



Java Básico

Clases y Objetos

Copyright

- Copyright (c) 2004
José M. Ordax
- Este documento puede ser distribuido solo bajo los términos y condiciones de la Licencia de Documentación de javaHispano v1.0 o posterior.
- La última versión se encuentra en
<http://www.javahispano.org/licencias/>

Clases

- La implementación de una clase Java debe ir en un fichero en formato texto con la extensión *.java y nombre idéntico a la clase implementada.
- La clase MiClase debe ir en un fichero: MiClase.java
- La declaración de una clase Java se realiza mediante la *keyword*: class seguida de su nombre.
- La *keyword* siempre va precedida por un modificador de acceso: public, protected, private o default (nada).

Clases

- La implementación de la clase irá contenida en un bloque { } justo después de la declaración.
- Declaración de una clase:
 - *modificador_acceso class nombre_clase*
{
}
- Ejemplo:
 - **public class** MiClase
{
}

Atributos y métodos

La implementación de una clase consiste en una serie de:

Atributos.

Métodos.

Declaración de un atributo:

modificador_acceso tipo nombre [= valor_inicial];

Ejemplo:

```
private boolean sw = true;  
private int i;
```

Atributos y métodos

Declaración de un método:

modificador_acceso tipo_retorno nombre([tipo parametro,..])
{
}

La implementación del método irá contenida en un bloque { } justo después de la declaración.

Ejemplo:

```
public int suma(int param1, int param2)  
{  
    return param1 + param2;  
}
```

Atributos y métodos

- J2SE 5.0 añade una novedad a la definición de un método mediante la característica: varargs
- Se permite definir un número indefinido de parámetros del mismo tipo mediante: ...
- Lo que recibimos es un array del tipo definido.
- Ejemplo:

```
public int suma(int... params)
{
    int acum = 0;
    for(int num: params) // He usado también el nuevo for/in
        acum = acum + num;
    return acum;
}
```

Atributos y métodos

- Hay que tener en cuenta que podemos recibir, cero, uno o varios valores en dicho parámetro.
- Tiene algunas restricciones:
 - Solo puede usarse una vez por método.
 - Siempre debe ser el último de todos en la definición.
- Ejemplo:

```
public int metodo(String param1, int param2, int... params)
{
}
```
- Veremos como trabajar con esta característica en el capítulo dedicado a J2SE 5.0

Constructores

- Existe un tipo de método especial en Java llamado constructor.
- Sirve para la construcción (instanciación) de objetos (instancias) a partir de esa clase.
- En su implementación se suele dar valores a los atributos para ese objeto.
- Su declaración es idéntica a la de los métodos convencionales con dos salvedades:
 - No tienen tipo de retorno.
 - Se tiene que llamar igual que la clase.

Constructores

- Declaración de un constructor:
 - `modificador_acceso nombre([tipo parametro,...])`
`{`
`}`
- Ejemplo:
 - `public MiClase(int param1, boolean param2)`
`{`
`}`
- Si nuestra clase no tiene constructores, el compilador añade por defecto uno sin parámetros.

Sobrecarga de métodos

- Se dice que un método está sobrecargado cuando existen dos métodos con el mismo nombre y tipo de retorno pero con parámetros distintos.
- De esta manera podemos tener en una clase varios constructores:
- Ejemplo:

```
public MiClase()  
{  
}  
public MiClase(int param1, boolean param2)  
{  
}
```

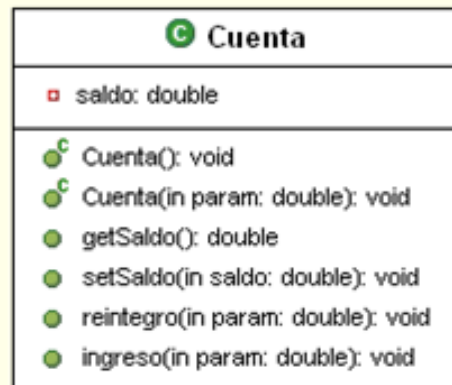
Sobrecarga de métodos

- J2SE 5.0 añade una novedad al respecto.
- Se permite la sobrecarga de métodos cambiando también el tipo de retorno, pero siempre que:
 - El método que se está sobrecargando sea de una clase padre (de la que heredamos directa o indirectamente).
 - El nuevo tipo de retorno sea hijo del tipo de retorno del método original (es decir, que herede de él directa o indirectamente).
- Por tanto, no es válido para tipos primitivos.
- Veremos con mas detalle esta nueva característica en el capítulo dedicado a J2SE 5.0

