Los Paradigmas

Qué es un Paradigma

Un Paradigma es un concepto abstracto, la definición de un modelo o patrón en cualquier disciplina física.
Todo paradigma nace de la filosofía de una comunidad científica que, especializada en un tema particular, decide plantear algún nuevo concepto abstracto o forma de pensar. En este caso en particular, una parte de la comunidad científica que estudiaba la computación como una rama matemática y se especializaba en ello, forja esta nueva ideología o forma de pensar que es el Paradigma de Programación Orientado a Objetos.
En programación, el paradigma, nos determina cómo van a ser los bloques de construcción de los programas, es decir, los elementos que los constituyen.

Tipos de Paradigmas

Imperativo: Los programas se componen de un conjunto de sentencias que cambian su estado. Son secuencias de comandos que ordenan acciones a la computadora.

Declarativo: Opuesto al imperativo. Los programas describen los resultados esperados sin mostrar explícitamente los pasos a llevar a cabo para alcanzarlos.

Lógico: El problema se modela con enunciados de lógica de primer orden.

Funcional: Los programas se componen de funciones, es decir, implementaciones de comportamiento que reciben un conjunto de datos de entrada y devuelven un valor de salida.

Orientado a objetos: El comportamiento del programa es llevado a cabo por objetos, entidades que representan elementos del problema a resolver y tienen atributos y comportamiento.

El Paradigma de Objetos

El paradigma de programación orientada a objetos plantea que todo sistema o proceso informático puede modelarse con Objetos que se encuentren vivos en algún tipo de Ambiente y se relacionan con otros Objetos enviando y recibiendo Mensajes.
Es decir, programar bajo este paradigma, implica que nuestros programas, deberán ser pensados sólo con objetos y mensajes.
Un objeto, en OOP, es un ente completamente abstracto que sabe escuchar, responder y enviar ciertos mensajes.
Un Mensaje es una comunicación dirigida de un objeto a otro. Información enviada desde emisor, hasta el receptor. El paradigma de objetos tiene otros conceptos que completan este esquema de Objeto-Mensaje, que se irán detallando a lo largo del curso.
Como todo concepto teórico o abstracto que quiera ser llevado a la vida real o a la práctica, requiere de algún tipo de

implementación.
Si queremos desarrollar un software, con la idea teórica o paradigma únicamente no nos alcanzaría, es necesario tener algo concreto con lo cual trabajar, una implementación del paradigma.
Un paradigma, en tanto que es un concepto teórico, tiene infinitas implementaciones. A la vez, por este mismo motivo, una implementación es una y sólo una instancia posible que responde al Paradigma.
Cuando hablamos de una "implementación" estamos hablando de un lenguaje de programación orientado a objetos (es decir, que contempla los conceptos teóricos del paradigma orientado a objetos). Cada uno de estos lenguajes implementará cada uno de estos conceptos de forma diferente. Por eso es importante diferenciar los "conceptos del paradigma" de la "implementación del paradigma". Es decir, por ejemplo, primero es necesario conocer la definición de un "objeto" según el paradigma, y luego, conocer la implementación de un objeto para determinado lenguaje de programación.
Vamos a ir clarificando cada uno de estos conceptos a lo largo de las clases.

La Programación Orientada a Objetos

OOP (Object Oriented Programming o en español Programación orientada a objetos en inglés) se basa en el concepto de agrupar código y datos juntos en una misma unidad llamada clase. Este proceso es usualmente referido como encapsulamiento o información oculta.
Estas clases son esencialmente la representación de un set de funciones (llamadas métodos) y variables (llamadas propiedades) designadas para trabajar juntas y proveer una interfaz específica.
Es importante entender que las clases son una estructura que no puede ser utilizada directamente, estas deben ser instanciadas en objetos, los cuales podrán así interactuar con el resto de las aplicaciones.
Es posible pensar en clases como un plano para crear un auto, mientras el objeto es de hecho el auto en si mismo, ya que se puede crear en una línea de producción. Podríamos decir que una clase es una fábrica de objetos.

