Encapsulación: clases y objetos

Java y Servicios Web I
Master en Ingeniería Matemática

Manuel Montenegro
Dpto. Sistemas Informáticos y Computación

Desp. 467 (Mat)

montenegro@fdi.ucm.es

Contenidos

- Clases y objetos. Atributos.
- Métodos.
- Modificadores de acceso (`public`/`private`)
- Constructores.
- Igualdad de objetos.
- Ejemplos.
- Atributos y métodos estáticos.
- Paquetes.

Clases

- Una clase es un tipo, definido mediante atributos y métodos.
- Los atributos son variables que definen el estado interno de los objetos de la clase.
- Los métodos son funciones/procedimientos que acceden y/o modifican los atributos de un objeto.
  - Conceptualmente, representan mensajes destinados a un objeto.

Atributos de una clase

```
// Fecha.java

public class Fecha {

    public int dia;
    public int mes;
    public int año;

}
```

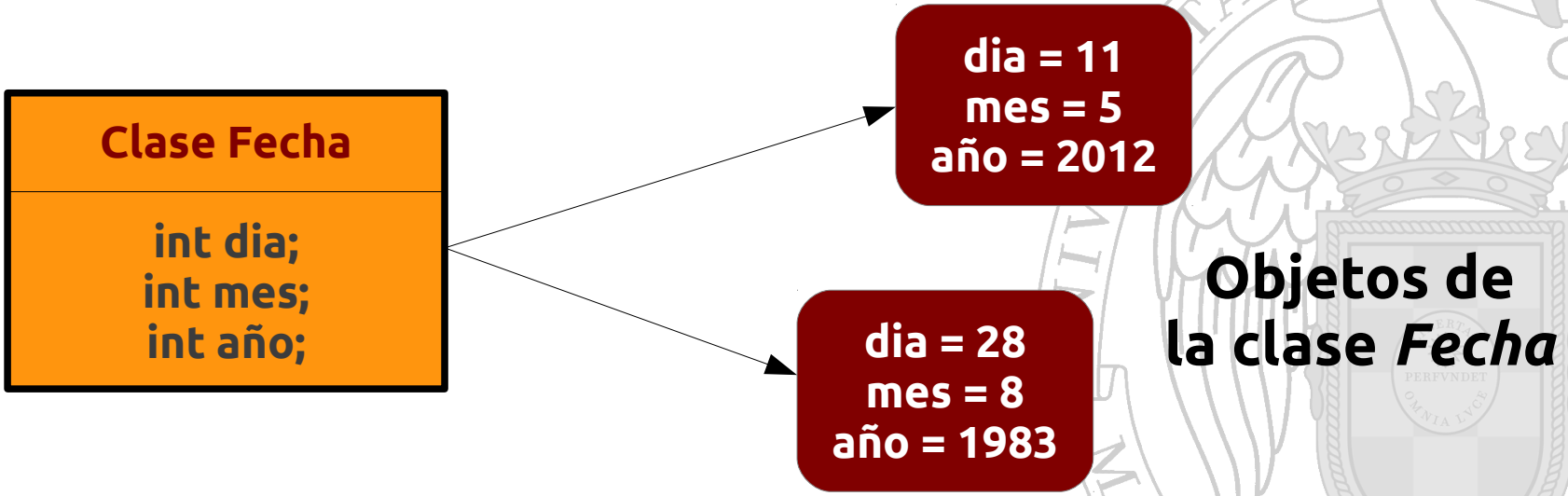
Atributos

Tipo

Nombre

Clases vs. Objetos

- Desde el punto de vista del compilador, las clases son tipos, y los objetos son variables con esos tipos.
- Un objeto es una asignación de los atributos definidos por la clase a valores concretos.
- Una clase puede considerarse como una plantilla, a partir de la cual se crean objetos, que son instancias de la clase.



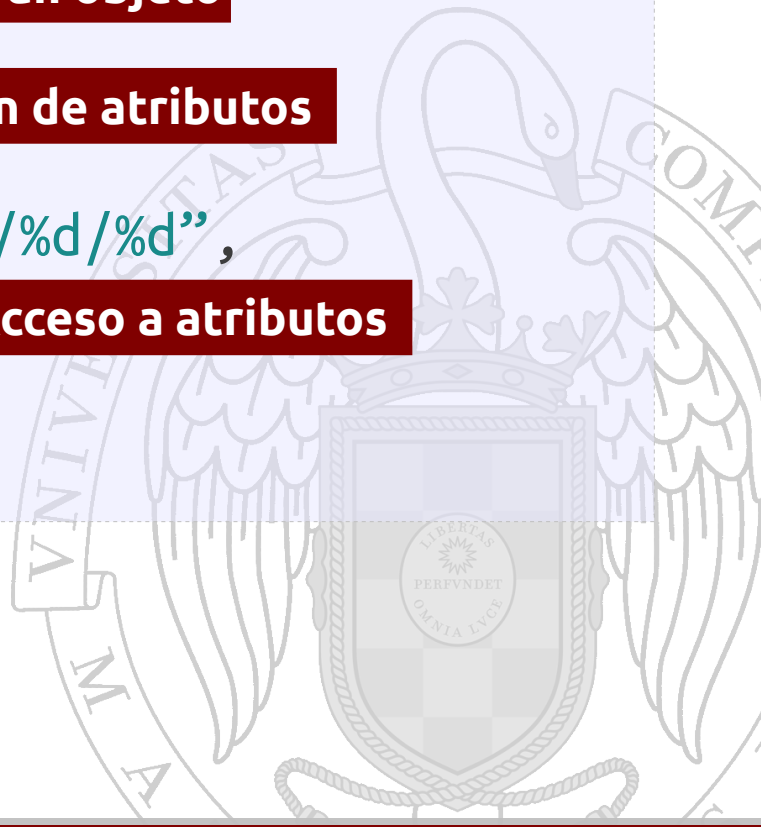
Creación de objetos

```
// Test.java
public class Test {
  public static void main(String[] args) {
    Fecha f;
    f = new Fecha();
    f.dia = 12;
    f.mes = 5;
    f.año = 2012;
    System.out.printf("Hoy es día: %d/%d/%d",
       f.dia, f.mes, f.año);
  }
}
```

