Apuntes de Java

Agenda

- Bienvenida
- Conociendo Java
- La Maquina Virtual
- Descargar e instalar el compilador
- El entorno de trabajo
- El paradigma de la programación orientada a objetos

¿Qué es Java?

- Un lenguaje de programación
- Un entorno de desarrollo
- Un entorno de aplicación
- Un entorno de despliegue
- Es similar en sintaxis de C + +.
- Se utiliza para el desarrollo de applets y aplicaciones.

Maquina Virtual

VM

http://netbeans.org/community/releases/71/

NetBeans

Java

Entorno de Trabajo

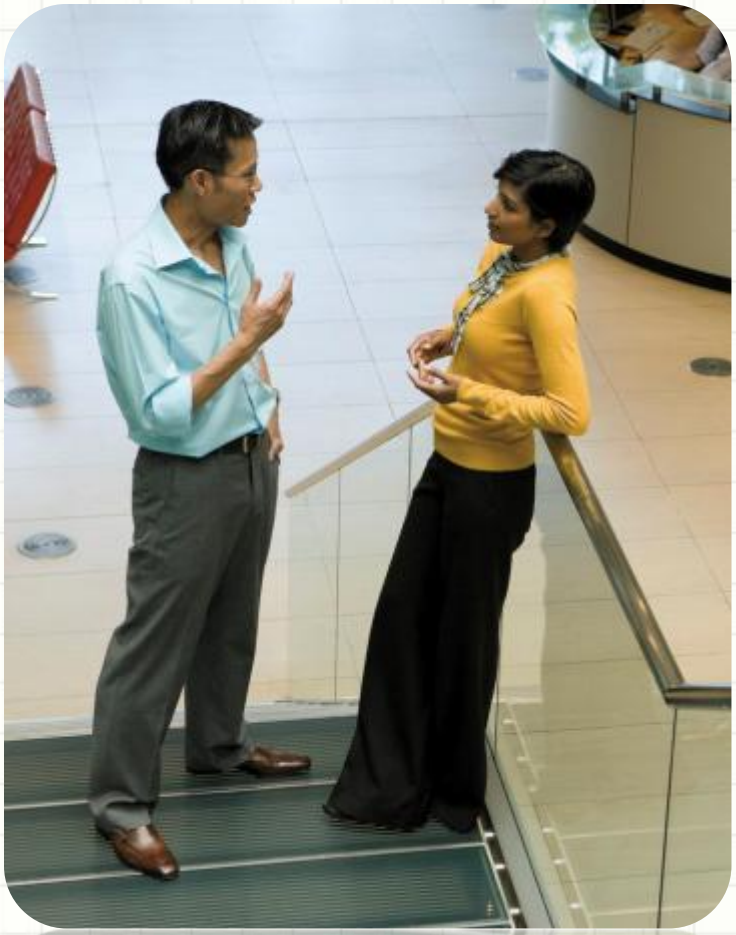
El paradigma de la POO

La **programación orientada a objetos** o **POO** (**OOP** según sus siglas en inglés) es un paradigma de programación que usa los objetos en sus interacciones, para diseñar aplicaciones y programas informáticos.

Está basado en varias técnicas, incluyendo herencia, abstracción, polimorfismo y encapsula miento. Su uso se popularizó a principios de la década de los años 1990. En la actualidad, existe variedad de lenguajes de programación que soportan la orientación a objetos.

Objeto

- Es cualquier cosa
- Tangible o Intangible
- Es parte de algo
- Interactúa con otros objetos
- Es una entidad
- Atributos



Clases

Agrupa los objetos

Es un Mecanismo

La Clase Incluye:

- Herencia
- Polimorfismo

Métodos

- Es la forma de operar
- Los mensajes se asocian con los métodos
- Se escribe e una clase de objetos
- Determina como actúa el objeto cuando recibe el mensaje
- Puede enviar un mensaje a otros objetos

Un Programa orientado a objetos realiza 3 cosas fundamentales:

- Creación de los objetos necesarios
- Los mensajes enviados a unos y otros objetos permiten el proceso interno de la información
- Cunado los objetos no son necesarios, son borrados.

