



# AspectJ

Juan Rodriguez Duran  
Johann Triana Olaya  
Jonatan Parra Toro





# Tabla de contenido

1. Historia y descripción
2. Ventajas y desventajas
3. Instalación de AspectJ en Eclipse
4. Sintaxis
5. Repaso Java
6. Caso de estudio



A decorative graphic on the left side of the slide. It features a large central hexagon with a blue-to-teal gradient, containing the white number '1'. Surrounding this central hexagon are several smaller hexagons of varying shades of blue and teal. Some of these smaller hexagons contain white icons: a lightbulb, a thumbs-up, a smartphone, a magnifying glass, and a gear. There is also a network-like icon with a central node and radiating lines, and a speech bubble icon.

1

# Historia

Breve resumen

# AspectJ Historia



Gregor Kiczales.



Extensión de java

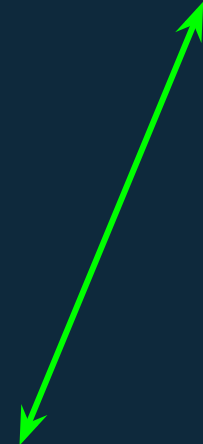
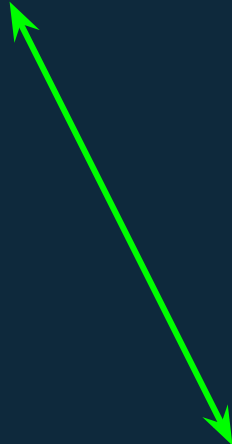
# Compatibilidad de AspectJ


Compatibilidad  
base

Compatibilidad de  
Plataforma




Compatibilidad de  
programación





Extiende Java para soportar el  
manejo de **aspectos** agregando a la  
semántica de Java cuatro entidades  
principales.



Puntos de enlace  
(Joint Point)



Consejos  
(Advice)



Puntos de corte  
(Pointcut)



Introducciones  
(introduction)





No es una traducción  
purista de la POA



No representa un intento agresivo  
para explorar las posibilidades del  
lenguaje

# Metas de AspectJ



La intención de AspectJ es ser un LOA  
práctico, que provea un conjunto sólido y  
maduro de características orientadas a  
aspectos, compatible con Java para aprovechar  
su popularidad.



A decorative pattern of hexagons in various shades of blue and cyan. Some hexagons contain white icons: a lightbulb, a thumbs-up, a smartphone, a magnifying glass, a gear, and a speech bubble. A network diagram with a central node and five peripheral nodes is also visible.

2

# Ventajas y Desventajas

AspectJ





## Ventajas

- Facilita/mejora la modularidad de los desarrollos de software.
- El código es organizado y comprimido
- Es reutilizable
- Pérmite modelar referencias transversales, extiende las capacidades de POO

## Desventajas

- Puede introducir nuevos errores y fallas de seguridad si no se implementa adecuadamente
- El paradigma puede emplearse mal quitando y/o delegando responsabilidades a ciertas clases
- Los Aspectos pueden ser mal utilizados.



A decorative graphic on the left side of the slide consists of a cluster of hexagons in various shades of blue and cyan. Some hexagons contain white icons: a lightbulb, a thumbs-up, a smartphone, a magnifying glass, and a gear. Other hexagons are empty or contain smaller, fainter icons like a network node and a speech bubble.

2.1


# Características

Del paradigma



# Características Principales

```
public class BankAccount {  
  
    public void withdraw (double amount) {  
  
        long startTime = System.currentTimeMillis();  
        try {  
            // Actual method body... }  
        finally {  
            long endTime = System.currentTimeMillis() -  
                           startTime;  
            System.out.println("withdraw took: " + endTime);  
        }  
    }  
}
```





# Características Principales

Un aspecto es una preocupación común a varias clases de una aplicación. El aspecto es “casi siempre” ortogonal a lo que la clase representa. Por ejemplo:

- Logger,
- requerimientos no funcionales,
- testing
- validación de contratos (parámetros)
- etc.





# AOP vs POO

## ¿Como hacer un logger?

Y si usamos...