Creación de un objeto

Modificación de atributos

Acceso a atributos



Creación de objetos

```
// Test.java
public class Test {
   public static void main(String[] args) {
      Fecha f;
      f = new Fecha();
      f.dia = 12;
      f.mes = 5;
      f.año = 2012;
      System.out.printf("Hoy es día: %d/%d/%d",
         f.dia, f.mes, f.año);
   }
}
```

Creación de un objeto

Modificación de atributos

Acceso a atributos

```
Fecha f = new Fecha();
```

Contenidos

- Clases y objetos. Atributos.
- Métodos.
- Modificadores de acceso (`public`/`private`)
- Constructores.
- Igualdad de objetos.
- Ejemplos.
- Atributos y métodos estáticos.
- Paquetes.

Métodos de una clase

- Son funciones que se declaran dentro del cuerpo de la clase.

```
// Fecha.java
public class Fecha {
    public int dia;
    public int mes;
    public int año;

    public void imprimir() {
      System.out.printf("%d/%d/%d", dia, mes, año);
    }
}
```

Llamada a métodos

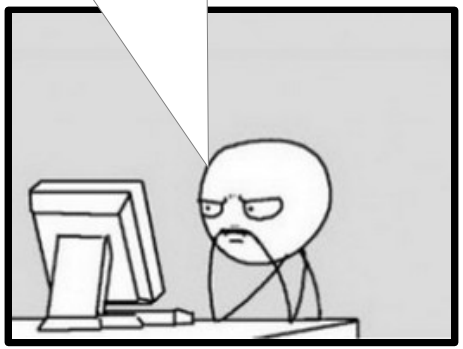
```
// Test.java
public class Test {
  public static void main(String[] args) {
    Fecha f;
    f = new Fecha();
    f.dia = 12;
    f.mes = 5;
    f.año = 2012;
    System.out.print("Hoy es día: ");
    f.imprimir();
  }
}
```

Llamada a método

Métodos de acceso (*getters*)

- En general, no conviene que el usuario de la clase maneje directamente los atributos de la misma.

```
type Fecha = record
  // número de días transcurridos desde el
  // 1 de enero de 1970
  numeroDias:integer;
end;

function DiferenciaFechas(f1, f2:Fecha):integer;
begin
  return f2.numeroDias - f1.numeroDias;
end;
```



```
var actual, f: fecha;

write(f.dia, '/', f.mes, '/', f.año)

if f.año < … then
  diff := DiferenciaFechas(actual, f);
end;

case f.mes of
 …
end;
```

Métodos de acceso (*getters*)

```
// Fecha.java
public class Fecha {

   …
   public int getDia() {
      return dia;
   }

   public int getMes() {
      return mes;
   }

   public int getAño() {
      return año;
   }
}
```

Métodos de modificación (*setters*)

```
// Fecha.java
public class Fecha {

   …
   public void setDia(int nuevoDia) {
      dia = nuevoDia;
   }

   public void setMes(int nuevoMes) {
      mes = nuevoMes;
   }

   public void setAño(int nuevoAño) {
      año = nuevoAño;
   }
}
```

Métodos de modificación (*setters*)

```
// Test.java
public class Test {
  public static void main(String[] args) {
    Fecha f;
    f = new Fecha();
    f.setDia(12);
    f.setMes(5);
    f.setAño(2012);
    System.out.print("Hoy es día: ");
    f.imprimir();
  }
}
```

Ventaja adicional: permite comprobación de errores.

Contenidos

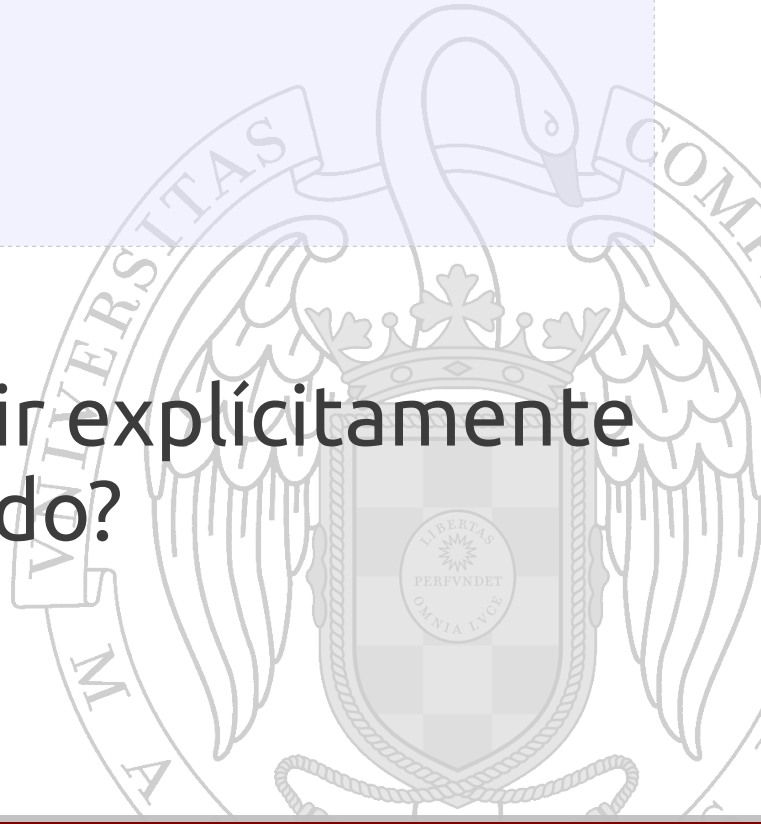
- Clases y objetos. Atributos.
- Métodos.
- Modificadores de acceso (`public`/`private`)
- Constructores.
- Igualdad de objetos.
- Ejemplos.
- Atributos y métodos estáticos.
- Paquetes.

