

Hito Grupal

Campus FP



REalizado por: Sebastian Gonzalez, adrian tauste y mario valverde

**Historia de la Programación Orientada a Objetos**

La historia de la POO comienza en el Centro de Cálculo Noruego en Oslo, un equipo de investigación, liderado por Krinsten Nygaard, ayudado por Ole-Johan Dahl, estos tuvieron problemas en el desarrollo de simulaciones de sistemas físicos, como el caso de simular el funcionamiento y rendimiento de un motor. Se dieron dos problemas en este punto:

1. Los programas eran muy complejos.
2. Las modificaciones, al ser una simulación, eran muy frecuentes

La solución que idearon fue diseñar el programa paralelamente al objeto físico. Es decir, si el objeto físico tenía cien componentes, el programa también tendría cien módulos, uno por cada pieza. Partiendo el programa de esta manera, había una total correspondencia entre el sistema físico y el sistema informático. Así, cada pieza física tenía su abstracción informática en un módulo. De la misma manera que los sistemas físicos se comunican enviando señales, los módulos informáticos se comunican enviándole mensajes.

Además de resolverse las dificultades anteriores. Se obtuvo otro beneficio: reusabilidad. Cuando se construye un programa, se obtienen piezas para futuros programas.

A partir de esta situación, se desarrolló un lenguaje de programación con algunos conceptos pilares de la Programación Orientada a Objetos. Este se denominó SIMULA 67, y fue presentado en 1967 al público. En este lenguaje aparecen por primera vez los conceptos de clases y objetos.

Cronología de algunos lenguajes de Programación Orientada a Objetos:

* 1960: Simula 67
* 1970: Smalltalk
* 1980: C++
* 1990: JAVA, Ruby, Python
* 2000: C#, PHP 5
* 2010: Go

**Programación**

La Programación es el proceso al que se recurre para crear algún tipo de aplicación o software, para materializar un concepto o proyecto que requiere de la utilización de un lenguaje informático para poder llevarse a cabo. Hay muchos tipos de programación, dependiendo del lenguaje a utilizar.

**Diferentes tipos de entornos que hemos usado**

**PseInt**

PSeInt es una herramienta para asistir a un estudiante en sus primeros pasos en programación. Mediante un simple e intuitivo pseudolenguaje en español (complementado con un editor de diagramas de flujo), le permite centrar su atención en los conceptos fundamentales de la algoritmia computacional, minimizando las dificultades propias de un lenguaje y proporcionando un entorno de trabajo con numerosas ayudas y recursos didácticos. Entre las funcionalidades más destacables de PSeInt se pueden mencionar aspectos como las funciones de autocompletado, el coloreado de sintaxis, los listados de funciones, operadores y variables, o la posibilidad de convertir determinados algoritmos de pseudocódigo a código equivalente en varios lenguajes de programación.

**Eclipse**

Eclipse es un entorno de desarrollo integrado, de Código abierto y Multiplataforma. Mayoritariamente se utiliza para desarrollar lo que se conoce como Aplicaciones de Cliente Enriquecido, entorno de desarrollo integrado, opuesto a las aplicaciones Cliente-liviano, entorno de desarrollo integrado basadas en navegadores. Es una potente y completa plataforma de Programación, desarrollo y compilación de elementos tan variados como sitios web, programas en C + + o aplicaciones Java.

Gestión de proyectos: el desarrollo sobre Eclipse se basa en los proyectos, que son el conjunto de recursos relacionados entre sí, como puede ser el código fuente, documentación, ficheros configuración, árbol de directorios.

Depurador de código: se incluye un potente depurador, de uso fácil e intuitivo, y que visualmente nos ayuda a mejorar nuestro código. Para ello sólo debemos ejecutar el programa en modo depuración (con un simple botón).

**Fundamentos de la programación**

**-Algoritmos:**

Un algoritmo informático es un conjunto de instrucciones definidas, ordenadas y acotadas para resolver un problema o realizar una tarea.

En programación, supone el paso previo a ponerse a escribir el código. Primero debemos encontrar la solución al problema (definir el algoritmo informático), para luego, a través del código, poder indicarle a la máquina qué acciones queremos que lleve a cabo. De este modo, un programa informático no sería más que un conjunto de algoritmos ordenados y codificados en un lenguaje de programación para poder ser ejecutados en un ordenador.

**-Variables y constantes:**

Una constante es un dato cuyo valor no puede cambiar durante la ejecución del programa. Recibe un valor en el momento de la compilación y este permanece inalterado durante todo el programa.

Una variable es un nombre asociado a un elemento de datos que está situado en posiciones contiguas de la memoria principal, y su valor puede cambiar durante la ejecución de un programa.