- Un wrapper
- Una interfaz
- Herencia
- Listeners
- Reflexión de las clases

Resultado

- Código muy complejo en la llamada a los métodos
- Posibilidad de olvidar la implementación o llamar al super
- Olvidar el notify a los listeners
- Malas prácticas en POO y mayor complejidad



A decorative graphic on the left side of the slide. It features a large central hexagon with a blue-to-cyan gradient, containing the number '3'. Surrounding this central hexagon are several smaller hexagons of varying shades of blue and cyan. Some of these smaller hexagons contain white icons: a lightbulb, a thumbs-up, a smartphone, a magnifying glass, and a gear. There is also a network-like icon with a central node and radiating lines, and a speech bubble icon.

3

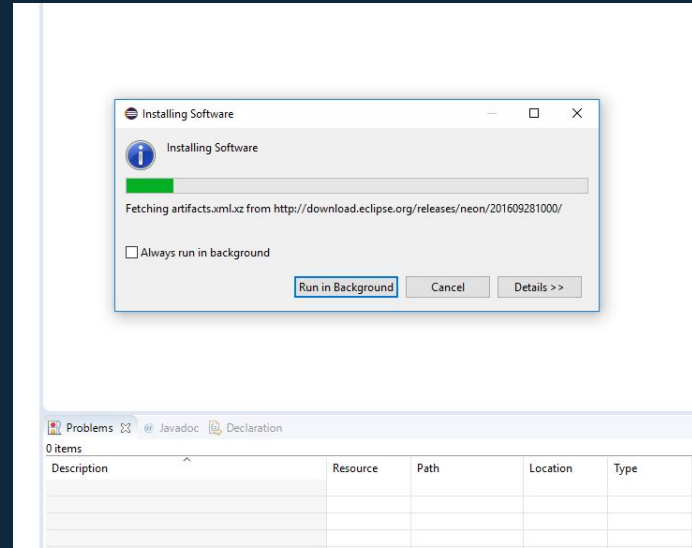
# Instalación en Eclipse

Paso a paso

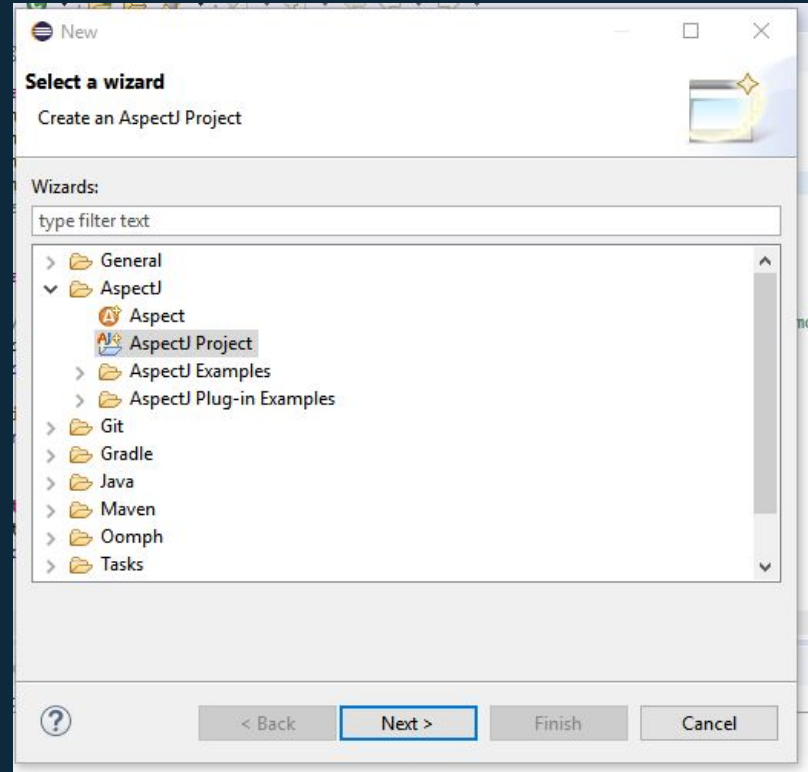
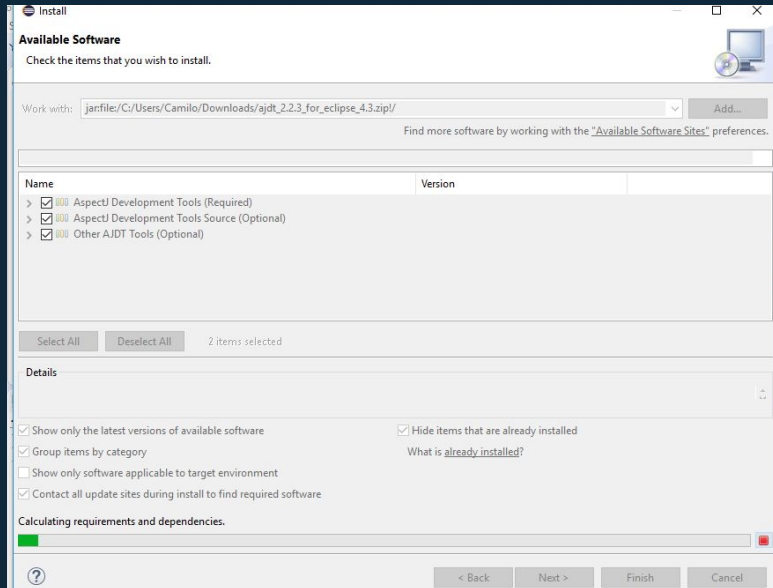
# Instalación de aspectj en eclipse