Modificadores de acceso

- ¡Pero esto no impide que el usuario de la clase pueda acceder a los atributos directamente!

```
Fecha f;
f = new Fecha();
f.dia = 12;
f.setMes(5);
```

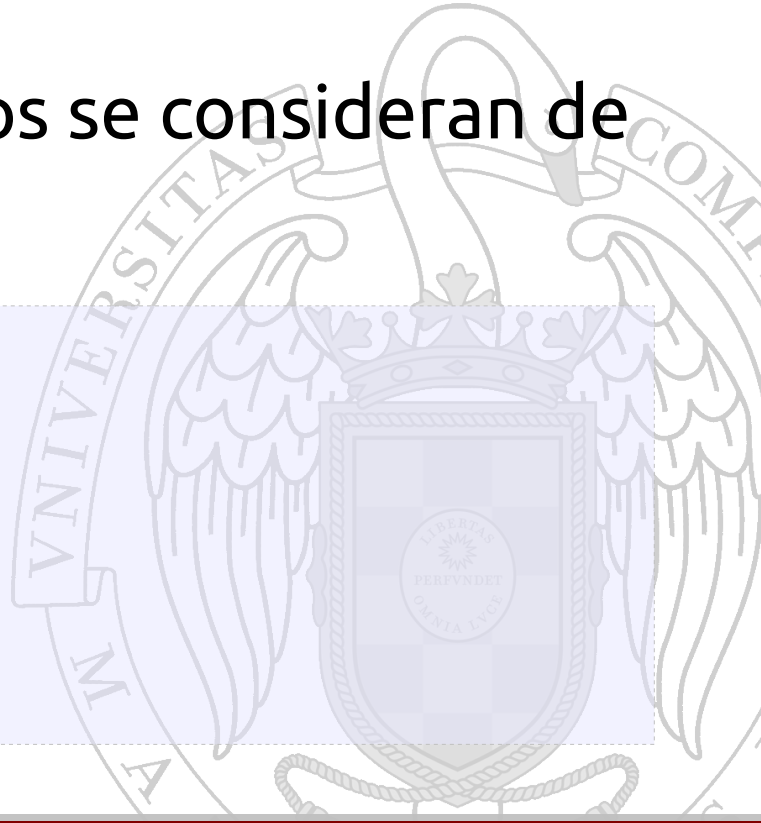
- ¿Hay alguna manera de prohibir explícitamente el acceso a un atributo o método?



Modificadores de acceso

- `public` : Puede accederse desde fuera de la clase.
- `private`: Sólo es visible desde los métodos de la clase.
  - Los atributos y métodos privados se consideran de uso interno por la clase.

```
public class Fecha {

   private int dia;
   private int mes;
   private int año;

   …

}
```



Modificadores de acceso

- `public` : Puede accederse desde fuera de la clase.
- `private`: Sólo es visible desde los métodos de la clase.
  - Los atributos y métodos privados se consideran de uso interno por la clase.

```
public class Fecha {

   private int dia;
   private int mes;
   private int año;

   …
```

Modificadores de acceso Puede ser `public`, `private`, o `protected`

Contenidos

- Clases y objetos. Atributos.
- Métodos.
- Modificadores de acceso (`public`/`private`)
- Constructores.
- Igualdad de objetos.
- Ejemplos.
- Atributos y métodos estáticos.
- Paquetes.

Constructores

- Son métodos que sirven para inicializar un objeto.
- Se caracterizan por llamarse igual que el nombre de la clase, y no tener tipo de retorno.

```
// Fecha.java
public class Fecha {

   …
   public Fecha(int nuevoDia, int nuevoMes, int nuevoAño) {
     dia = nuevoDia;
     mes = nuevoMes;
     año = nuevoAño;
   }
}
```

Constructores

- Los constructores son llamados durante la creación del objeto.

```
// Test.java
public class Test {
  public static void main(String[] args) {
    Fecha f;
    f = new Fecha(12, 5, 2012);
    System.out.print("Hoy es día: ");
    f.imprimir();
  }
}
```

Parámetros del constructor

Constructores

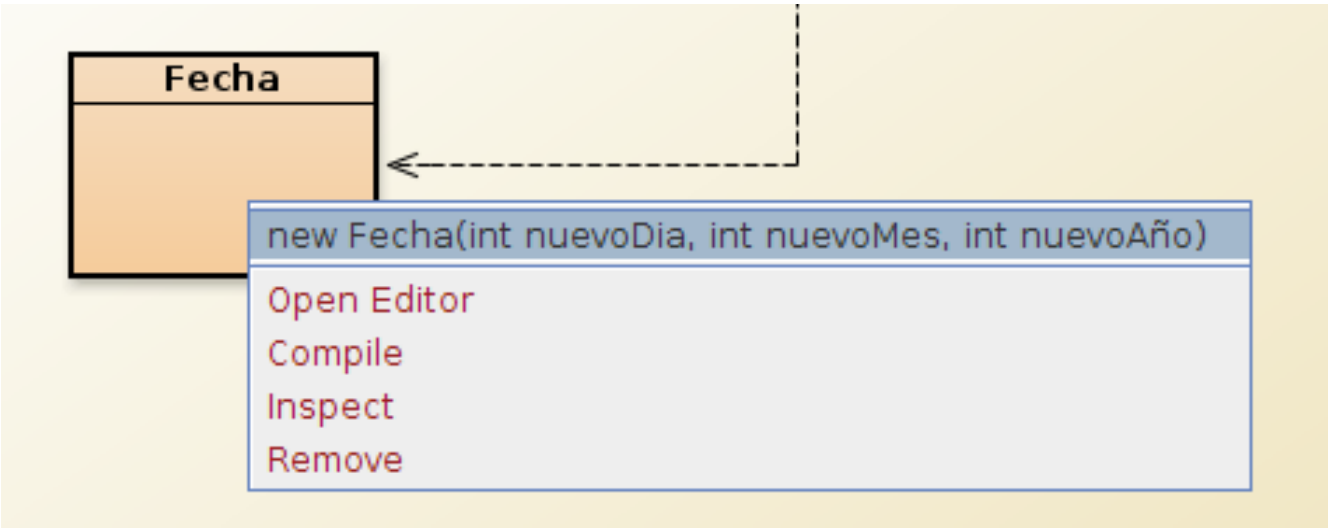
- Si una clase no tiene constructor, el compilador añade un constructor por defecto sin parámetros.
  - Inicializa todos los atributos a sus valores por defecto.