Convenciones en Java

- El nombre de las clases comenzará con mayúsculas:
 - MiClase, String, Circulo, Cuenta, CuentaCorriente,.....
- El nombre de los atributos comenzará con minúsculas:
 - contador, switch, i, segundoContador,.....
- El nombre de los métodos comenzará con minúsculas (a excepción de los constructores):
 - ingresar, miMetodo, sumar,.....

Ejemplo



Ejemplo

```
public class Cuenta
{
    // Atributos
    private double saldo;

    // Constructores
    public Cuenta()
    {
        saldo = 0.0;
    }

    public Cuenta(double param)
    {
        saldo = param;
    }

    // Métodos
    public void reintegro(double param)
    {
        saldo = saldo - param;
    }

    public void ingreso(double param)
    {
        saldo = saldo + param;
    }

    public double getSaldo()
    {
        return saldo;
    }

    public void setSaldo(double param)
    {
        saldo = param;
    }
}
```

Objetos

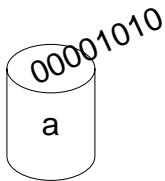
- Los objetos en Java no son mas que variables de tipo complejo, frente a las de tipo primitivo.
- El tipo de un objeto es la clase de la que se ha instanciado.
- La declaración de un objeto es idéntica a la declaración de una variable de tipo primitivo:
 - tipo identificador;*
 - Cuenta miCuenta;

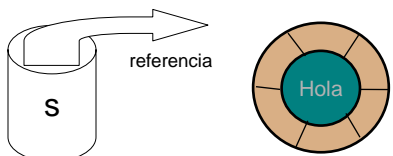
Objetos

- El valor por defecto de un objeto sin inicializar es:
null
- La inicialización de un objeto si que es algo distinta a la inicialización de las variables de tipo primitivo:
 - Se utiliza el operador *new*.
 - Se llama a un constructor de la clase de la que queremos instanciar.
- Es decir:
 - tipo identificador = new tipo([parametro,...]);*
 - Cuenta miCuenta = **new** Cuenta(1200.75);

Variables primitivas vs. complejas

- Una variable de tipo primitivo contiene el dato directamente:

byte a = 10;

- Una variable de tipo complejo contiene una referencia (puntero) a la zona de memoria donde está el objeto:

String s = new String("Hola");


Ejemplo

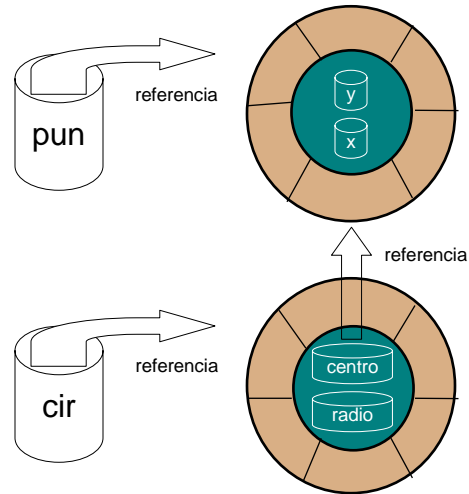
```
public class Punto
{
    private int x = 0;
    private int y = 0;

    public Punto(int param1, int param2)
    {
        x = param1;
        y = param2;
    }
}

public class Circulo
{
    private Punto centro = null;
    private int radio = 0;

    public Circulo(Punto param1, int param2)
    {
        centro = param1;
        radio = param2;
    }
}

public class Test
{
    public static void main(String[] args)
    {
        Punto pun = new Punto(2,2);
        Circulo cir = new Circulo(pun,3);
    }
}
```



Manejo de objetos

- El trabajo con un objeto consiste en acceder:
 - A sus atributos.
 - A sus métodos.
- En ambos casos utilizaremos el operador . (punto).
- Acceso a un atributo:
 - *objeto.atributo*
 - `miCuenta.saldo = 0;`

Manejo de objetos

- Acceso a un método (lo que en Orientación a Objetos se denominaba mensaje):
 - `objeto.metodo([parametro,...])`
 - `miCuenta.reintegro(13.7);`
- La posibilidad de acceso a un atributo o a un método de un objeto dependerá del modificador de acceso que exista en su definición.