Try adding this site:

<http://download.eclipse.org/tools/ajdt/46/dev/update>



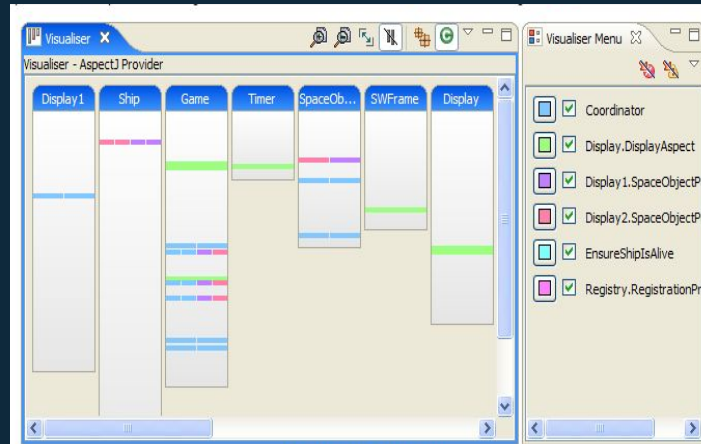
# Instalación de aspectj en eclipse





# Instalación de aspectj en eclipse

```
package bussinesLogic;  
  
public aspect MemoryAspect {  
  
    private long prev;  
  
    pointcut memoryUse() :call (public void JavaMethods.*(..)) ;  
  
    pointcut sysTime() :call (static * JavaMethods.*(..)) ;
```



A decorative graphic on the left side of the slide. It features a large central hexagon with a blue-to-cyan gradient, containing the white number '4'. Surrounding this central hexagon are several smaller hexagons of varying shades of blue and cyan. Some of these smaller hexagons contain white icons: a lightbulb, a thumbs-up, a smartphone, a magnifying glass, and a gear. There is also a network-like icon with a central node and radiating lines, and a speech bubble icon.

4

# Sintaxis

Básica de los comandos




# Creación de un Aspecto

```
[privileged] [Modifiers] aspect Id [extends Type] [implements TypeList]
```

```
public aspect Mi_aspecto {  
    // Codigo  
}
```

Privileged: Un aspecto con este modificador puede acceder Incluso a los métodos privados de las clases.



# Creación de un Aspecto

- Como un archivo

Paquete: `mi_paquete`  
Aspecto: `Mi_aspecto.aj`



`mi_paquete`



`Mi_aspecto.aj`

- Como parte de una clase.

```
public class MyClass{  
...  
    public aspect Mi_aspecto{  
        ...  
    }  
}
```



`MyClass.java`

- Extends, Implements, Abstract...