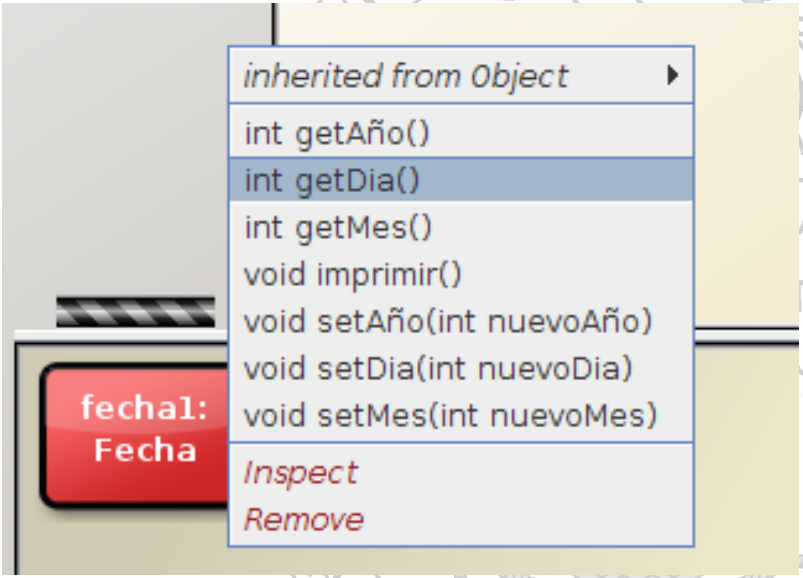
```
Fecha f = new Fecha();
```
← **Constructor por defecto**

- Cuando se define un constructor en una clase, el constructor por defecto no se añade.

```
Fecha f = new Fecha();
```
← **ERROR**

Manejo interactivo de objetos en *BlueJ*





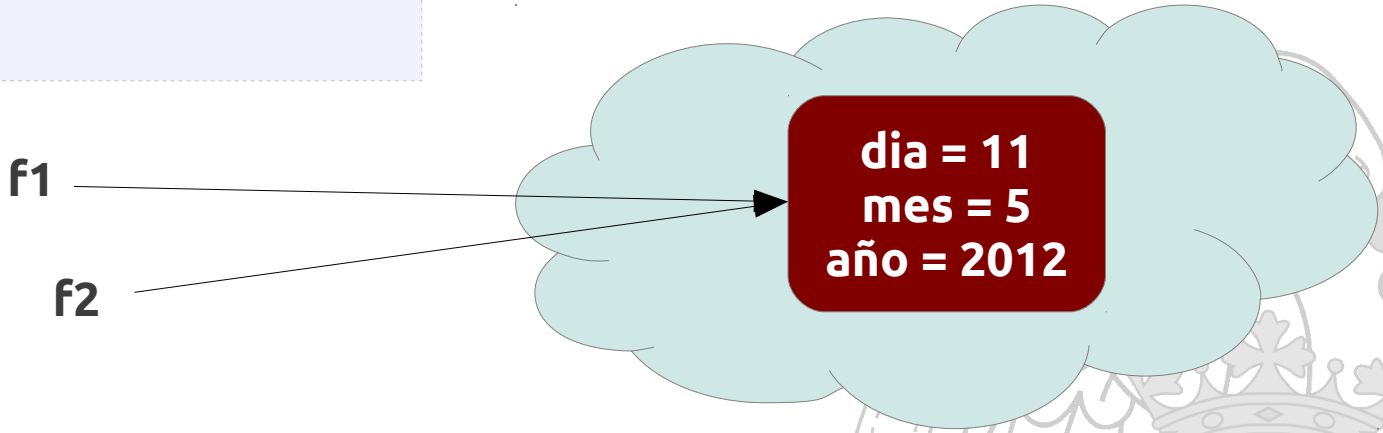
Contenidos

- Clases y objetos. Atributos.
- Métodos.
- Modificadores de acceso (`public`/`private`)
- Constructores.
- Igualdad de objetos.
- Ejemplos.
- Atributos y métodos estáticos.
- Paquetes.

Igualdad de objetos

- Los objetos se almacenan en el *heap*.
- Las variables son referencias a esos objetos.

```
Fecha f1 = new Fecha(11, 5, 2012);
Fecha f2 = f1;
```

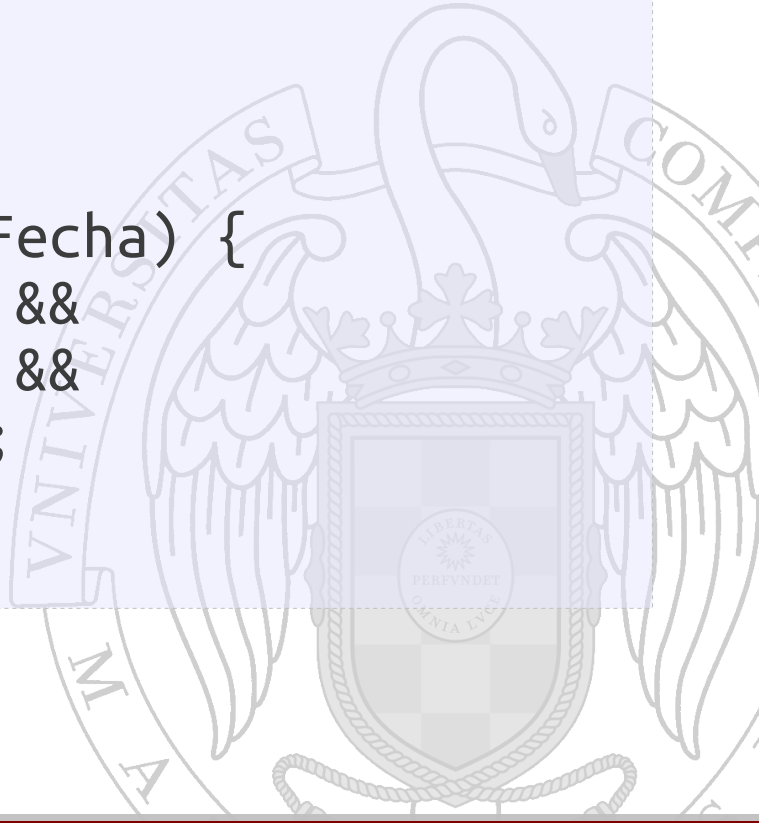


- La igualdad entre dos variables de tipo objeto (==) es cierta sólo si apuntan al mismo objeto.