Ejemplo

```
public class Cuenta
{
    // Atributos
    private double saldo;

    // Constructores
    public Cuenta()
    {
        saldo = 0.0;
    }

    public Cuenta(double param)
    {
        saldo = param;
    }

    // Métodos
    public void reintegro(double param)
    {
        saldo = saldo - param;
    }

    public void ingreso(double param)
    {
        saldo = saldo + param;
    }

    public double getSaldo()
    {
        return saldo;
    }

    public void setSaldo(double param)
    {
        saldo = param;
    }
}
```

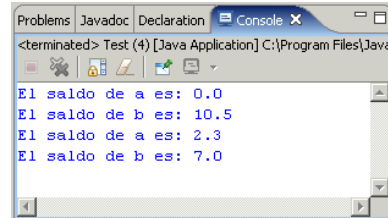
Ejemplo

```
public class Test
{
    public static void main(String[] args)
    {
        double aux = 0;
        Cuenta a = new Cuenta();
        Cuenta b = new Cuenta(10.5);

        aux = a.getSaldo();
        System.out.println("El saldo de a es: " + aux);
        aux = b.getSaldo();
        System.out.println("El saldo de b es: " + b.getSaldo());

        a.ingreso(2.3);
        b.reintegro(3.5);

        System.out.println("El saldo de a es: " + a.getSaldo());
        System.out.println("El saldo de b es: " + b.getSaldo());
    }
}
```



Manejo de objetos

Las llamadas a métodos se pueden encadenar:

```
String s1 = new String("abc");  
char c = s1.toUpperCase().charAt(0);
```

Equivaldría a:

```
String s1 = new String("abc");  
String s2 = s1.toUpperCase();  
char c = s2.charAt(0);
```

El método main y la clase “truco”

- Existe un método especial en Java llamado main:
○ **public static void** main(String[] args)
- Es el método donde comienza la ejecución de un programa Java.
- Las clases representaban entidades que participaban en la resolución de un problema. ¿En qué entidad tiene sentido incluir el método main?
- En ninguna. Por eso crearemos siempre una clase a parte, que solo tenga el método main.

Destructores

- Los destructores son unos métodos encargados de eliminar los objetos de memoria.
- En Java no existe este tipo de métodos.
- En Java lo que existe es un proceso que se ejecuta en la JVM a la vez que nuestra aplicación y que se encarga de buscar todos aquellos objetos en memoria no utilizados y limpiarlos.
- Este proceso se llama Garbage Collector.

Garbage Collector

- ¿Cómo sabe el Garbage Collector que un objeto ya no está siendo utilizado por la aplicación y que por tanto puede ser eliminado?
- Porque no está referenciado por ninguna variable.
- Existen tres motivos por los que una variable deja de referenciar a un objeto:
 - Se iguala a null.
 - Se iguala a otro objeto.
 - Se termina su ámbito.

Ejemplo

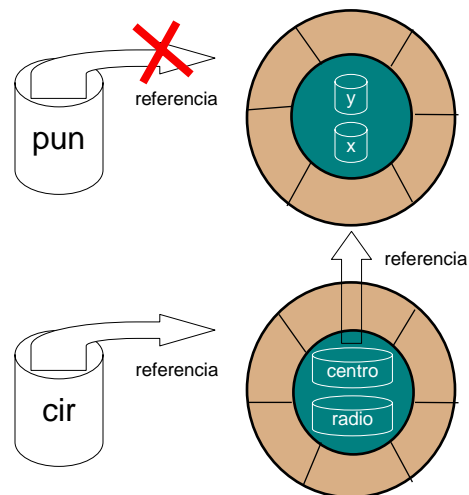
```
public class Punto
{
    private int x = 0;
    private int y = 0;

    public Punto(int param1, int param2)
    {
        x = param1;
        y = param2;
    }
}

public class Circulo
{
    private Punto centro = null;
    private int radio = 0;

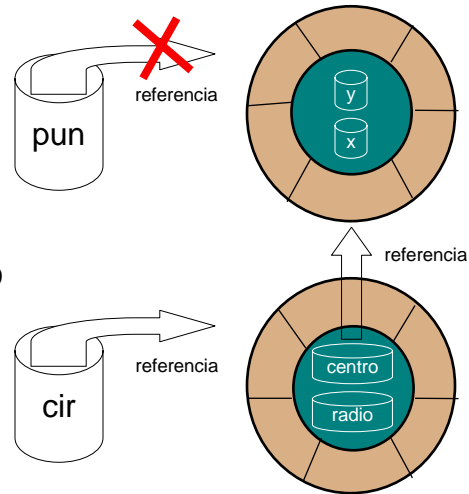
    public Circulo(Punto param1, int param2)
    {
        centro = param1;
        radio = param2;
    }
}

public class Test
{
    public static void main(String[] args)
    {
        Punto pun = new Punto(2,2);
        Circulo cir = new Circulo(pun,3);
        pun = null;
    }
}
```



