UNCUYO UNIVERSIDAD NACIONAL DE CUYO

FACULTAD DE INGENIERÍA

Programación Orientada a Objetos

Introducción

Mg. Ing. César Aranda

cesar.aranda@ingenieria.uncuyo.edu.ar
unidatos@gmail.com

Ingeniería en Mecatrónica

Objetivos

- Presentar el Paradigma Orientado a Objetos dentro de un contexto global.
- Elección de un OOPL en particular
- Presentar los elementos tecnológicos y metodológicos básicos de la POO
- Discutir las ventajas y desventajas de la POO

Contenidos

- Paradigmas de programación
- Crisis de la Programación Modular
- Programación Orientada a Objetos
- Ventajas y Desventajas de la POO
- Actualidad de los lenguajes OO. Tendencias
- OO vs. IDE/CASE
- Construcción de Aplicaciones OO.

¿Qué es un Paradigma?

- Paradigma (gr.) significa "patrón" o "ejemplo"
- "Conjunto de prácticas que definen a una disciplina científica durante un período de tiempo en particular" (Thomas Kuhn)
- "La probabilidad de que se produzca un avance continuo en la programación requerirá de una invención continua además de la elaboración y colaboración de nuevos paradigmas" (Robert W. Floyd en *The Paradigms of Programming*)

Paradigmas de Programación

- Programación Imperativa
  - Se describen sentencias que modifican el estado de un programa.
  - La solución del problema se expresa especificando una secuencia de acciones a realizar a través de uno o más procedimientos denominados subrutinas o funciones.
    - Programación **estructurada**: C/C++, BASIC, Pascal, PHP, …
    - Programación **modular**: C/C++, BASIC, Pascal, …
    - Programación **orientada a objetos**: C++, C#, PHP, …
- Programación Declarativa
  - Se describe la lógica de computación necesaria para resolver un problema sin describir un flujo de control de ningún tipo.
  - La solución del problema se expresa especificando un conjunto de mecanismos internos de control.
    - Programación **funcional**: Haskell, Scheme, Erlang, Lisp, …
    - Programación **lógica**: Prolog, Lisp, …
    - Programación **restringida o con restricciones**: B-Prolog, ChipV5, …

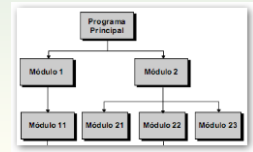
Ing. César Aranda 5

Programación Estructurada (60')

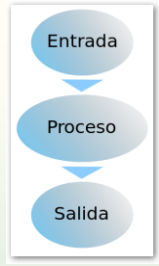
- Teorema del Programa Estructurado: de Böhm-Jacopini, "*sólo tres instrucciones de control*"
  - Secuencial
  - Selectiva
  - Iterativa (bucle con condición al principio)
- Principio de Anidamiento
- Prohibición de Saltos Incondicionales (goto)
- Código Autodocumentado

Ing. César Aranda 6

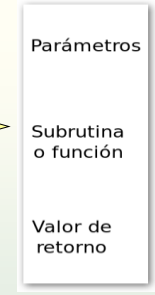
Programación Modular (70')



- Técnica de Diseño Top-Down
  - o de Refinamiento Sucesivo
- Principio de Modularidad
  - Dividir un programa en módulos o subprogramas
  - Cada módulo resuelve una tarea específica



En su forma más simple es

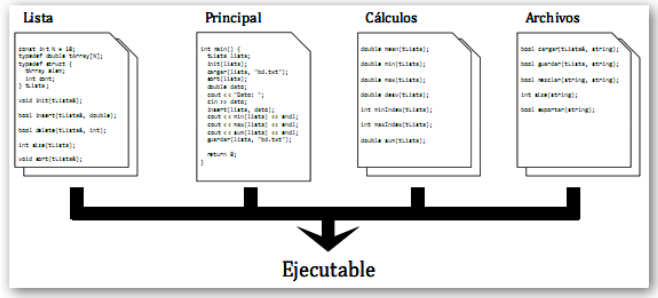


  - **!!!** Módulo ≠ Función o Procedimiento
  - Programas Multiarchivo con Compilación Separada

