Programación Orientada a Aspectos

- Danny Julca
- Bruno Bedon
- Kevin Molina
- Miguel Mejia

Introducción

La Programación Orientada a Aspectos es un paradigma de programación, se basa en subject-oriented programming, metaobject protocols y adaptative programming. Fue desarrollado por Gregor Kiczales y sus colegas de Xerox PARC los cuales tuvieron un conceptos explícito de AOP el cual tuvo su primera implementación denominada AspectJ que es una extensión para Java en el 2001. El equipo de desarrollo de IBM diseñaron un lenguaje Hyper/J a su vez en el 2001.

Objetivos

El principal objetivo de la POA es la separación de las funcionalidades dentro del sistema:

- Por un lado funcionalidades comunes utilizadas a lo largo de la aplicación.
- Por otro lado, las funcionalidades propias de cada módulo.

Cada funcionalidad común se encapsulará en una entidad.

Conceptos básicos

Aspect

Es la combinación de el pointcut y un advice.

JoinPoint

Un conjunto de joinPoints se denomina pointcut. Es una especificación de donde, en el programa, el aspecto debe ser ejecutado.

Advice

Es un código adicional que quieres añadir al modelo, es decir, código adicional (puede ser un log) que se aplica con una acción.

Conceptos básicos

PointCut

Es una determinada línea en el programa donde sucede una acción.

Introduction

Permite añadir métodos o atributos a clases ya existentes.

Target

Es el objeto o instancia de una clase que funciona como advice.

Weaving

Toma las instrucciones de las clases y aspectos y crean nuevas clases con los aspectos definidos, las instrucciones son conocidas como advice, usa los pointcuts y joinpoints.

Diferencia con otros paradigmas

Programación Orientada a Objetos

En la programación orientada a objetos los sistemas se modelan como un conjunto de objetos que interactúan entre sí, sin embargo, falla al modelar los conceptos que se entrecruzan.

Entonces, la diferencia radica en que mientras la programación orientada a aspectos se enfoca en los conceptos que se entrecruzan, la programación orientada a objetos se enfoca en los conceptos comunes.

Herramientas de desarrollo

- **AspectC++** es un compilador que permite desarrollar aspectos en C++.
- **AspectJ** es una extensión Java del proyecto Eclipse para ayudar en el desarrollo orientado a aspectos.
- **Aspect** es un módulo Perl disponible en CPAN para la Programación Orientada a Aspectos (en inglés).
- **PHP-AOP (AOP.io)** es una lib que proporciona todo el paradigma de la POA en PHP.
- **phpAspect** es una extensión PHP para implementar el paradigma de la POA, que, mediante árboles de decisión XML, realiza el weaving del software para ser ejecutado como PHP estándar.

Herramientas de desarrollo

- **FLOW3** es un framework MVC de PHP incluye un módulo para poder realizar Programación orientada a Aspectos en nuevos desarrollos.
- **AOP** con SpringFramework 2.5 es un Framework de Java que permite programar en el paradigma de Aspectos utilizando Anotación Java.
- **Aspyct AOP** es un módulo de Python que permite incluir Programación orientada a Aspectos a programas ya existentes escritos en Python o a nuevos desarrollos.

JoinPoint

- Un punto de unión es un punto bien definido en el flujo del programa:
  - Ejecutar algún código (“advice”) cada vez que se alcanza un JoinPoint.
  - Evitar saturar el código con indicadores explícitos diciendo “Este es un punto de unión”.
  - AspectJ proporciona una sintaxis para indicar que estos puntos se unen “desde fuera” al código real.
- Un JoinPoint es un punto en el flujo del programa donde “sucede algo”
- Ejemplos
  - Llamada a un método
  - Excepción
  - Acceso a una variable
  - Instancias de un objeto
  - Referencia a un objeto

PointCut

- Las definiciones de PointCut consisten en un lado izquierdo y un lado derecho, separados por dos puntos.
- El lado izquierdo está formado por el nombre del PointCut y sus parámetros (es decir, los datos disponibles cuando los eventos ocurra)
- El lado derecho consiste en el propio PointCut.

```
pointcut callSayHello () :
         call (* HelloAspectJDemo.sayHello () ) ;
```

  - El nombre de este PointCut es ’callSayHello’.

PointCut

- El pointcut no tiene parámetros.
- El pointcut en sí mismo es ’(* HelloAspectJDemo.sayHello ())’.
- El pointcut se refiere a cualquier momento en que se realiza la llamada al método ’HelloAspectJDemo.sayHello()’.