Por qué objetos

La programación orientada a objetos tiende a ser una simple idea. Esta simple idea es: en lugar de crear estructuras de datos por un lado y código por el otro, la idea es asociar piezas de código junto con los datos.
El mercado de lenguajes de programación, tiende a utilizar este paradigma ya que trae innumerables ventajas con respecto a la programación estructurada.
A continuación se enumeran algunas de ellas:

- **Escalabilidad:** Capacidad de crecimiento. Que un sistema sea escalable, significa que está preparado para poder crecer sin limitaciones ni prejuicios. En casos de desarrollo corporativos como por ejemplo el de un sistema bancario, donde constantemente se necesitan agregar nuevos módulos y cambios en su comportamiento, es necesario contar con lenguaje de programación que permita fácilmente administrar estos tipos de desarrollos empresariales.
- **Mantenimiento:** Una de las principales características del paradigma de objetos, es su capacidad realizar cambios en el programa a muy bajo costo (esfuerzo del programador). Gracias a una buena organización de las clases y sus herencias es posible hacer pequeños pero importantes cambios que impacten luego en todas las partes del sistema.
- **Seguridad:** Este paradigma posee lo que se denomina como encapsulamiento y visibilidad lo cual permite establecer diferentes niveles de seguridad dentro de un equipo de desarrolladores, organizando el trabajo en equipo y limitando la posible existencia de errores o incongruencias cometidos por los programadores.
- **Reutilización:** A través de la correcta utilización de jerarquía de clases, y la redefinición de funciones –

polimorfismo, se puede reutilizar un importante porcentaje de código, evitando escribir subprogramas parecidos dentro de un mismo desarrollo.

- **Simplicidad:** Cuando el código de un programa desarrollado con lenguaje estructurado, crece indefinidamente, se llega a un punto donde el mismo se vuelve confuso y engorroso. El paradigma de objetos provee simplicidad al poder organizar la complejidad del mismo a través de su estructura de clases y objetos.

Características de la OOP

Las características siguientes son las más importantes:

Abstracción

Denota las características esenciales de un objeto, donde se capturan sus comportamientos. Cada objeto en el sistema sirve como modelo de un "agente" abstracto que puede realizar trabajo, informar y cambiar su estado, y "comunicarse" con otros objetos en el sistema sin revelar cómo se implementan estas características. Los procesos, las funciones o los métodos pueden también ser abstraídos y cuando lo están, una variedad de técnicas son requeridas para ampliar una abstracción. El proceso de abstracción permite seleccionar las características relevantes dentro de un conjunto e identificar comportamientos comunes para definir nuevos tipos de entidades en el mundo real. La abstracción es clave en el proceso de análisis y diseño orientado a objetos, ya que mediante ella podemos llegar a armar un conjunto de clases que permitan modelar la realidad o el problema que se quiere atacar.

Encapsulamiento

Significa reunir a todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción. Esto permite aumentar la cohesión de los componentes del sistema. Algunos autores confunden este concepto con el principio de ocultación, principalmente porque se suelen emplear conjuntamente.
Principio de ocultación

Cada objeto está aislado del exterior, es un módulo natural, y cada tipo de objeto expone una interfaz a otros objetos que especifica cómo pueden interactuar con los objetos de la clase. El aislamiento protege a las propiedades de un objeto contra su modificación por quien no tenga derecho a acceder a ellas, solamente los propios métodos internos del objeto pueden acceder a su estado. Esto asegura que otros objetos no pueden cambiar el estado interno de un objeto de maneras inesperadas, eliminando efectos secundarios e interacciones inesperadas. Algunos lenguajes relajan esto, permitiendo un acceso directo a los datos internos del objeto de una manera controlada y limitando el grado de abstracción. La aplicación entera se reduce a un agregado o rompecabezas de objetos.

Polimorfismo

Comportamientos diferentes, asociados a objetos distintos, pueden compartir el mismo nombre, al llamarlos por ese nombre se utilizará el comportamiento correspondiente al objeto que se esté usando. O dicho de otro modo, las referencias y las colecciones de objetos pueden contener objetos de diferentes tipos, y la invocación de un comportamiento en una referencia producirá el comportamiento correcto para el tipo real del objeto referenciado. Cuando esto ocurre en "tiempo de ejecución", esta última característica se llama asignación tardía o asignación dinámica.