Declaraciones

- Identificadores
- Palabras reservadas
- Convenciones

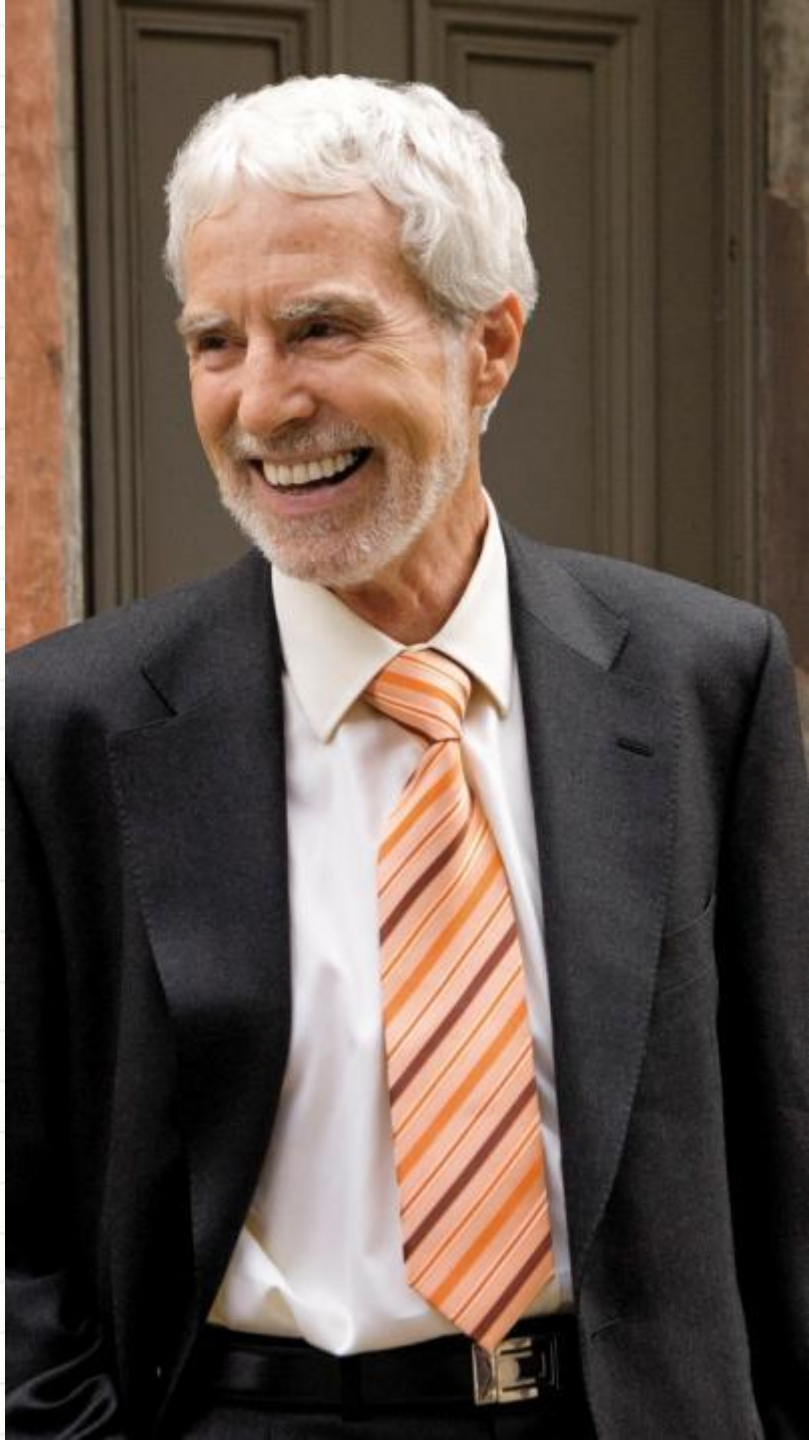
Identificadores

La sintaxis para formar un identificador es:

{ letra | _ | $ } [letra | dígito | _ | $ }]…

Algunos ejemplos son:

- Suma
- suma
- SUMA
- Calculo_Numeros_Primos
- $ordenar
- VisualizarDatos
- _nombre
- _Nombre
- _NOMBRE
- Nombre1
- N123



Palabras reservadas

abstract	default	if	private	this
boolean	do	implements	protected	throw
break	double	import	public	throws
byte	else	instanceof	return	transient
case	extends	int	short	try
catch	final	interface	static	void
char	finally	long	strictfp**	volatile
class	float	native	super	while
const*	for	new	switch	
continue	goto*	package	synchronized	

Convenciones

- Paquetes: en minúsculas (transporte.objetos, utiles.factura).
- Clases e Interfaces: la primera letra en Mayúscula (FacturaElectronica, Facturable).
- Métodos: combinaciones verbo-nombre en camelCase. (getCliente, setAltura, sumarAsientos).
- Variables: nombres en camelCase. (nuevoCliente, saldoMedioInteranual).
- Constantes: en Mayúsculas y si hay que separar usar el subrayado "_" (IPC, TOTAL_NOMINA).
- Estructuras de control: cuando las sentencias forman parte de una estructura de control de flujo, escribirlas entre llaves, aunque sean sentencias sencillas.
- Espacios: solo debe colocarse una sentencia por línea y utilizar sangrías de dos o cuatro espacios para facilitar la lectura.
- Comentarios: utilizar comentarios para explicar segmentos de código no obvios.