# Joint point

Es el punto de ejecución dentro del sistema donde un aspecto puede ser conectado

```
public class Compra {  
    private int cantidad;  
    public void buscarProducto(){  
        System.out.println("Se encontró su producto en "  
            + "la direccion producto.html");  
    }  
    public void comprar(){  
        System.out.println("Se ha realizado su compra");  
    }  
    public void setCantidad(int cantidad){  
        this.cantidad = cantidad;  
    }  
    public static void main(String[] args) {  
        Compra compra = new Compra();  
        compra.buscarProducto();  
        compra.buscarProducto();  
  
        Compra compra2 = new Compra();  
        compra2.buscarProducto();  
        compra2.setCantidad(5);  
        compra2.comprar();  
    }  
}
```




# PointCut

call(Signature)

```
public aspect Mi_aspecto {  
    // Asocia unicamente a un metodo  
    pointcut contar_busquedas(): call( public void buscarProducto() );  
}
```


```
public aspect Mi_aspecto {  
    // Asocia a dos metodos  
    pointcut contar_busquedas_y_compras():  
        call( public void buscarProducto() ) || call(public void comprar());  
}
```





# PointCut Pattern


<b>* method(..)</b>	Match method que retorna cualquier cosa y recibe cualquier cantidad de parámetros
<b>* method (*,*,*)</b>	Retorna cualquier cosa y recibe 3 parámetros cualesquiera
<b>void *(String)</b>	Todos los métodos que no retornan nada y reciben un parámetro String
<b>String method(String, int)</b>	Method que retorna string y recibe como parámetro un String y un entero
<b>* * (..)</b>	Todos los métodos





# Tipos de PointCut

Puntos de corte	Descripción
<code>call(method)</code>	Captura la llamada a un método o constructor de una clase
<code>execution(method)</code>	Captura la ejecución de un método o constructor.







4.1

# Muestra de ejemplo

Ejemplo de uso en eclipse

# PointCut: execution vs call

```
public class Humano{  
    public void estirarse();  
}
```

```
public class Estudiante extends Humano{  
    @Override  
    public void estirarse(){  
        Log.descansar("ñaaam");  
    }  
}
```

```
public class HumanoMain{  
    public static void main(String[] args) {  
        Humano juan = new Estudiante();  
        juan.estirarse();  
  
        Estudiante jonatan = new Estudiante();  
        jonatan.estirarse();  
    }  
}
```



# PointCut: execution vs call

```
public aspect verPereza{  
    pointcut seEstiro(): call (public void Estudiante.descansar());  
  
    before() : call(){  
        System.out.println("Hola peresozo");  
    }  
}
```



```
public aspect verPereza{
    pointcut seEstiro(): call(public void Estudiante.estirarse()
        && withincode(descansar()));
}
```

withincode(firma)


Captura la ejecución dentro del método o constructor indicado. La firma puede ser una clase o un paquete, en este caso capturará todos los métodos asociados a la firma.

```
public class Humano{
    public void estirarse(){
        System.out.println("aaaahm");
    }
}


public class Estudiante extends Humano{
    @Override
    public void estirarse(){
        super.estirarse();
        System.out.println("ñaaam");
    }
}
```

← Punto de captura





this(objeto/clase) target(objeto/clase)	This captura todos los Join Point que hagan interacción con el objeto/clase especificados Target captura todos los Join Point respecto a métodos o operaciones con atributos del objeto/clase especificados.
args(firma)	Captura todos los puntos de enlace donde los argumentos son instancias de una clase que coincide con el Patrón de Clase o con la clase del identificador
get(clase)	Captura la lectura de una variable de una clase
set(clase)	Captura la escritura de una variable de una clase
handler(Exception)	Captura el manejador de una excepción



# Advice

Código que se ejecutará cuando se llegue al pointcut este se puede definir de tres diferentes maneras:


- before ()
- after ()
- around ()

```
public aspect ExceptionGotchaAspect {  
    before (Exception e): handler(Exception+) {  
        System.err.println("Caught by aspect: " + e.toString());  
        e.printStackTrace();  
    }  
}
```



## Implementacion de un advice after

```
public aspect Mi_aspecto {  
    pointcut nombre_point(): call (public void setCantidad(int));  
  
    after(): nombre_point(){  
        System.out.println("Este codigo se ejecutara"+  
            "al salir del metodo");  
    }  
}
```





# Ejemplo básico

Una tienda online necesita llevar el registro de cuantas veces se ha llamado al método buscarProducto() sin importar el objeto que lo haga.