Herencia

Las clases no están aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen. La herencia organiza y facilita el polimorfismo y el encapsulamiento permitiendo a los objetos ser definidos y creados como tipos especializados de objetos preexistentes. Estos pueden compartir (y extender) su comportamiento sin tener que volver a implementarlo. Esto suele hacerse habitualmente agrupando los objetos en clases y estas en árboles o enrejados que reflejan un comportamiento común. Cuando un objeto hereda de más de una clase se dice que hay herencia múltiple.

UML

El Lenguaje Unificado de Modelado (UML) fue creado para forjar un lenguaje de modelado visual común y semántica y sintácticamente rico para la arquitectura, el diseño y la implementación de sistemas de software complejos, tanto en estructura como en comportamiento.

Es comparable a los planos usados en otros campos y consiste en diferentes tipos de diagramas. En general, los diagramas UML describen los límites, la estructura y el comportamiento del sistema y los objetos que contiene.

UML no es un lenguaje de programación, pero existen herramientas que se pueden usar para generar código en diversos lenguajes usando los diagramas UML. UML guarda una relación directa con el análisis y el diseño orientados a objetos.

Motivación

Durante años, los programadores se han dedicado a construir aplicaciones muy parecidas que resolvían una y otra vez los mismos problemas. Para conseguir que los esfuerzos de los programadores puedan ser utilizados por otros programadores se creó la OOP. Que es una serie de normas de realizar las cosas de manera que otros programadores puedan utilizarlas y adelantar su trabajo, de manera que consigamos que el código se pueda reutilizar.

Cómo se piensa en objetos

Pensar en términos de objetos es muy parecido a cómo lo haríamos en la vida real. Por ejemplo vamos a pensar en un auto. Diríamos que el auto es el elemento principal que tiene una serie de características, como podrían ser el color, el modelo o la marca. Además tiene una serie de funcionalidades asociadas, como pueden ser ponerse en marcha, parar o detenerse.

Pues en un esquema OOP el auto sería el objeto, las propiedades serían las características como el color o el modelo y los métodos serían las funcionalidades asociadas como ponerse en marcha o parar.

Estos objetos se podrán utilizar en los programas, por ejemplo en un programa de concesionarios. Los programas Orientados a objetos utilizan muchos objetos para realizar las acciones que se desean realizar y ellos mismos también son objetos. Es decir, el taller de autos será un objeto que utilizará objetos auto, herramientas, mecánicos, etc.

Los objetos son entidades que combinan estado (atributo), comportamiento (método) e identidad.

Estados

Está compuesto de datos, será uno o varios atributos a los que se habrán asignado unos valores concretos (datos).

Comportamiento

Está definido por los procedimientos o métodos con que puede operar dicho objeto, es decir, qué operaciones se pueden realizar con él.

Identidad

Es una propiedad de un objeto que lo diferencia del resto, dicho con otras palabras, es su identificador .

Un objeto contiene toda la información que permite definirlo e identificarlo frente a otros objetos pertenecientes a otras clases e incluso frente a objetos de una misma clase, al poder tener valores bien diferenciados en sus atributos. A su vez, los objetos disponen de mecanismos de interacción llamados métodos, que favorecen la comunicación entre ellos. Esta comunicación favorece a su vez el cambio de estado en los propios objetos. Esta característica lleva a tratarlos como unidades indivisibles, en las que no se separa el estado y el comportamiento.
Los métodos (comportamiento) y atributos (estado) están estrechamente relacionados por la propiedad de conjunto. Esta propiedad destaca que una clase requiere de métodos para poder tratar los atributos con los que cuenta. El programador debe pensar indistintamente en ambos conceptos, sin separar ni darle mayor importancia a alguno de ellos. Hacerlo podría producir el hábito erróneo de crear clases contenedoras de información por un lado y clases con métodos que manejen a las primeras por el otro. De esta manera se estaría realizando una programación estructurada camuflada en un lenguaje de programación orientado a objetos.
La OOP difiere de la programación estructurada tradicional, en la que los datos y los procedimientos están separados y sin relación, ya que lo único que se busca es el procesamiento de unos datos de entrada para obtener otros de salida. La programación estructurada anima al programador a pensar sobre todo en términos de procedimientos o funciones, y en segundo lugar en las estructuras de datos que esos procedimientos manejan. En la programación estructurada sólo se escriben funciones que procesan datos. Los programadores que emplean OOP, en cambio, primero definen objetos para luego enviarles mensajes solicitandoles que realicen sus métodos por sí mismos.

