**Name:** Miquely Calvopiña

**Thinking in Java – Introduction Resume**

No necesariamente se debe saber C, pero si uno se siente no preparado puede leer thinking in C

**Progreso de la abstraccion**

Todos los lenguajes proporcionan abstracciones, el programador debe establecer la asociación entre el modelo de la máquina y el modelo del problema. Prolog convierte todos los problemas en cadenas de decisión.

El enfoque orientado a objetos trata de ir un paso más allá proporcionando herramientas al programador para representar los elementos en el espacio del problema, el objetivo es que cuando se lea el código que describa la solución, se estén leyendo palabras que también expresen el problema.

Alan Kay, resumió el Smalltalk, primer lenguaje de java en tener éxito

1) todo es un objeto

2)Un programa es un montón de objetos que se dicen entre si lo que tienen que hacer, enviándose mensajes

3)Cada objeto tiene su propia memoria formada por otros objetos

4) Todo objeto tiene un tipo asociado

5) todos los objetos de un tipo particular pueden recibir los mismos mensajes

Cada objeto tiene una dirección de memoria exclusiva, todo objeto tiene una interfaz. El programador define una clase para adaptar un problema en lugar de forzar el uso de un tipo de datos existente que fue diseñado para representar una unidad de almacenamiento en una máquina.

Una vez definida una clase, se pueden crear tantos objetos de dicha clase como se desee y dichos objetos pueden manipularse como si fueran los elementos del problema que se esté intentando resolver. un gran reto en POO es crear una correspondencia uno a uno entre los elementos del espacio del problema y los objetos del espacio de la solución, las solicitudes que se pueden hacer a un objeto se definen mediante su interfaz, un objeto proporciona servicios.

Una de las mejores formas de pensar en los objetos es como si fueran "proveedores de servicios", antes de crear un objeto se deben hacer preguntas como: ¿Qué servicios proporciona este objeto?, si se hace este planteamiento, el programador podrá decir "Este objeto es lo suficientemente sencillo como para escribirlo yo mismo"

Un problema común entre los programadores es asignar demasiadas funcionalidades al objeto.

**La implementación oculta**

El objetivo del programador cliente es reutilizar una caja de herramientas completas que usar para el desarrollo de aplicaciones. El objetivo del creador de clases es exponer al cliente solo lo necesario y mantener todo lo demás oculto.

En cualquier relación es importante mantener limites +

Modificadores de acceso en java: publica, prívate, protector

**Reutilización de implementación**

La forma más sencilla de revitalizar una clase consiste en emplear directamente un objeto de dicha clase = crear un objeto miembro.

Definir una nueva clase a partir de una clase existente se llama composición.

**Herencia**

Utilizando la herencia se puede construir una jerarquía de tipos que exprese el problema que está intentando resolver en términos de sus tipos, todos los mensajes que se pueden enviar a los objetos de la clase base también se pueden enviar a los objetos de la clase derivada no necesariamente se deben añadir métodos nuevos a la interfaz, para sustituir un método basta con crear una nueva definición para el mismo en la clase derivada .

Relaciones es un y es un como la relación entre la clase base y las clases derivadas se dice que "es un", añadir nuevos elementos de interfaz a un tipo derivado, ampliando la interfaz "es como un"

**objetos intercambiables con polimorfismo**

si no se sabe que fragmento de código se ejecutara entonces se añadirá un subtipo nuevo y el código que se ejecute puede ser diferente sin que sea necesario realizar cambios en el método que lo llama.

**La jerarquía de raíz única**

todas las clases de la última instancia deberían ser heredadas de una única clase base = si

la flexibilidad de c++, solo si se tiene mucho código en C, se puede que todos los objetos de una jerarquía de raíz única tengan una determinada funcionalidad.

**Contenedores**

no se necesita saber cuánto alacena un contenedor, basta con crear un objeto contenedor

dos tipos básicos de contenedores arraylist.

**tipos parametrizados**

un tipo parametrizado es una clase que el compilador puede personalizar automáticamente para

que funcione con cada tipo concreto.

**creación y vida de los objetos**

se debe saber si es o no necesario eliminar un objeto tan fácilmente, si hace falta un nuevo objeto se crea en el cumulo de memoria java tiene depurador de memoria, si un objeto no está en uso, lo destruye.

**tratamiento de excepciones**

una excepción es un objeto generado donde se ha producido el error

**programación concurrente**

el programa se ejecute por separado se llama hebras y el conjunto general se llama ocurrencia.

**Java e internet**

java resuelve programas que surgen de la www, en el repositorio de información esta centralizado el software, repositorio etc se le llama servidor, el que muestra en la propia maquina se llama cliente.