

Aspect Oriented Programming (AOP)

- Dal punto di vista architetturale (progettazione)
 - Durante la mappatura dei requisiti su moduli, può capitare che alcuni requisiti (*concern*) sono trasversali (*crosscutting*) cioè operano su vari moduli
 - Es. nei requisiti: “ogni volta che ...”, “prima che ...”, “se capita che ...”
 - Tracing, sincronizzazione, misure prestazioni, etc. sono crosscutting
 - Riconoscere tali requisiti e prevedere per essi un *aspetto*
 - Progettare prima i moduli per i requisiti di base (*core concern*)
 - Progettare dopo la parte che riguarda i requisiti trasversali

Ing. E. Tramontana - AOP - 26 - Ottobre - 06 1

Join Point

- Uno *join point* è un punto all'interno dell'esecuzione di un programma, es.
 - Invocazione metodi
 - Esecuzione metodi
 - Costruzione oggetti (chiamata a new)
 - Accesso ai campi (lettura e scrittura)
 - Eccezioni
 - Test condizionali, etc.
- Un *join point* è il punto dove interviene il crosscutting concern
 - Ovvero, dove interviene un aspetto

Ing. E. Tramontana - AOP - 26 - Ottobre - 06 3

Aspetti e AspectJ

- AspectJ è una estensione ad *Aspetti* di Java
- Un compilatore (weaver) AspectJ produce codice che la JVM può eseguire
- In AspectJ
 - I requisiti di base sono implementati come classi Java
 - I requisiti crosscutting sono implementati come aspetti
 - Un aspetto è un modulo che consiste di
 - Advice e pointcut
 - Ed è attaccato a certi punti di una classe chiamati join point

Ing. E. Tramontana - AOP - 26 - Ottobre - 06 2

Pointcut e Advice

- Un *pointcut* è il costrutto che seleziona i join point e colleziona il contesto di esecuzione per i join point selezionati
 - Es. di pointcut
 - L'esecuzione del metodo `putPrice()` della classe `Product`
 - L'esecuzione dei costruttori che prendono un argomento `int`
- Ovvero
 - Un pointcut specifica le regole di inserimento
 - Un join point soddisfa le regole specificate
- Un *advice* è il codice che è eseguito quando un join point è selezionato da un pointcut
 - Il codice di un advice è simile a quello di un metodo - incapsula la logica da eseguire quando si raggiunge un join point
 - Il codice di un advice può essere eseguito prima/dopo/al posto di

Ing. E. Tramontana - AOP - 26 - Ottobre - 06 4

Esempio

- Classe

```
public class Circle {
    int radius = 5;
    public int getRadius() {
        return radius;
    }
}
```

- Esempio di join point

- Esecuzione del metodo `getRadius()`

- Pointcut, chiamato `rad()`, che cattura l'esecuzione del metodo `getRadius()` della classe `Circle`

```
pointcut rad() : execution(int Circle.getRadius());
```

Advice

- Un advice può essere eseguito *before*, *after*, o *around* il join point corrispondente al pointcut specificato
- Advice che esegue prima del join point

```
before() : rad() {
    System.out.println("prima del metodo getRadius()");
}
```

- L'advice è eseguito quando il pointcut `rad()` cattura il flusso e prima dell'esecuzione di `Circle.getRadius()`

Weaver

- Un **weaver** è usato per fondere insieme aspetti e componenti e generare una applicazione che contiene solo costrutti tradizionali
 - Elimina i costrutti del linguaggio per aspetti
 - Genera un codice sorgente intermedio (senza aspetti) che viene processato da un normale compilatore
 - Il bytecode prodotto è eseguibile da una normale JVM

Siti web

- Sito di riferimento per AOP

www.aosd.net

AspectJ

eclipse.org/aspectj

scaricare il jar e installare

compilare programmi AOP con: `ajc <lista file java>`

AspectC++

www.aspectc.org

AspectC

www.aspectc.net