mayor soporte en cuanto a tecnologías de análisis, diseño y desarrollo de software.

A continuación vamos a realizar una breve comparación entre la programación estructurada y la orientada a objetos que nos servirá para entender mejor como tendremos que resolver los problemas bajo la óptica orientada a objetos.

Programación estructurada

La programación estructurada tiende a ser *orientada a* **la** *acción.* Los programadores se **concentran en escribir funciones**, que son grupos de acciones que ejecutan alguna tarea común y que se agrupan para formar programas. La **unidad fundamental de la programación es la función**. Es cierto que los datos son importantes, pero la óptica es que los datos son materia prima para las acciones que las funciones ejecutan. Semánticamente las acciones son verbos. Los *verbos* en una especificación de sistema ayudan al programador a determinar el conjunto de funciones que juntas funcionarán para ponerlo en marcha.

Programación orientada a objetos

La programación orientada a objetos como su nombre lo indica está *orientada al objeto (concepto).* El **concepto** se implementa en clases (*class*). **La clase es la unidad básica de programación.** Es una unidad que contiene los atributos y todas las funciones necesarias a su tratamiento.

Entonces cada clase contiene datos junto con un conjunto de funciones que manipula dichos datos. Los componentes de datos de una clase se llaman *datos miembros* o *atributos.* El comportamiento de la clase se implementa mediante *funciones miembro.*

El foco de atención está sobre los objetos, en vez de sobre las funciones. Los *sustantivos* en una especificación de sistema ayudan al programador a determinar el conjunto de clases, a partir de las cuales serán creados los objetos que funcionarán conjuntamente para poner en práctica el sistema.

Es muy importante que a la hora de analizar un problema utilicemos sustantivos y no verbos.

A continuación resolveremos un problema bajo la óptica orientada a objetos:

PROBLEMA

Dado el valor de los tres lados de un triángulo, calcular el perímetro.

ANALISIS

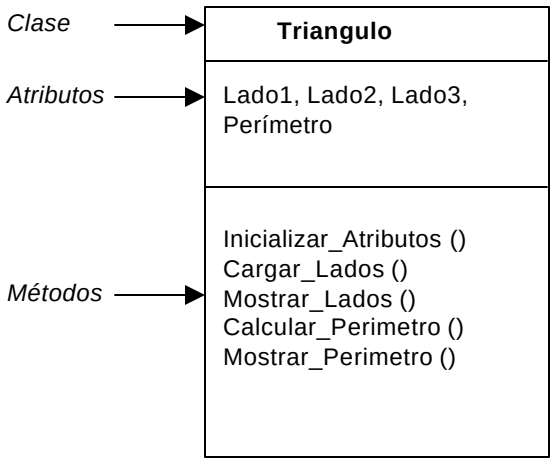
1. Identificar el objeto: identificar la entidad que engloba al problema, en nuestro ejemplo es el triángulo.

2. Identificar sus atributos: podemos decir que todo triángulo tiene como datos esenciales los tres lados, además, nuestro problema en particular necesitará el valor del perímetro, que es un datos calculable a partir de los tres lados y que puede figurar como atributo o no dependiendo si se quiere guardar su estado en el objeto. Como nuestro enunciado no nos restringe la forma de implementarlo vamos a definir como atributos los tres lados y el perímetro.

3. Identificar sus métodos: los métodos a elegir siempre dependerán de las operaciones que podemos realizar con los atributos, entonces podemos enumerar:

Inicializar los atributos: Inicializar en 0, por ejemplo, los lados y el perímetro.
Ingresar los lados: Se ingresa el valor de cada lado.
Mostrar los lados: Mostrar el valor de los lados.
Calcular el perímetro: formula que devuelve el perímetro del triángulo.
Mostrar el perímetro: Mostrar el valor del perímetro.

Graficando el resultado



CLASE PRACTICA

A continuación se sugieren ejercicios prácticos para la primera semana,

1. Dados las siguientes entidades definirlas, identificar la clase, los atributos, los métodos y los mensajes correspondientes.

