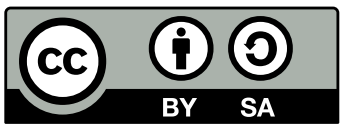
Programación Orientada a Objetos

Evolución histórica y principios fundamentales

José Fidel Argudo Argudo
Francisco Palomo Lozano
Inmaculada Medina Bulo
Gerardo Aburruzaga García

Versión 1.0



Índice

Historia

- La programación orientada a objetos (POO)
  - Puede verse como una evolución de la programación estructurada
  - Pero también como un nuevo paradigma de la programación
  - Actualmente es el que tiene mayor auge
- Cronología
  - Los 60: Simula 67, aparece el concepto de objeto
  - Los 70: se desarrolla SmallTalk en Xerox PARC
  - Los 80: se desarrolla C++ en los laboratorios Bell de AT&T
  - Los 90: se desarrolla Java en Sun Microsystems

Lenguajes orientados a objetos

- Deben tener como mínimo tres propiedades básicas
  - Estar basados en objetos
  - Estar basados en clases
  - Poseer conceptos como herencia y polimorfismo
- Existen pocos lenguajes que cumplan las tres propiedades
  - JavaScript está basado en objetos, pero no es orientado a objetos
  - CLU está basado en clases, pero no es orientado a objetos
  - C++ y Java cumplen estas tres propiedades
  - Smalltalk y Eiffel tienen también otras propiedades, son más puros

Historia de C++

- Cronología
  - 1985 Bjarne Stroustrup crea C++ en los laboratorios Bell de AT&T
  - 1989 se constituye el comité X3J16 de ANSI para su estandarización
  - 1997 se publica el estándar ANSI de C++
  - 1998 se publica el estándar ISO de C++ (C++98)
  - 2003 *technical corrigendum* de ISO C++
  - 2011 nuevo estándar de ISO (C++11)
- Supone un compromiso entre la programación a bajo y alto nivel
- Es un lenguaje multiparadigma (no es orientado a objetos puro)
  - El paradigma de la programación estructurada
  - El paradigma de la programación orientada a objetos
  - El paradigma de la programación genérica

Orientación a objetos

- Los sistemas se organizan en torno a objetos, no a datos y procesos
- Se puede pensar en un sistema como en un conjunto de objetos
  - Los objetos poseen características y comportamientos
  - Los objetos se comunican entre sí mediante mensajes
- Pasos:
  1. Identificar los objetos
  2. Agrupar los objetos con características y comportamientos comunes
  3. Identificar las clases
  4. Identificar las relaciones que existen entre las clases
  5. Identificar los mensajes

Principios fundamentales de la orientación a objetos

- Abstracción
  - Facilita la reutilización
- Encapsulado
  - Aumenta la cohesión
- Ocultación de la información
  - Reduce el acoplamiento
- Generalización
  - Disminuye la redundancia
- Polimorfismo
  - Aumenta la capacidad de adaptación y extensibilidad

Ventajas del enfoque orientado a objetos

- Simplicidad del modelo (facilidad de uso)
  - Cercanía de sus conceptos a los del mundo real
  - Uso de los mismos elementos en todo el proceso de desarrollo
- Reutilización
  - Proceso de desarrollo más sencillo y más rápido
- Facilidad de mantenimiento
  - Modificaciones, extensiones y adaptaciones más sencillas
- Fiabilidad
  - Corrección y robustez

Limitaciones del enfoque orientado a objetos

- Rechazo inicial
- Análisis incorrectos
- Lenguajes incompletos
- Menos eficiencia

Factores de calidad del software

- Corrección
  Es la capacidad de los sistemas software de realizar con exactitud sus tareas, tal y como se definen en la especificación
- Robustez
  Es la capacidad de los sistemas software de reaccionar apropiadamente ante condiciones excepcionales
- Extensibilidad
  Es la capacidad de adaptar los sistemas a los cambios de especificación
- Reutilización
  Es la capacidad de los componentes de los sistemas software de servir para la construcción de muchas aplicaciones diferentes

Factores de calidad del software

- **Compatibilidad**
  Es la capacidad de combinar unos componentes con otros
- **Eficiencia**
  Es la capacidad de los sistemas software para exigir la menor cantidad posible de recursos hardware, tales como tiempo de procesador o espacio ocupado de memoria
- **Portabilidad o transportabilidad**
  Es la facilidad de transferir los sistemas software a diferentes entornos
- **Facilidad de uso**
  Es la facilidad con la cual personas con diferentes formaciones pueden aprender a usar el software y aplicarlo a la resolución de problemas

Factores de calidad del software

- Funcionalidad
  Es el conjunto de posibilidades que proporciona un sistema software
- Oportunidad
  Es la capacidad de un sistema software de ser aceptado en el mercado
- Integridad
  Es la capacidad de los sistemas software de proteger sus diversos componentes contra modificaciones o accesos no autorizados
- Economía
  Es la capacidad que un sistema tiene de completarse con el presupuesto
- Facilidad de mantenimiento
  Es la facilidad con la que un sistema se pueda depurar y modificar

Aspectos claves del desarrollo orientado a objetos

- Método y lenguaje
  El enfoque y las notaciones que se deben emplear para analizar y producir software orientado a objetos
- Implementación y entorno
  Las herramientas que permiten a los desarrolladores aplicar la tecnología orientada a objetos
- Bibliotecas
  Los mecanismos que se deben emplear en el enfoque orientado a objetos para la utilización y producción de bibliotecas

Métodos y lenguajes orientados a objetos

- Clases: deben ser el concepto central
- Módulos: las clases deben ser los únicos módulos
- Tipos: todo tipo tiene que estar basado en una clase
- Computación basada en paso de mensajes
- Ocultación de la información
- Asertos: para monitorización y generación de documentación
- Excepciones: como mecanismo de control de errores
- Genericidad no restringida: clases parametrizadas

Métodos y lenguajes orientados a objetos

- Herencia
- Genericidad restringida: mediante la herencia
- Polimorfismo: redefiniciones y conversiones
- Ligadura dinámica: como consecuencia del polimorfismo
- Comprobación estática de tipos y en tiempo de ejecución
- Clases y operaciones diferidas: clases abstractas
- Gestión de memoria y recolección de basura
- Ciclo de vida: los métodos OO pueden aplicarse en todo el ciclo de vida

Implementación y entorno

- Actualización automática
  El análisis de las dependencias debe efectuarse con herramientas
- Actualización rápida
  El tiempo necesario para procesar un cambio debe estar en función del tamaño de lo modificado, y no del tamaño del sistema total
- Persistencia
  Debe disponerse de un mecanismo de almacenamiento de los objetos
- Documentación
  Debe disponerse de herramientas automáticas para su generación
- Navegación
  Debe disponerse de herramientas de navegación interactivas

Bibliotecas

- Bibliotecas básicas
  Estructuras de datos y algoritmos más frecuentes
- Bibliotecas genéricas
  - Interfaces gráficas de usuario
  - Bases de datos
- Mecanismos de indexación de las bibliotecas
  Para permitir una recuperación basada en propiedades