Utilizando un aspecto y sin modificar el código actual imprima en pantalla la cantidad de veces que se ha llamado este método.

```
public class Compra {  
    public void buscarProducto(){  
        System.out.println("Se encontró su producto en "  
            + "la direccion producto.html");  
    }  
    public static void main(String[] args) {  
        Compra compra = new Compra();  
        compra.buscarProducto();  
        compra.buscarProducto();  
        Compra compra2 = new Compra();  
        compra2.buscarProducto();  
        compra2.buscarProducto();  
    }  
}
```



## Aspecto que cuenta la cantidad de veces que se llamó al método

```
public aspect Mi_aspecto {  
    private int cant_busquedas = 0;  
    pointcut contar(): call( public void buscarProducto() );  
  
    after() : contar() {  
        cant_busquedas++;  
        System.out.println("Se ha realizado la accion de busqueda "  
            + cant_busquedas + " veces");  
    }  
}
```



## Salida en consola

```
Se encontró su producto en la direccion producto.html
Se ha realizado la accion de busqueda 1 veces
Se encontró su producto en la direccion producto.html
Se ha realizado la accion de busqueda 2 veces
Se encontró su producto en la direccion producto.html
Se ha realizado la accion de busqueda 3 veces
Se encontró su producto en la direccion producto.html
Se ha realizado la accion de busqueda 4 veces
```



A decorative graphic on the left side of the slide. It features a large central hexagon with a blue-to-cyan gradient containing the number '5'. Surrounding this central hexagon are several smaller hexagons of varying shades of blue and cyan. Some of these smaller hexagons contain white icons: a lightbulb, a thumbs-up, a smartphone, a magnifying glass, and a gear. There is also a network-like icon with a central node and radiating lines, and a speech bubble icon.

5

# Repaso Java

Ejemplos de uso



# Java 8 y anteriores

## Bloques estaticos

```
static {  
    primes = new ArrayList<>();  
    lastKnow = 2L;  
    System.out.println("call static block " + 10e-5);  
}
```



# Java 8 y anteriores

## Streams

```
private static void calcPrimes(long n) {  
    LongStream.iterate(lastKnow, i -> i + 1).filter(i -> {  
        for (Long prime : primes){  
            if (i % prime == 0){  
                return false;  
            }  
        }  
        return true;  
    })).limit(n).forEach(primes::add);  
}
```



# Java 8 y anteriores

## Lambda y map reduce paralelo

```
JavaMethods jm = new JavaMethods();
jm.hello();
jm.add("hola esto es una prueba hola");
jm.add("las pruebas que prueba la coherencia de un texto");
jm.add("el la lo son articulos de prueba para este conteo");
jm.add("sera una prueba de los numeros primos usados");
Map<String, Long> result = jm.getLines().stream().map(e -> {
    String[] split = e.split("\\s");
    Map<String, Long> map = new HashMap<>();
    for (String cad : split) {
        if (map.containsKey(cad)) {
            map.put(cad, map.get(cad) + 1L);
        } else {
            map.put(cad, 1L);
        }
    }
    return map;
}).reduce((m1, m2) -> JavaMethods.fussion(m1, m2)).get();
result.forEach((k, v) -> System.out.println("key " + k + " : " + v));
```

```
Map<String, Long> result = jm.getLines().parallelStream().map(e -> {
```

A decorative graphic on the left side of the slide. It features a large central hexagon with a blue-to-teal gradient, containing the white number '6'. Surrounding this central hexagon are several smaller hexagons of varying shades of blue and teal. Some of these smaller hexagons contain white icons: a lightbulb, a thumbs-up, a smartphone, a magnifying glass, and a gear. There is also a network-like icon with a central node and radiating lines, and a speech bubble icon.

6

# Caso de estudio

Ejemplo de uso



# Google Analytics

Mide la usabilidad de una aplicación por el número de clicks y enlaces seguidos, además de otros datos







# Características a Evaluar

Número de llamadas a métodos

Memoria asignada por llamada

Clicks realizados sobre cada elemento

```
/* Total amount of free memory available to the JVM */
System.out.println("Free memory (bytes): " +
Runtime.getRuntime().freeMemory());

/* This will return Long.MAX_VALUE if there is no preset limit */
long maxMemory = Runtime.getRuntime().maxMemory();
/* Maximum amount of memory the JVM will attempt to use */
System.out.println("Maximum memory (bytes): " +
(maxMemory == Long.MAX_VALUE ? "no limit" : maxMemory));

/* Total memory currently in use by the JVM */
System.out.println("Total memory (bytes): " +
Runtime.getRuntime().totalMemory());
```



6.1

# Muestra de ejemplo

Ejemplo de uso en eclipse



Muchas  
Gracias  
¿Preguntas?

