

Introducción a Java

Fernando Cerezal López

24 Noviembre 2005

Modificadores disponibles

- de acceso
 - public
 - private
 - protected
- de tipo
 - static
 - abstract
 - final
 - modificadores más avanzados que no vamos a ver...

Clase

Sintaxis

```
modificadores class nombreClase{}
```

Tipos primitivos

Tipo	Rango de valores	Capacidad
byte	[-128, 127]	1 byte
short	[-32768, 32767]	2 bytes
int	Enteros [-2147483648, 2147483647]	4 bytes
long	Enteros [-2^{63} , $2^{63} - 1$]	8 bytes
float	Racionales [10^{-46} , 10^{38}]	4 bytes
double	Racionales [10^{-324} , 10^{308}]	8 bytes
char	Carácter Unicode(UTF-16)	2 bytes
boolean	1=true ó 0=false	1 bit

Si a una variable aún no se le ha asignado un valor, su valor es «null». Si un método no devuelve nada se utiliza el tipo «void» (vacío).

Estructura de una clase - Atributos

Declaración de atributos

```
modificadores class nombreClase{  
modificadores tipo1 nombreAtributo1;  
modificadores tipo2 nombreAtributo2;  
modificadores tipo2 nombreAtributo3;  
modificadores tipo3 nombreAtributo4, nombreAtributo5;  
}
```

Inicializar variables

Inicializar variables

Si es un tipo primitivo o un «String» se puede inicializar (darle su primer valor) así:

```
public boolean nombreAtributo=true; //para un boolean
public static int nombreAtributo1=0; //para un entero byte,
short, int o long
private float nombreAtributo2=2.0; // para un racional float o
double
private static char nombreAtributo3='c'; //para un carácter
String nombreAtributo4="cadena de caracteres"; // para un
String
```

Estructura de una clase - Métodos

Sintaxis

```
modificadores tipoADevolver nombreMétodo(tipoParámetro  
nombreParametro1, tipoParámetro2 nombreParámetro2){  
return tipoADevolver;  
}
```

Si es «void» se pone solo return, si no hay «return» sale al llegar a la última instrucción.

```
Ejemplos: public static int sumar(int primero, int segundo){  
int total=primero+segundo;  
return total;  
}
```

Un método especial: el constructor

Sintaxis

```
modificadores class NombreClase{  
public void NombreClase(){} }
```

Un método especial: el constructor

Ejemplo

```
public class NombreClase{  
    int nombreAtributo1;  
    String nombreAtributo2;  
    float nombreAtributo3;  
    public void NombreClase(int cosa1, String cosa2){  
        nombreAtributo1=cosa1;  
        nombreAtributo2=cosa2;  
        nombreAtributo3=0.0;  
    }  
}
```

Paquetes

Sintaxis

Para la creación de un paquete solo hay que añadir una línea antes de la clase, al compilarlo se creará automáticamente. Si no se pone nada la clase pertenece al paquete por omisión.

Ejemplo

```
package nombrePaquete;  
public class NombreClase
```

Paquetes II

Sintaxis

Para poder utilizar las clases de un paquete hay que importarlo, esto es, decir al compilador que tiene que cargar esas clases.

Ejemplo

```
import nombrePaqueteGeneral.nombrePaquete;  
public class NombreClase
```

Modificadores - protected

Definición

Es una mezcla de «public» y «private». Se define como «public» para las clases derivadas de esta y como «private» para las demás.

Modificadores - static

Definición

Lo definido como «static» no se instancia, esto es, no se hace copia de ello y lo comparten todos los objetos de la misma clase. Así se crea un atributo de clase. Si es una clase, entonces no se hacen objetos de ella y todos sus métodos y atributos son «static».

Sintaxis

- Sintaxis para una clase:
modificadorAcceso static class NombreClase{}
- Sintaxis para un atributo:
modificadorAcceso static tipo nombreAtributo;
- Sintaxis para un método
modificadorAcceso static tipoADevolver
nombreMétodo(tipo parámetro){}

Un método especial: main

Sintaxis

```
public static void main(String args[]){}
```

Encapsulación

Implementación

Se cambian todos los atributos a «private» y se implementan unos métodos para acceder a ellos y establecer sus valores.

Ejemplo:

Sin encapsular

```
public class Persona{  
    String nombre;  
    String dni;  
}
```

Ver clase encapsulada en ejemploEncapsulado.java.

Polimorfismo I - Sobrecarga

Definición

La sobrecarga consiste en que puede haber varios métodos que se llamen igual, pero que se les pase distinto número de parámetros o de distinto tipo (si el lenguaje es tipado).

Polimorfismo I - Sobrecarga

Ejemplos:

- `public void metodo1(tipo1 variable){}`
- `public void metodo1(tipo1 variable,tipo2 variable2){}`
- `public void metodo1(tipo2 variable2){}`
- `public int metodo1(tipo1 variable){}` -¿ Error

Herencia

Sintaxis

```
modificadores class NombreClase extends  
ClaseDeLaQueHereda
```

Herencia por delegación

Definición

Como en Java solo se puede heredar de una clase, se utiliza una solución ad-hoc: la herencia por delegación. Esta se consigue añadiendo un atributo de la clase que se quiere heredar y haciendo que los métodos de la clase llamen a los métodos del atributo.

Herencia por delegación

Ejemplo

Queremos que ClaseNueva herede de Clase1 y Clase2, método1 es de Clase2:

```
public class ClaseNueva extends Clase1 {
    Clase2 nombreAtributo;
    public void ClaseNueva() {
        atributo = new Clase2();
    }
    public void metodo1() {
        nombreAtributo.metodo1();
    }
}
```

Palabras clave - super

Definición

Se puede llamar a la superclase de una clase, esto es, la clase de la que ha heredado con la palabra clave «super».

Sintaxis

```
super.metodoClaseMadre();
```

Palabras clave - this

Definición

El `this` se sustituye por la clase en la que se use, esto es, `this.atributo` se refiere al atributo de la misma clase.

Es útil para saber cuando se refiere a los atributos de la clase y para diferenciarlo cuando una variable local se llama igual que un atributo.

Interfaces

Definición

Es la estructura de una clase, donde solo se definen las cabeceras de los métodos, no su implementación. Se utiliza cuando se hace una generalización de clases que comparten los nombres de métodos, pero no las acciones que estos llevan a cabo. Hay que implementar todos los métodos que defina la interfaz.

Sintaxis

Al definir:

```
public interface nombreInterfaz {}
```

Al utilizar:

```
public class implements nombreInterfaz {}
```

Modificadores - abstract

Definición

Un método se define como «abstract» cuando solo se define la cabecera del método, no su implementación, dentro de una clase. Si una clase contiene al menos un método «abstract», entonces la clase se debe declarar «abstract».

Sintaxis

```
public abstract class NombreClase{  
    public abstract void nombreMétodo{  
        acciones...  
    }  
}
```

Casting

Sintaxis

`tipo1 nombre= (tipo1) variableTipo2`, transforma una variable de tipo2 en tipo1.

Dudas

¿?

Bibliografía

- API de Java: <http://java.sun.com/reference/api/>
- Javahispano: <http://www.javahispano.org>
- Curso de David Muñoz: <http://cofio.gul.uc3m.es/wiki/docs-n-howtos/java/publicstaticvoidmain.pdf>
- Listas de correo usuarios y desarrolladores de Java. . .
- Lista del GUL-uc3m: gul@gul.uc3m.es

Agradecimientos

- A la gente del GUL por organizar estos cursos.
- A vosotros por venir.

Agradecimientos

- A la gente del GUL por organizar estos cursos.
- A vosotros por venir.