Origen

Los conceptos de la programación orientada a objetos tienen origen en Simula 67, un lenguaje diseñado para hacer simulaciones, creado por Ole-Johan Dahl y Kristen Nygaard del Centro de Cómputo Noruego en Oslo. En este centro, se trabajaba en simulaciones de naves, que fueron confundidas por la explosión combinatoria de cómo las diversas cualidades de diferentes naves podían afectar unas a las otras. La idea ocurrió para agrupar los diversos tipos de naves en diversas clases de objetos, siendo responsable cada clase de objetos de definir sus propios datos y comportamientos. Fueron refinados más tarde en Smalltalk, que fue desarrollado en Simula en Xerox PARC pero diseñado para ser un sistema completamente dinámico en el cual los objetos se podrían crear y modificar "en marcha" en lugar de tener un sistema basado en programas estáticos.
La programación orientada a objetos tomó posición como el estilo de programación dominante a mediados de los años ochenta, en gran parte debido a la influencia de C++, una extensión del lenguaje de programación C. Su dominación fue consolidada gracias al auge de las Interfaces gráficas de usuario, para las cuales la programación orientada a objetos está particularmente bien adaptada.

Modelado de Objetos

Qué es un Modelo

Un modelo es una posibilidad de visualizar a escala o de una manera simulada algo que será construido en la realidad. Es posible decir que antes de construir un edificio se realizan maquetas a escala que representa modelos a seguir, así como también cuando se construye un auto se confeccionan distintos planos o modelos a escala para intentar simular o prever su comportamiento.
Académicamente, es posible definirla como una abstracción de la realidad, es una simplificación de la realidad, con el objetivo final de pasar del modelo a producto real.
Los modelos pueden ser expresados en distintos niveles de precisión, desde algo muy genérico presentando una visión, hasta algo mucho más específico que representa un gran compromiso con el producto a construir.

Qué es un Objeto

Un objeto en OOP es un ente completamente Abstracto que sabe escuchar, responder, y enviar ciertos mensajes.
Bajando más a informática, podemos definir un objeto como un ente computacional, que está dentro del software, y puede representar o no un ente de la realidad que está fuera de la computadora.
Por ejemplo, un objeto Persona, o un Usuario, puede representar una persona real, en el manejo de usuarios de una aplicación.
Si seguimos con el ejemplo de Persona, una persona puede comer, vestirse, trasladarse, leer, y muchas cosas más! Si revisamos la primera definición, dice que sabe "responder ciertos mensajes" entonces, estos son, los mensajes de los que hablábamos.

¿Y quién le envía los mensajes?

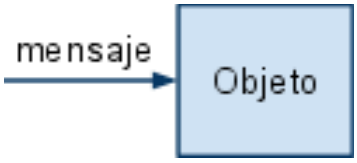
La respuesta es fácil, otro objetos, recordemos, que en el paradigma de objetos, sólo tenemos objetos y mensajes, y nada más.

¿Y qué pasa con los números? ¿También son objetos?

Va a depender del lenguaje. En principio, en un lenguaje orientado a objetos, todo debería ser un objeto, pero esto no es tan así, existen algunos lenguajes que eligen no respetar eso a rajatabla y modelan algunos conceptos muy básicos como tipos primitivos, como por ejemplo los número, los booleanos, los caracteres.

