

# Aspektorientierte Programmierung – Einführung

Thomas Grzenkowski  
[sithgrze@stud.informatik.uni-erlangen.de](mailto:sithgrze@stud.informatik.uni-erlangen.de)

# Motivation

Verbesserung der Wiederverwendung von Klassen / Systemen:

Oft lassen sich gewisse Anforderungen nicht in eigene Klassen kapseln.

⇒ Klassen an Einsatzkontext gebunden!

# Wie soll das erreicht werden?

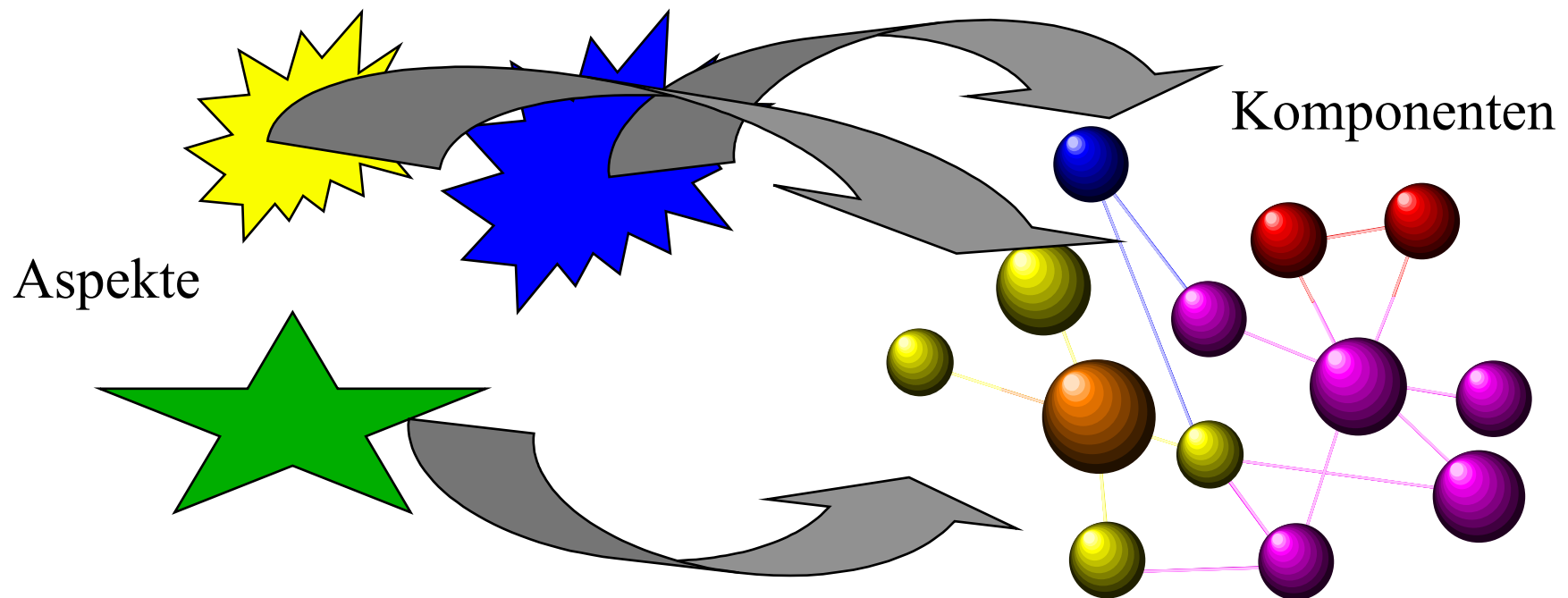
Trennung von Belangen auf Implementierungsebene:

⇒ Auslagern von aspektspezifischen Anweisungen

# AOP = Aspect-Oriented Programming

- Ansatz zum Erkennen und Abtrennen von Anliegen / Aspekten
- Erlaubt verschieden Anliegen in existierende Anwendungen zu weben
- Anliegen stellen für gewöhnlich einen Querschnitt (crosscut) zur objekt-orientierten Struktur dar.