Igualdad de objetos

- Si se quiere implementar otro tipo de igualdad, ha de hacerse mediante un método de la clase.

```
// Fecha.java
public class Fecha {

   …
   public boolean igualA(Fecha otraFecha) {
      return (dia == otraFecha.dia) &&
             (mes == otraFecha.mes) &&
             (año == otraFecha.año);
   }
}
```



Igualdad de objetos

- Si se quiere implementar otro tipo de igualdad, ha de hacerse mediante un método de la clase.

```
// Test.java

…

f1 = new Fecha(14, 2, 2000);
f2 = new Fecha(14, 2, 2000);
if (f1.igualA(f2)) {
    System.out.println("Son iguales");
}
```

Contenidos

- Clases y objetos. Atributos.
- Métodos.
- Modificadores de acceso (`public`/`private`)
- Constructores.
- Igualdad de objetos.
- Ejemplos.
- Atributos y métodos estáticos.
- Paquetes.

Ejemplo: clase Punto

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

    public Punto(int x, int y)  {
        this.x = x;
        this.y = y;
    }

    public int getX() { return x; }
    public int getY() { return y; }

}
```

Ejemplo: clase Punto

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

    public Punto(int x, int y)  {
        this.x = x;
        this.y = y;
    }

    public int getX() { return this.x; }
    public int getY() { return this.y; }

}
```

this = objeto que está siendo construido

Atributos de la clase

this = objeto que recibe el mensaje

Ejemplo: clase Rectangulo

```
public class Rectangulo
{
    private Punto posición; // Posición de la esquina superior izquierda
    private double ancho, alto;

    public Rectangulo(Punto posicion, double ancho, double alto) {
       this.posicion = posicion;
       this.ancho = ancho;
       this.alto = alto;
    }

    public Rectangulo(Punto esqSupIzqda, Punto esqInfDcha) {
       this posicion = esqSupIzqda;
       this.ancho = esqInfDcha.getX() - esqSupIzqda.getX();
       this.alto = esqInfDcha.getY() - esqSupIzqda.getY();
    }

    …
```

- Sobrecarga de constructores: puedo tener varios constructores por método, siempre que no coincidan en número y tipo de parámetros.
- También se aplica a los métodos.

Ejemplo: clase Rectangulo

```
public class Rectangulo
{

    …
    public double area() {
       return ancho * alto;
    }

    public double perimetro() {
       return 2*alto + 2*ancho;
    }


    public void dibujar(Ventana v) {
       v.dibujarRectangulo(origen.getX(), origen.getY(),
                           ancho, alto);
    }
}
```

Ejemplo: clase Circulo

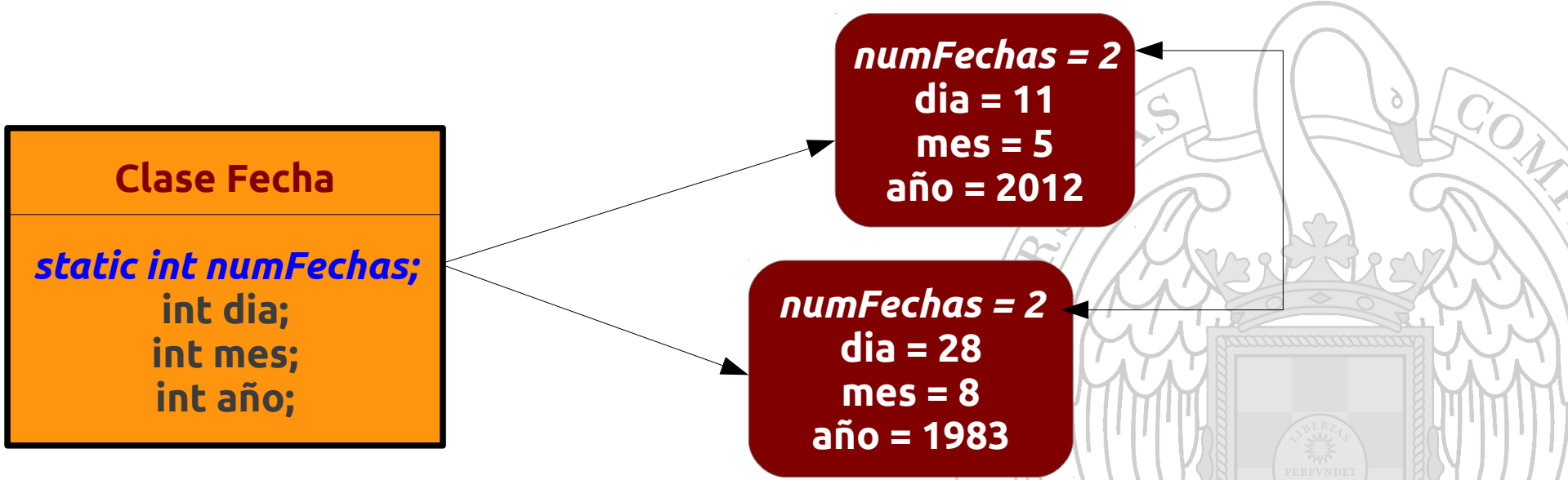
```
public class Circulo
{
    private Punto centro;
    private double radio;

    public Circulo(Punto centro, double radio) { … }

    public double getRadio() { … }
    public Punto getCentro() { … }

    public double area() { … }
    public double perimetro() { … }
    public void dibujar(Ventana v) { … }
}
```

Contenidos

- Clases y objetos. Atributos.
- Métodos.
- Modificadores de acceso (`public`/`private`)
- Constructores.
- Igualdad de objetos.
- Ejemplos.
- Atributos y métodos estáticos.
- Paquetes.