Advices

- Before

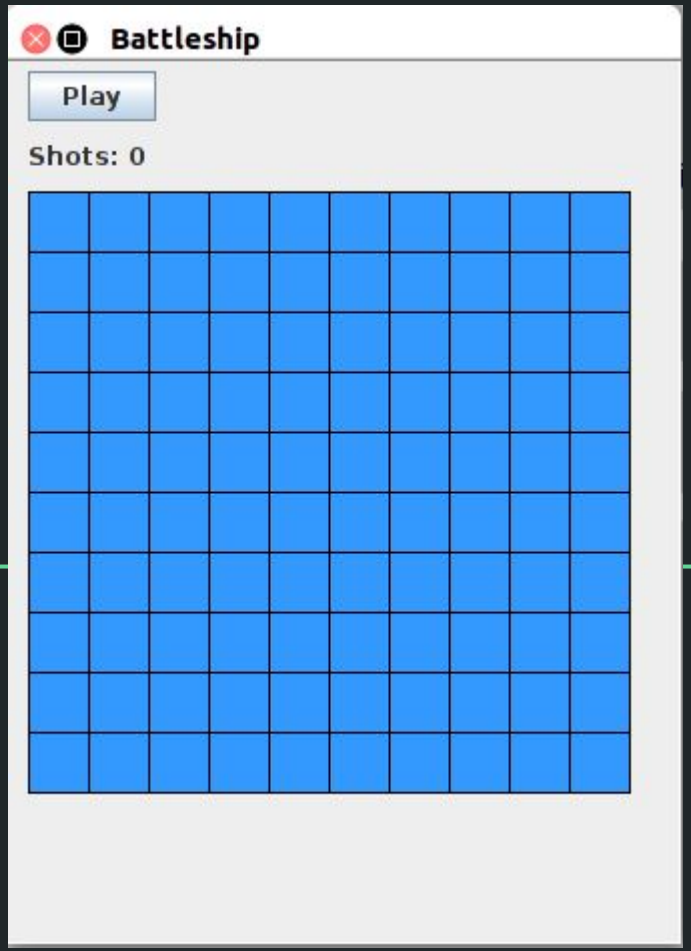
```
before () : callSayHello () {
     System.out.println(" Before execution ") ;
}
```

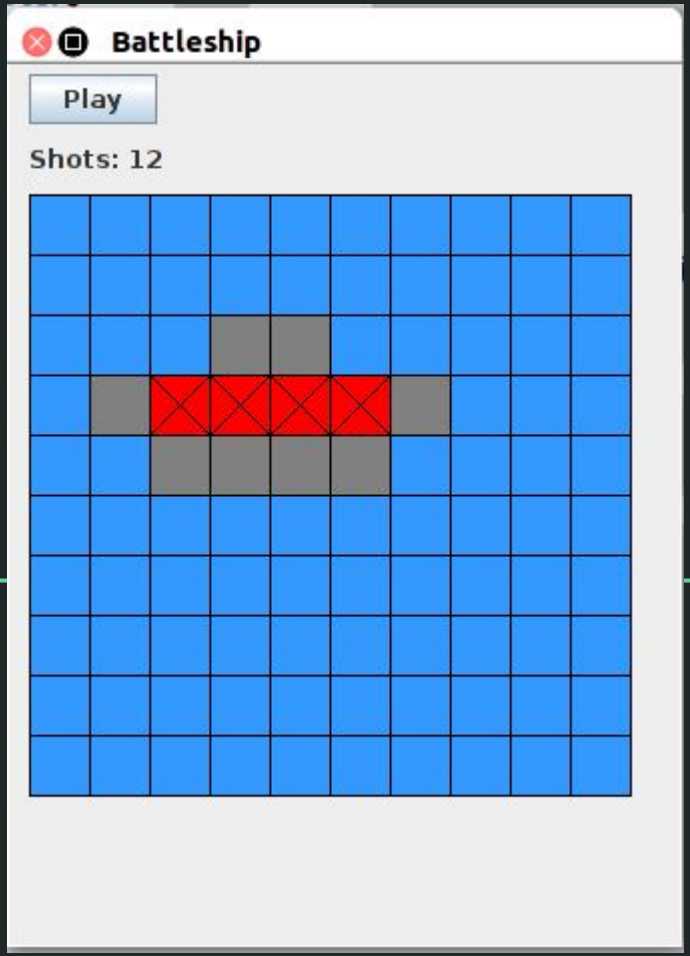
- After

```
after () : callSayHello () {
     System.out.println(" After execution ") ;
}
```

Implementación

```
package ext;
import battleship.model.Ship;
import java.io.*;
import javax.sound.sampled.*;

public aspect AddSound{
    private static final String SOUND_DIR = "./src/sounds/";

    public static void playAudio(String filename) {
      try {
          File audio = new File(SOUND_DIR + filename);
          AudioInputStream audioIn = AudioSystem.getAudioInputStream(audio);
          Clip clip = AudioSystem.getClip();
          clip.open(audioIn);
          clip.start();
      } catch (UnsupportedAudioFileException
            | IOException | LineUnavailableException e) {
          e.printStackTrace();
      }
    }

    pointcut hitSound() : execution(void battleship.model.Place.hit());
    pointcut sinkSound(): call(void battleship.model.Board.notifyShipSunk(Ship));

    before() : hitSound()
    {
        playAudio("explosion.wav");
    }
    after() : sinkSound()
    {
        playAudio("sink.wav");
    }
}
```

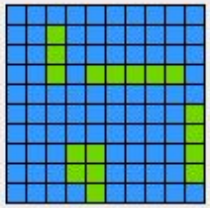

Battleship
Play
Shots: 0


Battleship
Play
Shots: 12

Battleship

Practice | Play | Random

Porta aviones
Battleship
Submarinos
Fragatas
Minesweeper



Todos los barcos destruidos con 35 disparos!