1. Persona	2. Cliente	3. Proveedor	4. Producto
5. Computadora	6. Mesa	7. Automóvil	8. Avión
9. Cuenta bancaria	10. Comprobante de factura		

11. Definir cualquier entidad de la vida real, que se utilice en la facultad, trabajo, hogar, etc.

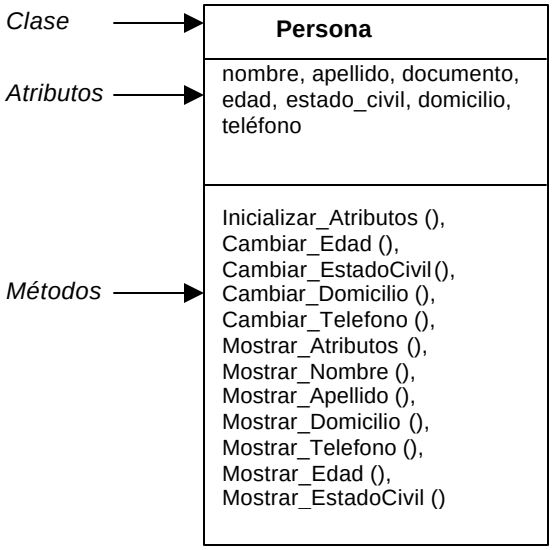
Resoluciones

1.1. Persona

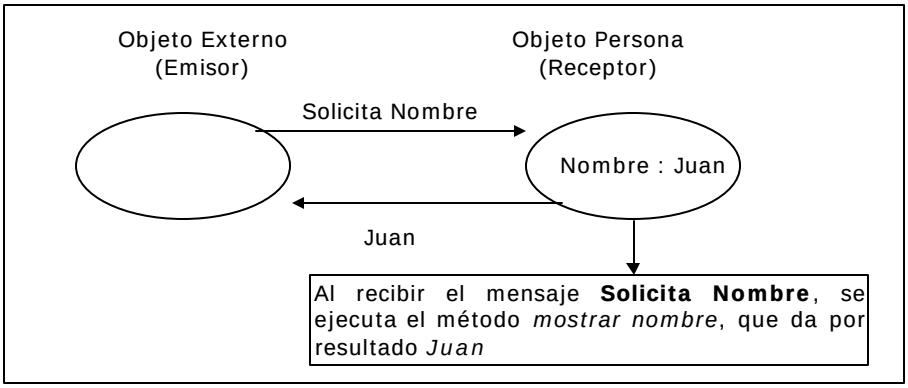
Atributos : nombre, apellido, documento, edad, estado civil, domicilio, teléfono, etc.

Métodos : Inicializar atributos, cambiar edad, cambiar estado civil, cambiar domicilio, cambiar teléfono, Mostrar todos los atributos, Mostrar Nombre, mostrar documento, mostrar domicilio, mostrar teléfono, etc.

Mensajes : 1. Pedir el nombre. 2. Pedir cambio de edad. 3. Pedir el domicilio. Etc.



Ej : 1 Método Pedir el nombre

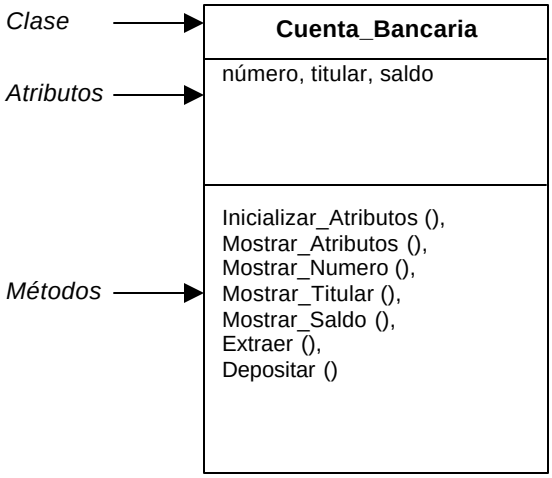


1.9. Cuenta bancaria

Atributos : Numero, Titular, Saldo

Métodos : Inicializar atributos, Mostrar numero, mostrar titular, mostrar saldo, depositar, extraer, etc.

Mensajes : 1. Pedir el numero. 2. Pedir el titular. 3. Solicitar un deposito. Etc.


Clase
Atributos
Métodos
Cuenta_Bancaria
número, titular, saldo
Inicializar_Atributos (),
Mostrar_Atributos (),
Mostrar_Numero (),
Mostrar_Titular (),
Mostrar_Saldo (),
Extraer (),
Depositar ()