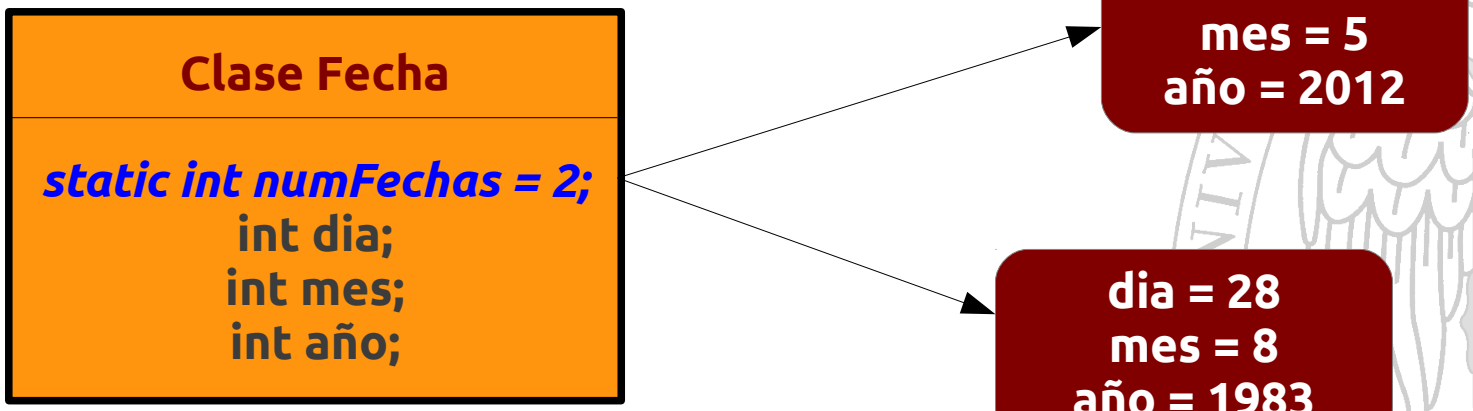
Atributos y métodos estáticos

- Un miembro estático de una clase es un atributo o un método que es compartido por todos los objetos de dicha clase.



Atributos y métodos estáticos

- Una atributo estático representa información a nivel de clase.
- Los métodos estáticos realizan operaciones que no van asociados a ninguna instancia particular de la clase.



Atributos y métodos estáticos

```
// Empleado.java
public class Empleado {
    private String nombre;
    private int DNI;
    private Fecha fechaNacimiento;
    private int numeroEmpleado;

    private static int contadorEmpleados = 0;

    public Empleado(String nombre, int DNI, Fecha fechaNacimiento) {
      this.nombre = nombre;
      this.DNI = DNI;
      this.fechaNacimiento = fechaNacimiento;
      numeroEmpleado = contadorEmpleados;
      contadorEmpleados++;
    }

    …

}
```