Qué es un Mensaje

En una aplicación orientada a objetos, entonces, tenemos objetos que se piden cosas entre sí. Por lo que, un mensaje, es una comunicación entre dos objetos, emisor y receptor, donde el emisor le pide algo al receptor, y puede (o no) obtener algo como respuesta.



Por ejemplo, si a un objeto Persona le pido que me diga su edad, voy a recibir como respuesta un número que representa la edad de la persona.
Si tengo un objeto Alumno, podría pedirle los cursos que está haciendo actualmente, ¿y en este caso que me devuelve? Me devuelve un objeto que representa el conjunto de cursos que está haciendo.

¿Un objeto entiende cualquier mensaje que se le envía?
No! Nosotros, como programadores de objetos, debemos definir cuales son los mensajes que entiende cada objeto.
¿Y qué pasa si le mando un mensaje que no entiende?
Esto dependerá del lenguaje, pero en principio, esto causa un error, ya que si un objeto no entiendo el mensaje que se le está mandando, entonces nuestro programa no estaría andando bien.

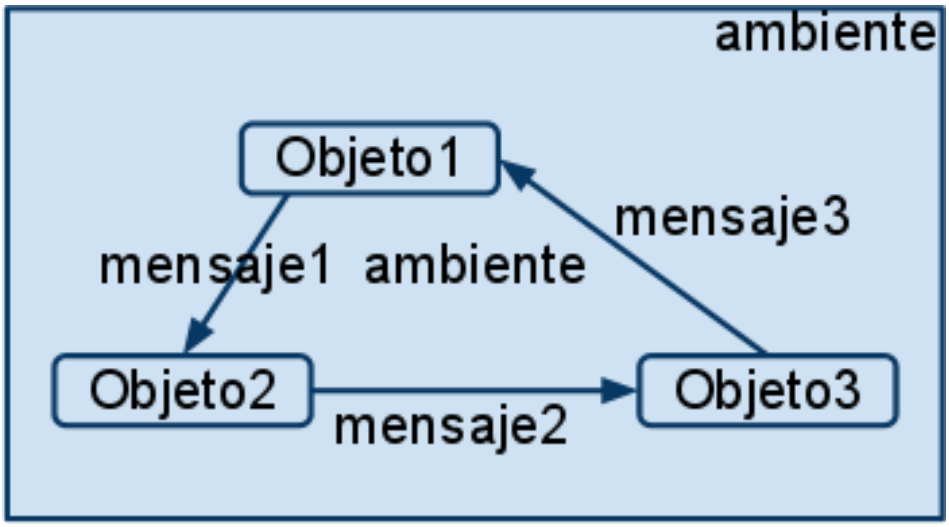
Qué es el Comportamiento

Ya dijimos que un objetos era un ente abstracto, que sabía enviar y responder mensajes. Y que nosotros debemos definir cuales son los mensajes que un objeto debe entender.
El comportamiento, es el conjunto de mensajes que un objeto entiende, y a los que me puede o no responder. Lo importante, es que lo entiende.
Con este nuevo concepto, podríamos entonces entender un objeto como:
"Un ente computacional que exhibe comportamiento".
De esta forma, dado un objeto, por ejemplo Persona, uno podría preguntar, ¿cuál es el comportamiento de una Persona? Y ahí uno debe responder con todo lo que el objeto persona sabe hacer. Esto, lo define el programador de ese objeto.
Dicho de otra manera, el "comportamiento" son las cosas que sabe hacer un objeto, el conjunto de mensajes que un objeto le va a poder enviar.

Qué es un Ambiente

Los objetos nacen y mueren, o sea "viven". En general nacen porque alguien los creó ¿quién es ese alguien? un objeto! como todo... y se mueren porque alguien los destruye, esto también dependerá del lenguaje, pero vamos a hablar de esto más adelante.

Los objetos viven “dentro de la computadora”, en algún lugar determinado, que es el responsable de contenerlos, manejar su ciclo de vida, dar el contexto para la creación y el envío de mensajes.