Declaración de una clase

La sintaxis para declarar una clase es la siguiente:

- [Modificadores] class NombreClase [extends SuperClase] [implementes Interface]{

}

La mínima expresión de una declaración de clase sería:
class MinimaClase{

}

Reglas de Declaración en el fichero fuente

- Solo puede existir una clase pública en un fichero .java
- El nombre del fichero debe coincidir con el de la clase pública.
- La sentencia package (si existe) debe ser la primera sentencia del fichero.
- Las sentencias import (si existen) deben seguir a la sentencia package y preceder a las declaraciones de clases.
- Pueden existir más clases en el fichero pero no pueden ser públicas.
- Las sentencias package e import afectarán a todas las clases declaradas en el fichero.
- El nombre de los ficheros que solo tengan declaraciones de clase no públicas no tiene que coincidir con ninguna de las clases.

Modificadores de acceso

- private: solo es accesible dentro de su clase.

  no se especifica (nivel de paquete): es accesible dentro de su clase y por todas las clases de su paquete.
- protected: es accesible dentro de su clase, por todas las clases de su paquete y por las clases hijas que estén en otro paquete diferente.
- public: es accesible para cualquier clase Java.

Interfaz

- una clase implementa una interfaz cuando implementa todos los métodos declarados en ella.
- Varias clases pueden implementar la misma interfaz. Una sola clase puede implementar varias interfaces.
- La sintaxis para declarar una interfaz es la siguiente:

```
[Modificadores] interface
    NombreInterface [extends InterfacePadre]{
            <public static final atributos>
            <public abstract metodos>
     }
```

- Todos los métodos declarados en una interfaz son public y abstract, aunque no se especifique.
- Todos los atributos en una interfaz son public, static y final, aunque no se especifique. Es decir, constantes.
- Los métodos no pueden ser static ni final, strictfp, o native.

Modificadores

- Modifica el nivel de protección predeterminado del método
- Los niveles de protección son:
  - Paquete
  - Publico
  - Privado
  - Protegido

Argumentos variables

- Para especificar un método con un número variable de argumentos, se define el tipo seguido de '…', un espacio y un nombre para el array asociado.

- El argumento variable debe ser el último en la declaración del método y solo se permite uno.

[Modificador] ValorDeRetorno NombreMetodo (Tipo… a) {}

Constructores

- Conjunto de sentencias para inicializar una instancia.
- No se consideran métodos.
- No tienen valores de retorno ni se heredan.
- Debe coincidir con el de la clase.

Cada clase tiene al menos un constructor. Si no se escribe ninguno Java suministra uno por defecto. En ese caso el constructor no tiene argumentos y su cuerpo está vacío.

Constructores

En el momento que nosotros escribamos un constructor se pierde el constructor por defecto.

La sintaxis es:

[Modificadores] nombreClase (<argumentos>){}

Variables

Por el tipo se pueden dividir en:

- variables de tipos primitivos.
- variables de tipos de referencia.

Variables

Por el lugar en que se definen:

- variables locales (dentro de un método, parámetros del método)
- variables miembro (fuera de un método, en la definición de la clase)
- variable de clase (fuera de un método, en la definición de una clase, lleva la palabra static)

Tipos Primitivos en Java

Son simples valores, no objetos.

Los tipos de referencia se utilizan para tipos más complejos.

Java define ocho tipos de datos primitivos, que se dividen en cuatro categorías:

- Lógicos: boolean
- Texto: char
- Enteros: byte, short, int, long
- Reales: float, double

Variables Locales

- Se definen dentro de un método, también reciben el nombre de variables temporales o de pila.
- Se crean cuando el programa empieza a ejecutar el método.
- Se destruyen cuando finaliza dicha ejecución.
- El único modificador que pueden usar las variables locales es static.

Variables Miembro y de Clase

- Las variables de clase, marcadas con static, se crean cuando se carga la clase y siguen existiendo mientras ésta se mantenga cargada.
- Las variables miembro existen mientras exista el objeto asociado.
- Las variables de clase y variables miembro se inicializan automáticamente en el momento de crearse.
- Las variables no locales (miembro y de clase) pueden utilizar los siguientes modificadores: public, protected, private, static, final, transient, volatile.

Variables Miembro y de Clase

Tipo de la variable	Valor predeterminado
byte, short, int	0
long	0L
float	0.0F
double	0.0D
char	'\u0000'
boolean	false
Tipos de referencia	null