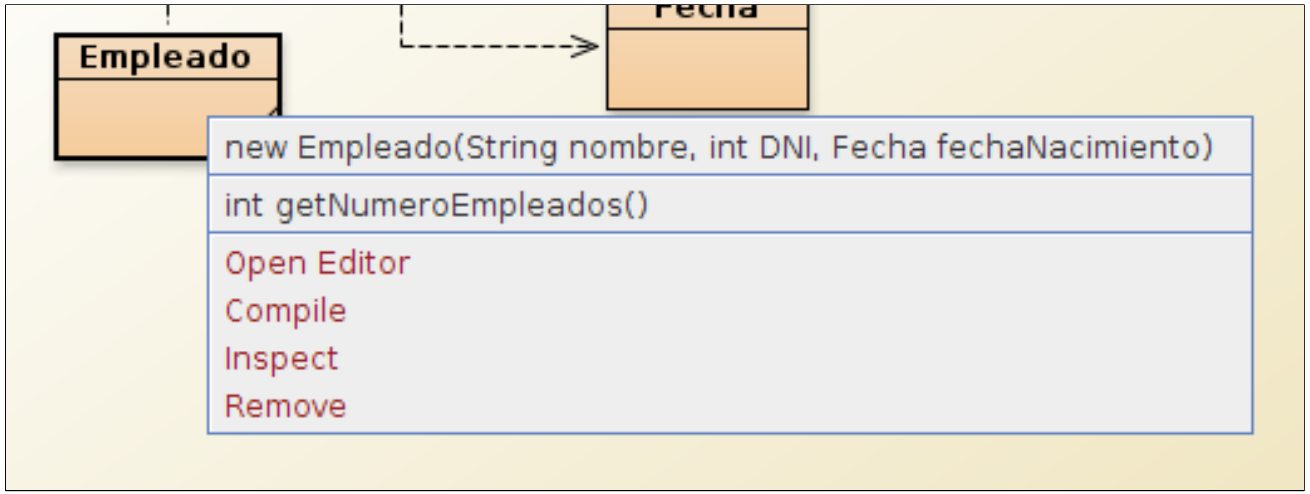
Atributos y métodos estáticos

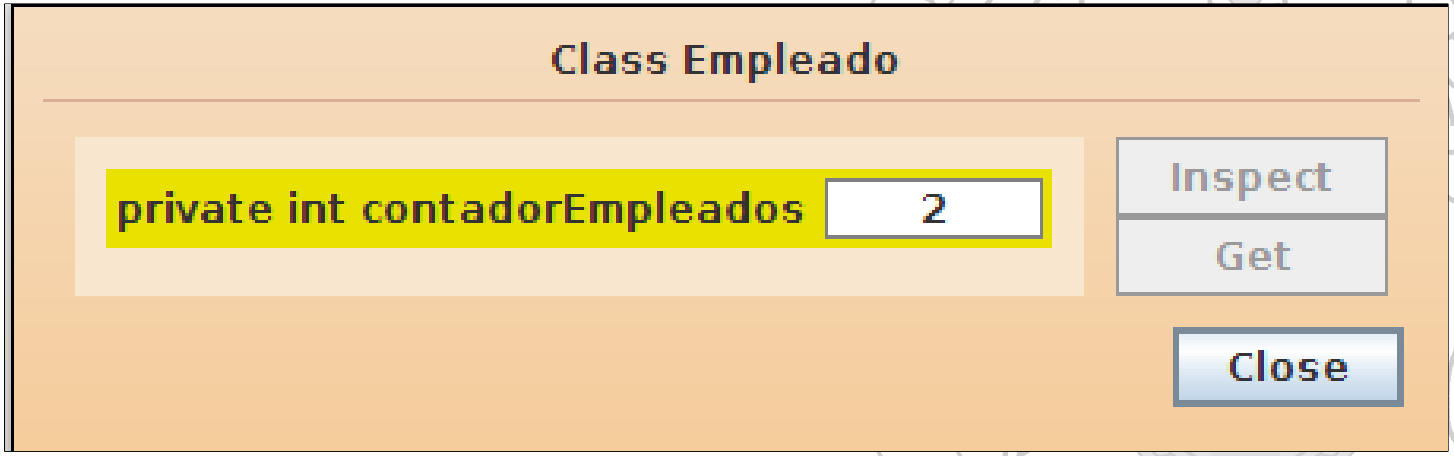
```
// Empleado.java
public class Empleado {

   …
   public static int getNumeroEmpleados() {
      return contadorEmpleados;
   }
}
```

```
// Test.java
public class Test {

   public static void main (String[] args) {
     Empleado e1 = new Empleado("Luis Fernández", 46852391,
                                new Fecha(12, 4, 1979));
     Empleado e2 = new Empleado("Javier Barajas", 23771829,
                                new Fecha(21, 11, 1980));

     System.out.prinln(Empleado.getNumeroEmpleados());
   }
}
```

Nombre de la clase

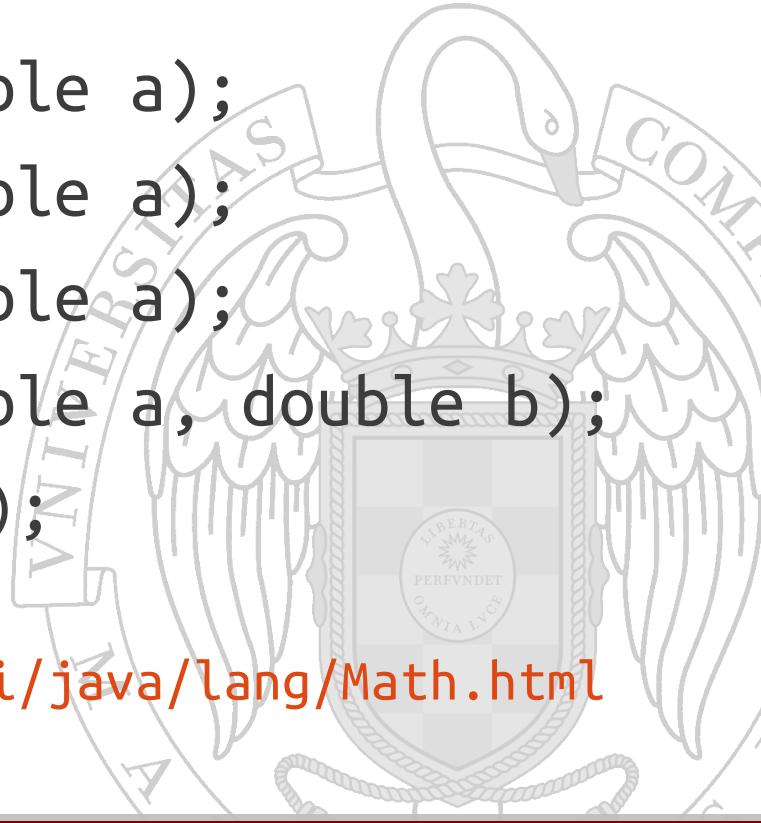
Atributos y métodos estáticos





La clase Math de Java

- Es una clase formada exclusivamente por atributos y métodos estáticos.
  - `public static double E;`
  - `public static double PI;`
  - `public static double abs(double a);`
  - `public static double sin(double a);`
  - `public static double exp(double a);`
  - `public static double pow(double a, double b);`
  - `public static double random();`
- Más información:
  http://docs.oracle.com/javase/6/docs/api/java/lang/Math.html

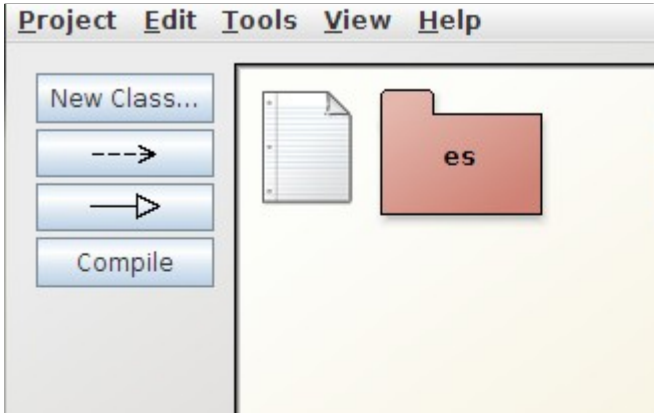


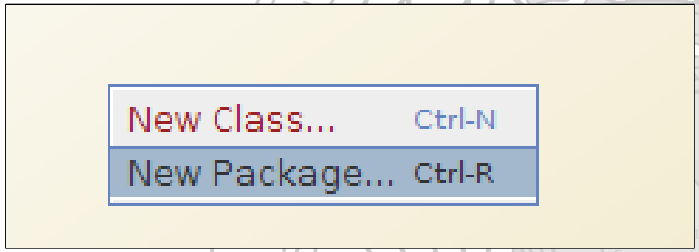
Contenidos

- Clases y objetos. Atributos.
- Métodos.
- Modificadores de acceso (`public`/`private`)
- Constructores.
- Igualdad de objetos.
- Ejemplos.
- Atributos y métodos estáticos.
- Paquetes.