El Software Orientado a Objetos

Con estos conceptos, podemos decir que es el software. Es el resultado de la interacción de un conjunto de objetos que viven en un ambiente y que se comunican entre sí enviándole mensajes.
Es importante definir software desde el punto de vista de la programación orientada a objetos, ya que esta es la forma que voy a tener de pensar en una programa, en la construcción de un programa.
Entonces, de ahora en más, para pensar un programa, tenemos que pensar:

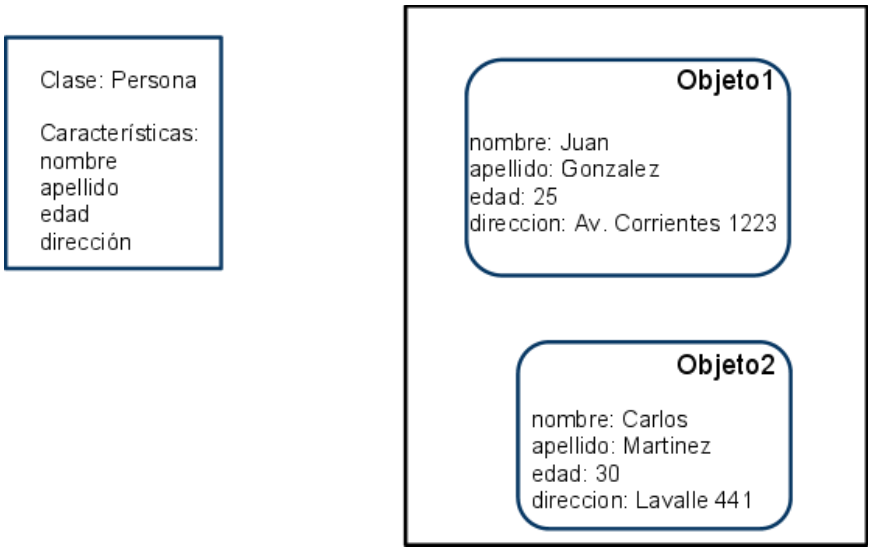
- Qué objetos vamos a incluir en la aplicación
- Qué mensajes va a entender cada objeto
- Cómo van a interactuar los objetos entre sí

Introducción al Paradigma de Objetos

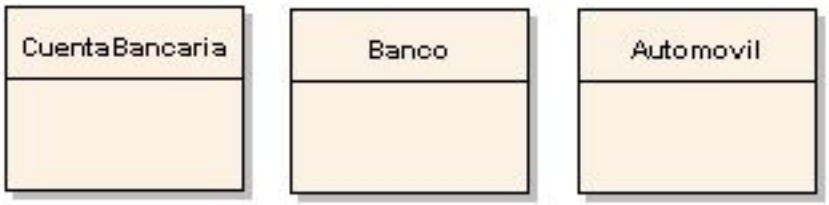
Los Objetos y las Clases

En este paradigma todo se lleva a cabo con “clases” y “objetos”. ¿Qué es un objeto? Un objeto es un conjunto de atributos (estado) y operaciones (comportamiento).
Debemos entender los objetos como abstracciones de la realidad con ciertas características representativas. Estos objetos representan (o deberían representar) ideas del mundo real, ideas concretas como por ejemplo animales o personas.
En el POO (Paradigma Orientado a Objetos) existen dos conceptos claves que deben ser comprendidos, los conceptos de clase y objeto.
Las clases son plantillas que contienen atributos y operaciones, se consideran los moldes para los objetos. Las clases se detectan como sustantivos en singular el 99% de las veces, esto significa que suelen tener nombres como “CuentaCorriente” o “Automovil”.

Ahora bién, ¿qué es un objeto exactamente? Un objeto es la instancia de una clase, podemos decir que el objeto representa algo en particular y la clase algo en general. Por ejemplo si tuviéramos la clase "Automovil", podríamos crear las instancias (objetos) llamados "Renault", "Volkswagen" o "Ford".