**-Tipos de datos:**

En programación, un tipo de dato informático o simplemente tipo es un atributo de los datos que indica al ordenador (y/o al programador) sobre la clase de datos que se va a trabajar. Esto incluye imponer restricciones en los datos, como qué valores pueden tomar y qué operaciones se pueden realizar.

Los tipos de datos comunes son: números enteros, números con signo (negativos), números de coma flotante (decimales), cadenas alfanuméricas, estados (booleano), etc.

-**Operadores:**

Los operadores aritméticos son aquellos que "manipulan" datos numéricos, tanto enteros como reales. Hay 2 tipos de operadores aritméticos: unarios y binarios. Los operadores unarios se anteponen a la expresión aritmética, y son los operadores de signo. Los operadores binarios se sitúan entre 2 expresiones aritméticas.

**-Estructuras de control:**

Permiten modificar el flujo de ejecución de las instrucciones de un programa.

Con las estructuras de control se puede:

**-If**: Un if en programación se utiliza para evaluar una expresión condicional: si se cumple la condición (es verdadera), ejecutará un bloque de código. Si es falsa, es posible ejecutar otras sentencias. Como hemos dicho, la condición es una condición lógica, sólo devolverá true o false, y se ejecutará si su valor es true.

**-For**: Un for en programación se usa cuando queremos repetir un conjunto de instrucciones un número finito de veces.

**-While**: Los ciclos while son también una estructura cíclica, que nos permite ejecutar una o varias líneas de código de manera repetitiva sin necesidad de tener un valor inicial e incluso a veces sin siquiera conocer cuando se va a dar el valor final que esperamos, los ciclos while, no dependen directamente de valores numéricos.

**-Subprocesos**: Es la unidad básica a la que el sistema operativo asigna tiempo de procesador. Cada subproceso tiene una prioridad de programación y mantiene un conjunto de estructuras que el sistema usa para guardar el contexto del subproceso mientras la ejecución del subproceso está pausada.

**-Cadenas**: Es una secuencia ordenada de elementos que pertenecen a un cierto lenguaje formal o alfabeto análogas a una fórmula o a una oración. En general, una cadena de caracteres es una sucesión de caracteres.

- **(Do-While):** Ejecutar un grupo de sentencias solo cuando se cumpla una condición.

**-(while-do):** Ejecutar un grupo de sentencias hasta que se cumpla una condición.

**-Arreglos:** Un arreglo (vector, array, matriz) es un conjunto de datos o una estructura de datos homogéneos que se encuentran ubicados en forma consecutiva en la memoria RAM (sirve para almacenar datos en forma temporal).

Tipos:

* De una dimensión.
* De dos dimensiones.
* De tres o más dimensiones.

**Arreglos unidimensionales**

Es un tipo de datos estructurado que está formado de una colección finita y ordenada de datos del mismo tipo. Es la estructura natural para modelar listas de elementos iguales. Están formados por un conjunto de elementos de un mismo tipo de datos que se almacenan bajo un mismo nombre, y se diferencian por la posición que tiene cada elemento dentro del arreglo de datos. Al declarar un arreglo, se debe inicializar sus elementos antes de utilizarlos. Para declarar un arreglo tiene que indicar su tipo, un nombre único y la cantidad de elementos que va a contener.

**Arreglos multidimensionales**

Es un tipo de dato estructurado, que está compuesto por dimensiones. Para hacer referencia a cada componente del arreglo es necesario utilizar n índices, uno para cada dimensión. El término dimensión representa el número de índices utilizados para referirse a un elemento particular en el arreglo. Los arreglos de más de una dimensión se llaman arreglos multidimensionales.

**Qué es la programación orientada a objetos**

La Programación Orientada a Objetos (POO) es un paradigma de programación, es decir, un modelo o un estilo de programación que nos da unas guías sobre cómo trabajar con él. Se basa en el concepto de clases y objetos. Este tipo de programación se utiliza para estructurar un programa de software en piezas simples y reutilizables.

Con el paradigma de Programación Orientado a Objetos lo que buscamos es dejar de centrarnos en la lógica pura de los programas, para empezar a pensar en objetos, lo que constituye la base de este paradigma. Esto nos ayuda muchísimo en sistemas grandes, ya que, en vez de pensar en funciones, pensamos en las relaciones o interacciones de los diferentes componentes del sistema.

**¿Cómo se crean los programas orientados a objetos?**

Resumiendo, consistiría en hacer clases y crear objetos a partir de estas clases. Las clases forman el modelo a partir del que se estructuran los datos y los comportamientos.

Una clase es una plantilla. Define de manera genérica cómo van a ser los objetos de un determinado tipo. Por ejemplo, una clase para representar a animales puede llamarse ‘animal’ y tener una serie de atributos, como ‘nombre’ o ‘edad’. Y una serie con los comportamientos que estos pueden tener, como caminar o comer, y que a su vez se implementan como métodos de la clase.

