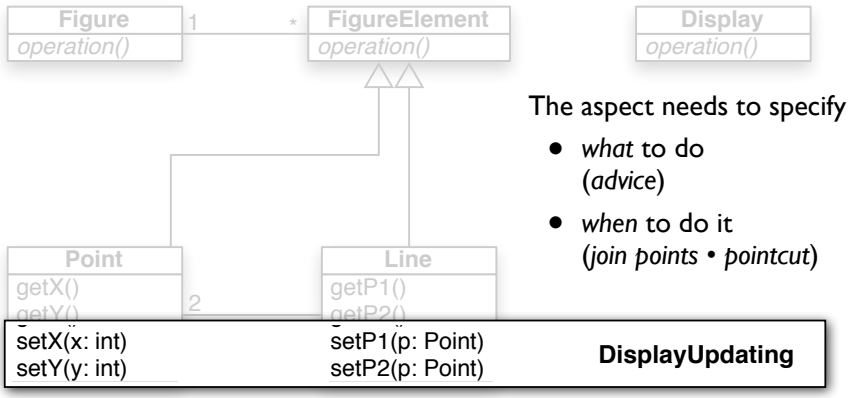
An Aspect

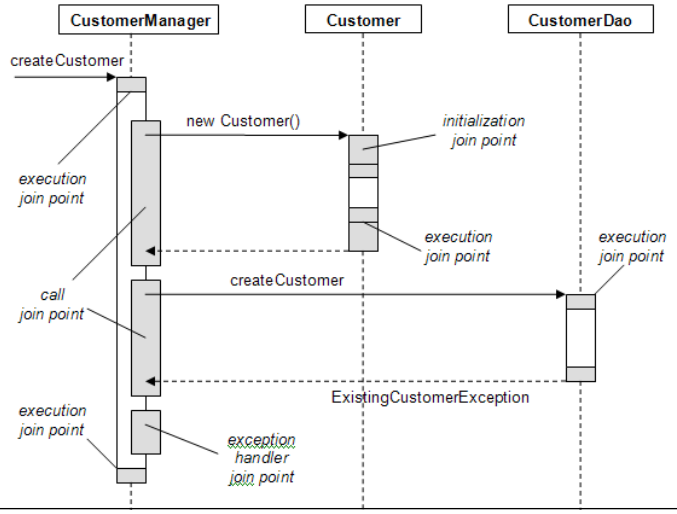


The aspect needs to specify

- *what* to do (*advice*)
- *when* to do it (*join points* • *pointcut*)

For reference, see www.eclipse.org/aspectj

Join Point Types



A Logging Aspect

```
aspect Logging {

  pointcut move():
    call(void FigureElement.setXY(int,int)) ||
    call(void Point.setX(int))              ||
    call(void Point.setY(int))              ||
    call(void Line.setP1(Point))            ||
    call(void Line.setP2(Point));

  after() returning: move() {
    System.out.println("just moved");
  }
}
```

pointcut: when to call the advice

advice: code called at pointcuts

Patterns

```
aspect PublicErrorLogging {
  Log log = new Log();

  pointcut publicMethodCall():
    call (public * com.xerox.*.*(..));

  after() throwing (Error e):
    publicMethodCall() { log.write(e); }
}
```

all calls to xerox functions (all types, all args)

- logs all exceptions being thrown from Xerox

Flow

```
aspect SetsInRotateCounting {
  int rotateCount = 0;
  int setCount = 0;

  before(): call(void Line.rotate(double)) {
    rotateCount++;
  }

  before(): call(void Point.set*(int))
   && cflow(call(void Line.rotate(double))) {
    setCount++;
  }
}
```

use for profiling

aspects are singletons

function is active

- counts all calls to set*() while rotate() is active

Context

```
aspect Logging {
  pointcut move(Line a_line, Point p):
    call(void Line.setP*(Point))
    && target(a_line)
    && args(p);

  after(Line a_line, Point p)
    returning: move(a_line, p) {
    System.out.println(
      "Line " + a_line + " moved to " + p + ". ");
  }
}
```

object being called

call arguments

target() is used to access the object whose method is being called; args() allows to access the arguments.