Ing. César Aranda 7

Programas Multiarchivo

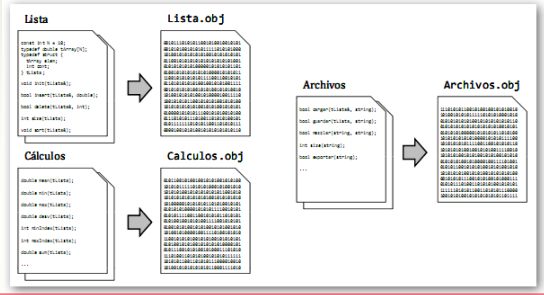
- El código fuente del programa se reparte entre varios archivos (módulos), cada uno con las declaraciones y los subprogramas que tengan relación.
  - Módulo **=** archivo de código fuente de una unidad funcional: una estructura de datos, un conjunto de utilidades, ...



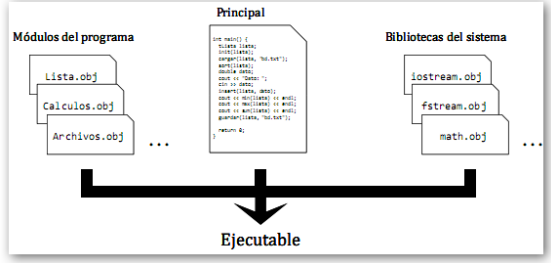
Ing. César Aranda 8

Compilación Separada

- Cada módulo se compila a código fuente de forma independiente



- Al compilar el programa principal, se adjuntan los módulos compilados



Ing. César Aranda 9

Crisis de la Programación Modular

- Los programas de sistemas empresariales se van volviendo cada vez más grandes y complejos
- Aumenta la cantidad de reglas de negocios
- Muchas de esas reglas se presentan como volátiles.
- No existe uniformidad de representación entre las fases de análisis, diseño e implementación.
- Los mismos datos son gestionados desde diferentes porciones de código, lo cual vuelve poco flexibles las tareas de mantenimiento.
- Los datos estables y los datos que cambian con frecuencia se encuentran dispersos en el sistema.
- El Concepto de Módulo no resuelve naturalmente estos problemas

Ing. César Aranda 10

Programación Orientada a Objetos (80')

- Propuesta por Ole-Johan Dahl y Kristen Nygaard ante la necesidad de simular problemas oceánicos en Oslo (1969).
- El concepto fundamental es el Objeto
- Propone una serie de técnicas y mecanismos
  - Herencia
  - Abstracción
  - Polimorfismo
  - Encapsulamiento
  - Paso de mensajes

Ing. César Aranda 11

Procedimientos vs. Objetos

- Descomposición funcional (¿procedimental?)
  - Enfoque principal: Módulos construidos alrededor de las operaciones.
  - Presencia de Datos globales o distribuidos entre módulos.
  - Perspectiva de Diseño: Entrada+Proceso+Salida.
  - Subproducto básico: Organigramas de flujo de datos y/o control.
- Orientación a objetos
  - Enfoque principal: Módulos construidos alrededor de las clases.
  - Presencia de Clases débilmente acopladas, y sin datos globales
  - Perspectiva de Diseño: Encapsulamiento+mensajes.
  - Subproducto básico: Diagramas jerárquicos de clases

Ing. César Aranda 12

Lenguajes OO y Tendencias

- Entre los OOPL actuales se destacan:
  - ABAP, ABL, ActionScript, ActionScript 3, Ada,
  - C++, C#, Clarion, Clipper v5, D, Delphi(Object Pascal),
  - Gambas, Harbour, Eiffel, Fortran 90/95, Java, JavaScript,
  - Lexico, Objective-C, Ocaml, Oz,
  - Perl, PHP v5, PowerBuilder, Python,
  - R, Ruby, Scala, Smalltalk, Magik (SmallWorld), Vala,
  - VB.NET, Visual FoxPro v6, Visual Basic 6.0,
  - Visual DataFlex, Visual Objects, XBase++
- Tendencias
  - Añadir extensiones OO: OOCobol, OOLisp, OOProlog y Object Rexx
  - Nuevos paradigmas de programación: POA (programación orientada a aspectos)