Si tenemos una clase "CuentaBancaria", podemos tener infinita cantidad de objetos de "CuentaBancaria", pero cada una tendrá sus valores individuales, aunque todas pertenezcan a esta clase.
Decimos que un objeto pertenece a una clase, cuando es una instanciación de esa clase; es decir, al crear un objeto de la clase "Banco" y asignarle el nombre "bancoNacion", el objeto "bancoNacion" es una instanciación de "Banco". Continuando con este ejemplo, podríamos crear más instanciaciones de "Banco" y asignarles nombres como "bancoRio" o "bancoCityBank", y todos ellos tendrían algo en común, que pertenecen a la clase "Banco", a pesar de que cada objeto tenga sus distintos valores, como por ejemplo, distinta cantidad de clientes, de sucursales, etc (representados por los atributos).
A continuación se muestran algunas clases representadas gráficamente:



Las clases están compuestas por atributos y operaciones, y serán presentadas a continuación.

Los Atributos

Como mencionamos anteriormente, los atributos son las variables contenidas por los objetos. Las clases definen los atributos y los objetos los "completan". Las variables de una clase definen las características de sus objetos, por ejemplo:

En este caso creamos una clase "Automovil" y colocamos atributos color, modelo, precio, y usado que al adquirir valores en un objeto, lo diferenciarán a este de otras instancias (objetos) de la clase "Automovil".
Lo mismo podríamos hacer con una clase "CuentaBancaria", como se muestra a continuación:

En este caso la clase "CuentaBancaria" cuenta con un atributo "saldo".
En resumen, los atributos son las características de una clase.

Los Tipos de Dato

Podemos encontrar la definición de atributo, mediante la cual definimos que un objeto tiene un conjunto de "características" que lo describe, y que se definen en una clase.
Por ejemplo, el atributo nombre, va ser una palabra, es decir, va a ser un String, de la misma manera, la edad, es un número, es decir, va a ser un "int".
"int" y "String" son tipos. Un tipo es la forma de describir o almacenar un dato.
Cuando decimos que tenemos un dato del tipo "int", estamos diciendo que podría ser cualquier número de tipo entero.
De la misma manera, un objeto Persona podría tener un atributo "Trabajo", donde trabajo es también un objeto con sus propias características, es decir, el trabajo podría tener un empleador, un nombre, una ubicación, etc. De esta manera, estamos diciendo que Persona, tiene como atributo "Trabajo", que será de qué tipo? De tipo trabajo!
Acá empezamos a ver la potencia de los objetos.
Anteriormente, nosotros trabajamos con un conjuntos acotado de "tipos" para las variables (int, String, boolean, etc), ahora, con los objetos, nosotros podemos crear nuestros propios "tipos". Cualquier clase que nosotros creamos, de ahora en más, eso va a ser un nuevo tipo con el que podemos trabajar.
De la misma manera si creamos un clase "Banco", de ahora en más vamos a tener objetos que son instancias de la clase Banco, o también podemos decir que tenemos objetos del tipo Banco.

Las Operaciones

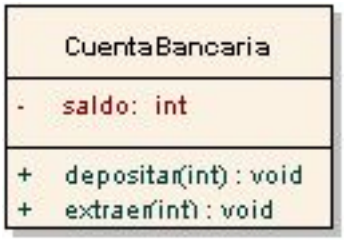
Las operaciones son acciones contenidas por el objeto que definen su comportamiento.

En este caso hemos agregado dos operaciones a la clase "Automovil". Estas operaciones le permiten acelerar o frenar, como indican sus respectivos nombres (cuanto más autoexplicativos sean los nombres, mejor).

Las operaciones son, en el 99% de los casos, verbos.

Las operaciones pueden admitir parámetros. Los parámetros son variables que utilizará la operación para generar cierto comportamiento. Las operaciones acelerar y frenar tienen como parámetros dos valores "int" (números enteros) que representan la velocidad y el tiempo que acelerará o frenará.

Las operaciones de una clase "CuentaBancaria" podrían ser las siguientes:



Las operaciones "depositar" y "extraer" tienen un solo parámetro, este parámetro representa el monto a extraer o depositar.

En resumen, las operaciones son acciones contenidas por un objeto que definen su comportamiento.