Ejemplo

Pero ¡ojo!, el objeto referenciado por *pun* no se puede limpiar porque sigue referenciado por *centro* desde el objeto referenciado por *cir*.



Ejercicio

 Identificar si hay algo mal en este código:

```
public class RadioCasette
{
    boolean puedeGrabar = false;

    void escucharCinta()
    {
        System.out.println("Escuchándose cinta");
    }

    void grabarCinta()
    {
        System.out.println("Grabándose cinta");
    }
}

public class Test
{
    public static void main(String[] args)
    {
        r.escucharCinta();

        if(r.puedeGrabar)
            r.grabarCinta();
    }
}
```

Ejercicio (solución)

 Estaba mal. No habíamos creado el objeto r.

```
public class RadioCasette
{
    boolean puedeGrabar = false;

    void escucharCinta()
    {
        System.out.println("Escuchándose cinta");
    }

    void grabarCinta()
    {
        System.out.println("Grabándose cinta");
    }
}

public class Test
{
    public static void main(String[] args)
    {
        RadioCasette r = new RadioCasette();
        r.escucharCinta();

        if(r.puedeGrabar)
            r.grabarCinta();
    }
}
```

Ejercicio

 Identificar si hay algo mal en este código:

```
public class ReproductorDVD
{
    boolean puedeGrabar = false;

    void grabarDVD()
    {
        System.out.println("Grabándose");
    }
}

public class Test
{
    public static void main(String[] args)
    {
        ReproductorDVD r = new ReproductorDVD();
        r.verDVD();

        if(r.puedeGrabar)
            r.grabarDVD();
    }
}
```


Ejercicio

 Estaba mal. Se estaba llamando a un método inexistente.


```
public class ReproductorDVD
{
    boolean puedeGrabar = false;

    void verDVD()
    {
        System.out.println("Viéndose");
    }

    void grabarDVD()
    {
        System.out.println("Grabándose");
    }
}

public class Test
{
    public static void main(String[] args)
    {
        ReproductorDVD r = new ReproductorDVD();
        r.verDVD();
        if(r.puedeGrabar)
            r.grabarDVD();
    }
}
```

Ejercicio

 Identificar si hay algo mal en este código, suponiendo que la clase Rectangulo existe.

```
public class Temp
{
    public static void main(String[] args)
    {
        Rectangulo miRect;
        miRect.ancho = 40;
        miRect.alto = 50;
        System.out.println("El área del rectángulo es" + miRect.area());
    }
}
```

Ejercicio (solución)

- Estaba mal. El objeto miRect no está inicializado, por tanto vale null. Con null no podemos hablarnos.

```
public class Temp
{
    public static void main(String[] args)
    {
        Rectangulo miRect = new Rectangulo();
        miRect.ancho = 40;
        miRect.alto = 50;
        System.out.println("El área del rectángulo es" + miRect.area());
    }
}
```

Bibliografía

- Head First Java
Kathy Sierra y Bert Bates.
O'Reilly
- Learning Java (2nd edition)
Patrick Niemeyer y Jonathan Knudsen.
O'Reilly.
- Thinking in Java (3rd edition)
Bruce Eckel.
Prentice Hall.
- The Java tutorial
<http://java.sun.com/docs/books/tutorial/>