Ing. César Aranda 13

C++/Python IDEs

- Integrated Development Environment
  - Editor de código fuente
  - Editor GUI
  - Compilador
  - Depurador
  - Documentador
  - Ventanas de Salida/Informe de Estado/Errores
  - Gestor de Proyectos
  - Navegador (FS)
  - Gestores de plugins, complementos, librerías
  - Administrador de dependencias, …

Ing. César Aranda 14

OO-CASEs

- Object Oriented Computer Aided Software Engineering
- Características Deseables
  - De Arquitectura: uso de patrones, reutilización, …
  - Del Proceso de Desarrollo: diferentes vistas del modelo, soporte para el profeso de desarrollo, integración con otras herramientas, soporte de notaciones, verificación y simulación de modelos, herramientas visuales, selector de metodología, …
  - De Roundtrip: generación de código, ingeniería inversa, sincronización de artefactos, mecanismos ida/vuelta, integración de código como vista del modelo, refactorización, automatización, …
  - De Métricas: incorporación de métricas, estadísticas, análisis de tendencias, SQA, …
  - De Aspectos de uso: soporte para desarrollo en equipo, CVS, controles de complejidad del modelo, posibilidades de extensibilidad de herramientas, gestión de repositorio, gestión de proyectos, manejo de costos, …

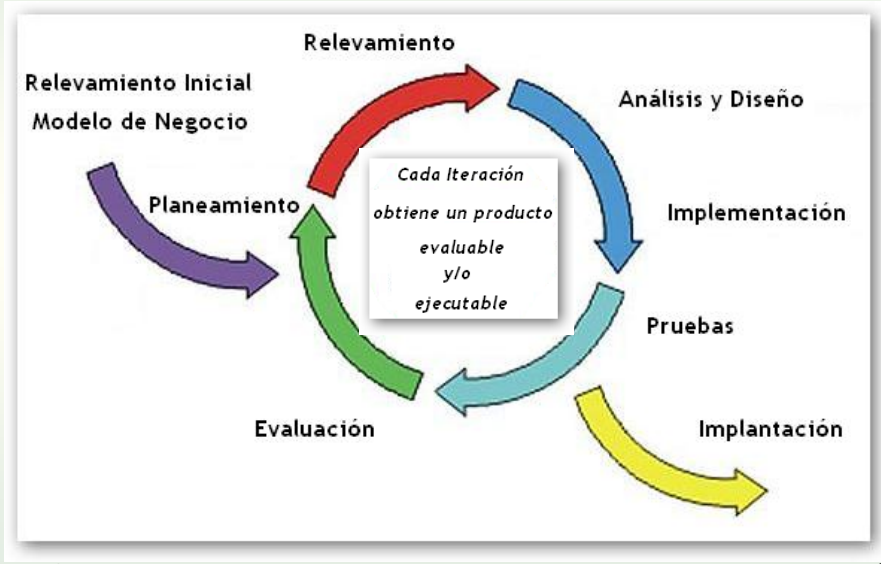
Ing. César Aranda 15

Desarrollo OO

- Un buen desarrollo de software orientado a objetos requiere de Análisis OO, Diseño OO e Implementación en Lenguaje OO.
- El modelamiento visual es clave para realizar AyDOO.
- UML se ha impuesto como un estándar para el modelamiento de sistemas OO.
- Cuando los modelos OO se construyen en forma correcta, son fáciles de comunicar, cambiar, expandir, validar y verificar.