Un objeto es un ente creado que posee atributos y comportamientos con los cuales una persona puede interactuar. Siguiendo con el ejemplo podemos crear un objeto perro llamado roky y un gato llamado garfield que están definidos por la clase animal,es decir, tienen un nombre y una edad, pero que a la hora de llamar a uno u otro puede que realicen acciones diferentes.

**Principios de la programación orientada a objetos**

**La encapsulación**

La encapsulación contiene toda la información importante de un objeto dentro del mismo y solo expone la información seleccionada al mundo exterior.

Esta propiedad permite asegurar que la información de un objeto esté oculta para el mundo exterior, agrupando en una Clase las características o atributos que cuentan con un acceso privado, y los comportamientos o métodos que presentan un acceso público.

La encapsulación de cada objeto es responsable de su propia información y de su propio estado. La única forma en la que este se puede modificar es mediante los propios métodos del objeto. Por lo tanto, los atributos internos de un objeto deberían ser inaccesibles desde fuera, pudiéndose modificar sólo llamando a las funciones correspondientes. Con esto conseguimos mantener el estado a salvo de usos indebidos o que puedan resultar inesperados.

En el ejemplo creado tenemos las propiedades de la clase animal con un modificador de tipo privado que no nos permite acceder a su información de manera directa.

Para ello se usan los métodos get y set, para establecer de manera indirecta la información y posteriormente obtenerla.

**La abstracción**

La abstracción es cuando el usuario interactúa solo con los atributos y métodos seleccionados de un objeto, utilizando herramientas simplificadas de alto nivel para acceder a un objeto complejo.

En la programación orientada a objetos, los programas suelen ser muy grandes y los objetos se comunican mucho entre sí. El concepto de abstracción facilita el mantenimiento de un código de gran tamaño, donde a lo largo del tiempo pueden surgir diferentes cambios.

**La herencia**

La herencia define relaciones jerárquicas entre clases, de forma que atributos y métodos comunes puedan ser reutilizados. Las clases principales extienden atributos y comportamientos a las clases secundarias. A través de la definición en una clase de los atributos y comportamientos básicos, se pueden crear clases secundarias, ampliando así la funcionalidad de la clase principal y agregando atributos y comportamientos adicionales.

Volviendo al ejemplo de los animales, se puede usar una sola clase de animal y agregar un atributo de tipo de animal que especifique el tipo de animal. Los diferentes tipos de animales necesitarán diferentes métodos, por ejemplo, las aves deben poder poner huevos y los peces, nadan. Incluso cuando los animales tienen un método en común, como moverse, la implementación necesitaría muchas declaraciones «si» para garantizar el comportamiento de movimiento correcto. Por ejemplo, las ranas saltan, mientras que las serpientes se deslizan. El principio de herencia nos permite solucionar este problema.

En nuestro ejemplo tenemos la clase perro y una clase gato que heredan las propiedades y atributos de la clase animal, y se añaden métodos específicos de cada animal.

**Polimorfismo**

El polimorfismo consiste en diseñar objetos para compartir comportamientos lo que permite que el mismo método ejecute diferentes comportamientos de dos formas: anulación de método y sobrecarga de método.

En nuestro ejemplo al llamar al método comer desde el objeto roky, nos muestra un mensaje por consola distinto al que se muestra cuando se llama desde el objeto garfield.

**Beneficios de Programación Orientada a Objetos**

* Reutilización del código.
* Convierte cosas complejas en estructuras simples reproducibles.
* Evita la duplicación de código.
* Permite trabajar en equipo gracias al encapsulamiento ya que minimiza la posibilidad de duplicar funciones cuando varias personas trabajan sobre un mismo objeto al mismo tiempo.
* Al estar la clase bien estructurada permite la corrección de errores en varios lugares del código.
* Protege la información a través de la encapsulación, ya que solo se puede acceder a los datos del objeto a través de propiedades y métodos privados.
* La abstracción nos permite construir sistemas más complejos y de una forma más sencilla y organizada.

**Conclusión**

La Programación Orientada a Objetos es actualmente el paradigma que más se utiliza para diseñar aplicaciones y programas informáticos. Son muchas sus ventajas, principalmente cuando necesitas resolver desafíos de programación complejos. Permite una mejor estructura de datos y reutilización del código, lo que facilita el ahorro de tiempo a largo plazo. Eso sí, para ello se requiere pensar bien en la estructura del programa, planificar al comienzo de la codificación, así como analizar los requisitos en clases simples y reutilizables que se pueden usar para diseñar instancias de objetos.