FIUBA

75-08 Sistemas Operativos 2004

Programación Orientada a Objetos

Ing. Osvaldo Clúa

Bibliografía: Para quien quiera entrar en análisis y diseño (como encontrar un objeto, una clase y una jerarquía en el mundo real), un panorama de UML hecho por un tesista aparece en http://www.lysator.liu.se/~alla/dia/umltut/ (recuperado por la Wayback Machine de http://www.archive.org/) y un tutorial mas completo en de la Universidad de Kennesaw, Georgia: http://pigseye.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/

La idea de dar persistencia a los objetos fueron explotadas en Grasshopper en la Universidad de Sidney. Aunque el proyecto concluyó, una visita a los *papers* de http://www.gh.cs.su.oz.au/Grasshopper/index.html puede ayudar a comprender la complejidad detrás de una idea aparentemente simple.

Un tema particularmente oscuro, **reflexión** puede leerse de las FAQ de TUNES, un sistema operativo metaprogramado, reflexivo y Orientado a Objetos (clásico de la gente que abraza el credo GNU) http://tunes.org/

En el mismo lugar en que encontró este texto, debió encontrar un archivo *zip* con los códigos listos para compilar y probar. Se usaron los compiladores Java se SUN (java.sun.com), C++ de la *gnu (gcc)* disponible para múltiples plataformas, entre ellas *Win* en *Cygwin* yAda95 *gnat* también disponible gratuitamente en la New York University, conviene acceder desde www.gnat.com ya que el *repository* cambia a veces de ubicación. Tambien *Cygwin* tiene una versión gratuita para Win.

Estos códigos se usarán en clase.

1. Usando los códigos ejemplos, puede observar diferencias en la notación de la orientación a objetos. Sin embargo, puede darnos una definición general de **Clase** y **Objeto.**
2. **Java y C++** son mas parecidos en su notación y en la forma de encapsular los Objetos y hacen extensión de la *typedef struct* de C para integrar procedimientos (métodos). Además requieren que estos métodos sean llamados desde el contexto de un objeto (Ejemplifique, ¿Qué pasa con los métodos de clase, como se llaman en estos casos?). En cambio Ada toma los *tagged records* de su versión 83 (una variación de *variant records* de Pascal) como elemento de herencia, pero los procedimientos quedan externos a los objetos. (Ejemplifique). En los tres casos indique la sintaxis (aproximada) de la herencia.
3. Describa los objetivos del análisis y diseño orientado a objetos.
4. Indique con ejemplos los distintos casos de uso de la herencia para admitir en el programa relaciones de *Clasificación, Agregación, Especialización* o *Agrupamiento.*
5. ¿Qué son los *constructores* y *destructores* de Objetos?
6. ¿Cuál es la discusión acerca de la herencia múltiple?

7. Defina en sus propias palabras *promotion, demotion, casting, overloading* y *polimorfismo*. ¿En qué se diferencia el *Polimorfismo* del resto?.
8. Diferencie *Polimorfismo* de Programación genérica.
9. Describa el concepto de *Framework* y explique con un ejemplo.
10. Compare OLE (COM, DCOM o como queira que se lo llame ahora) con el DOM usado por Mozilla.
11. ¿Qué significa dar *persistencia* a los objetos, y cuales son sus principales problemas?
12. ¿Qué es la programación *reflexiva*? Siga el código entregado.
13. Describa la técnica de *Metaprogramación.*
14. ¿Cómo usaría la Reflexión para responder a cambios de la infraestructura del sistema (por ejemplo, al agregarse un nuevo objeto a la "comunidad")?.