Ing. César Aranda 16

Proceso de Desarrollo OO



Ing. César Aranda 17

Tips OO

- Cualquier lenguaje o método OO puede ser bueno o malo. Depende del uso que se le dé.
- Un ingeniero de software indisciplinado con una herramienta software resulta un peligroso ingeniero de software (A. Davis)
- Una buena herramienta en manos de un mal ingeniero de software produce software de mala calidad con muchísima rapidez (A. Davis)
- La mejor herramienta OOCASE es la que no existe
- Si vas a usar C++, estudia C++ (B. Stroustrup)

Ing. César Aranda 18

Referencias

- Lectura complementaria en archivo: Lectura_02.pdf

- STROUSTRUP, B. (2009): Programming Principles & Practices using C++, Addison Wesley
- WEITZENFELD, A. (2002): Ingeniería de Software Orientada a Objetos Con Uml, Java e Internet, Thomson Learning.
- ACERA GARCIA, M.A. (2010): C/C++, Anaya Multimedia
- ELLIS, M.A. y STROUSTRUP, B. (2004): C++ Manual De Referencia, Ed. Diaz De Santos
- URLs:
  - http://www.buenastareas.com/ensayos/Mito-8-Existen-Demasiadas-Metodologías-De/1868125.html
  - http://www.developerdotstar.com/mag/articles/oo_case.html
  - http://trese.cs.utwente.nl/automatingOOSD
  - http://www.selectbs.com/analysis-and-design/select-architect

Ing. César Aranda 19

ANEXO

Ing. César Aranda 20

Ventajas de la POO

- Aumenta la Similitud del Programa al Mundo Real
- Maximiza la Modularidad y el Encapsulamiento
- Datos separados del Diseño
- Mejor entendimiento de la lógica del programa
- Fácil documentacion y diseño del programa
- Dinamismo en el manejo de los datos
- Reutilización del código que facilita el Mantenimiento y la Extensibilidad del Software
- Facilita la creación de programas visuales
- Facilita la construcción de prototipos
- Facilita el trabajo en equipo
- Permite crear sistemas más complejos

Ing. César Aranda 21

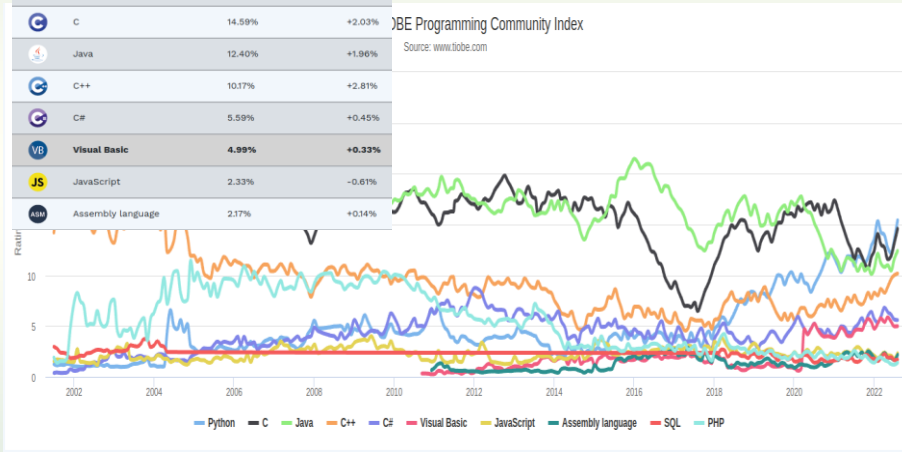
Desventajas de la POO

- El aprendizaje inicial es más costoso
- Complejidad para adaptarse
- Se debe escribir mayor cantidad de código (al menos al principio)
- No es aplicable para resolver todos los tipos de problemas
- Requiere de mayor esfuerzo durante las fases de análisis y diseño
- La depuración en OO es más compleja que la depuración de programas estructurados
- Existe dependencia del lenguaje de programación
- Los estándares están en continua evolución

Ing. César Aranda 22

Posición de C++/Python según TIOBE

Programming Language	Ratings	Change
Python	15.42%	+3.56%
C	14.59%	+2.03%
Java	12.40%	+1.96%
C++	10.17%	+2.81%
C#	5.59%	+0.45%
Visual Basic	**4.99%**	**+0.33%**
JavaScript	2.33%	-0.61%
Assembly language	2.17%	+0.14%



- https://www.tiobe.com/tiobe-index/

Ing. César Aranda 23