Paquetes

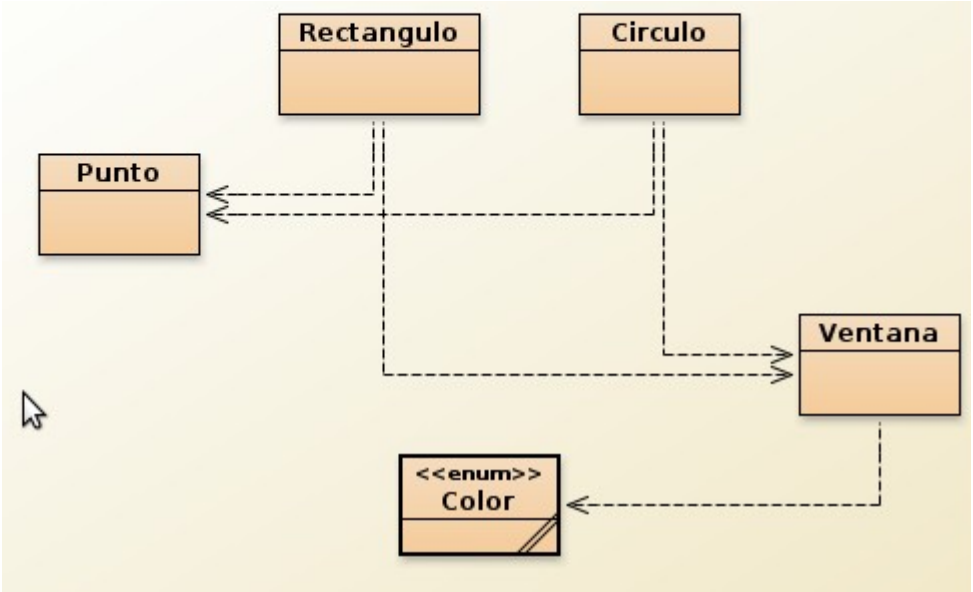
- Un paquete en Java es una agrupación de clases que comparten una temática o funcionalidad similar.
- Sirven para evitar conflictos de nombres entre clases creadas por distintas personas o empresas.





Paquetes

- Una clase puede acceder a todas las clases públicas que están en su mismo paquete, sin necesidad de indicar el nombre de dicho paquete.




```
// Circulo.java
public class Circulo {
  private Punto centro;
  …

}
```

Paquetes

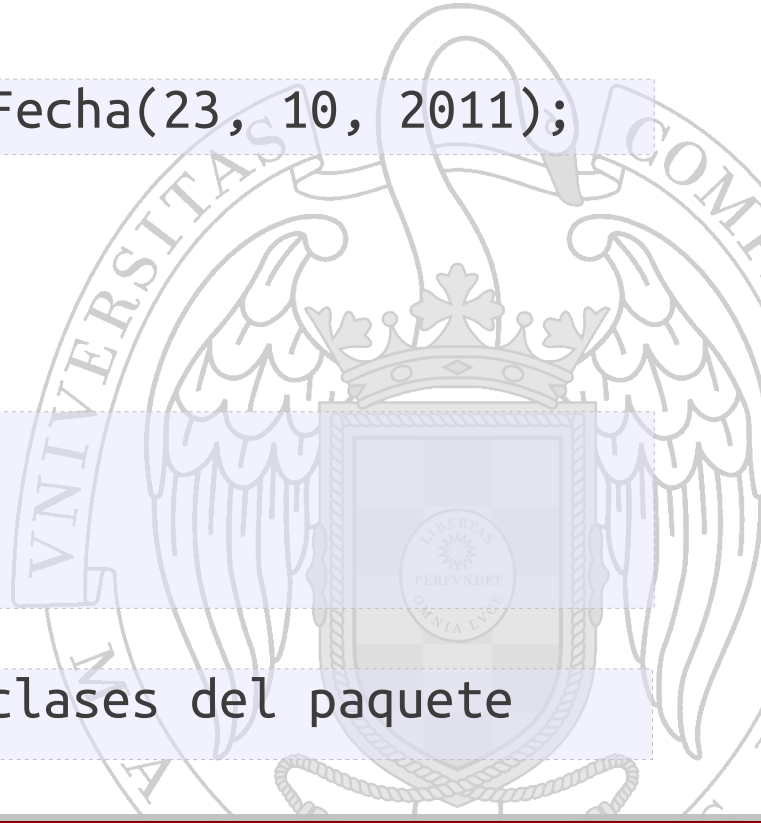
- Si una clase quiere acceder a otras que no están en su mismo paquete, hay dos opciones:
  - Indicar explícitamente el nombre del paquete en el que se encuentran.

```
es.ucm.mat.Fecha fecha = new es.ucm.mat.Fecha(23, 10, 2011);
```

  - Utilizar la directiva `import`.

```
import es.ucm.mat.Fecha;
…
   Fecha fecha = new Fecha(23, 10, 2011);
```

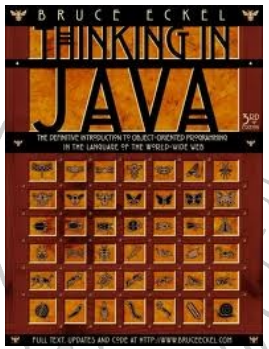
```
import es.ucm.mat.*; // Importa todas las clases del paquete
```



Referencias

- P. Deitel, H. Deitel
  Java. How to Program (9th Edition)
  Caps. 3, 6, 8.

- B. Eckel
  Thinking in Java (3rd Edition)
  Caps. 2, 4, 5, 6.



- Documentación de librerías de Java
  http://docs.oracle.com/javase/6/docs/api/
- Cómo documentar las clases (Javadoc):
  Deitel & Deitel: Apéndice M.
  B.Eckel: Cap. 2.