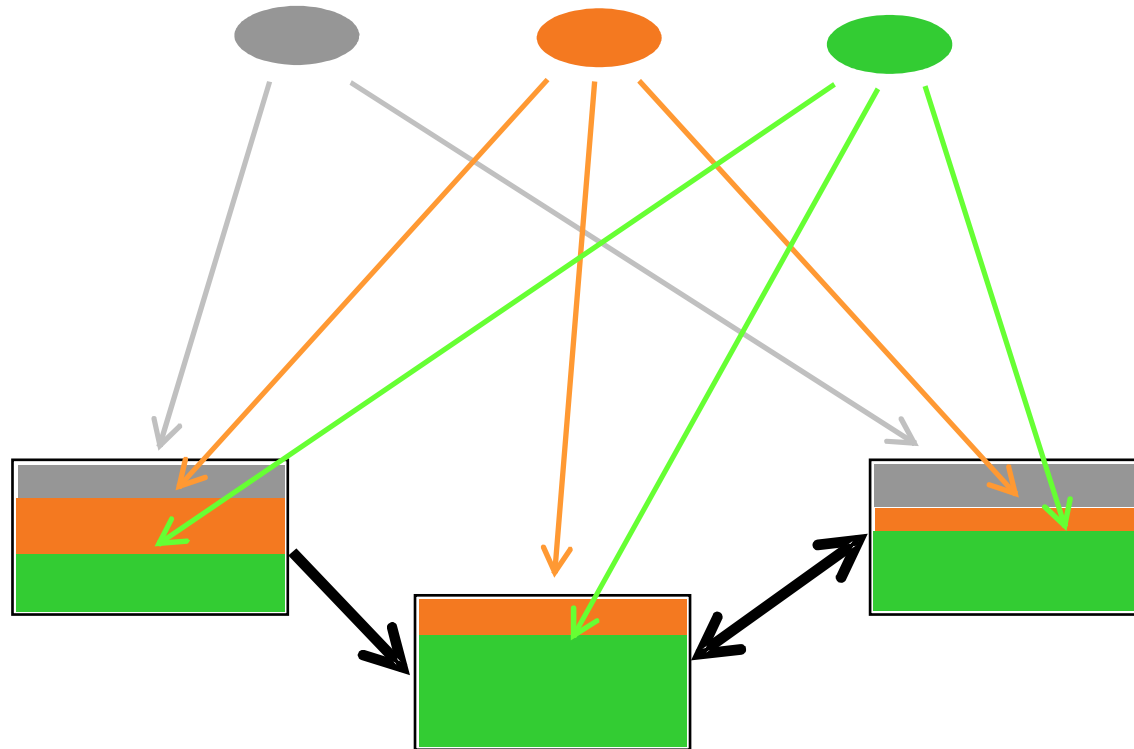
# AOP und Komponenten



# AOP und Komponenten

Use Case:

Komponenten:



# AOP und Code

```
class Book {
    String title;
    String author;
    int year;
    int pages;
    boolean isAvailable;

    public Book(String title, String author, int year, int pages) {
        this.title = title;
        this.author = author;
        this.year = year;
        this.pages = pages;
        this.isAvailable = true;
    }

    public void print() {
        System.out.println("Book: " + title + " by " + author + ", " + year + ", " + pages + " pages");
    }
}

class Library {
    List<Book> books;

    public Library() {
        books = new ArrayList<>();
    }

    public void addBook(Book book) {
        books.add(book);
    }

    public void removeBook(Book book) {
        books.remove(book);
    }

    public void printAllBooks() {
        for (Book book : books) {
            book.print();
        }
    }
}
```

Klassen

```
public class Printer {
    void print(Book book) {
        book.print();
    }
}

public class Library {
    Book find(String title) {
        for (Book book : books) {
            if (book.title.equals(title)) {
                return book;
            }
        }
        return null;
    }
}
```

Aspekte



```
class Book {
    String title;
    String author;
    int year;
    int pages;
    boolean isAvailable;

    public Book(String title, String author, int year, int pages) {
        this.title = title;
        this.author = author;
        this.year = year;
        this.pages = pages;
        this.isAvailable = true;
    }

    public void print() {
        System.out.println("Book: " + title + " by " + author + ", " + year + ", " + pages + " pages");
    }
}

class Library {
    List<Book> books;

    public Library() {
        books = new ArrayList<>();
    }

    public void addBook(Book book) {
        books.add(book);
    }

    public void removeBook(Book book) {
        books.remove(book);
    }

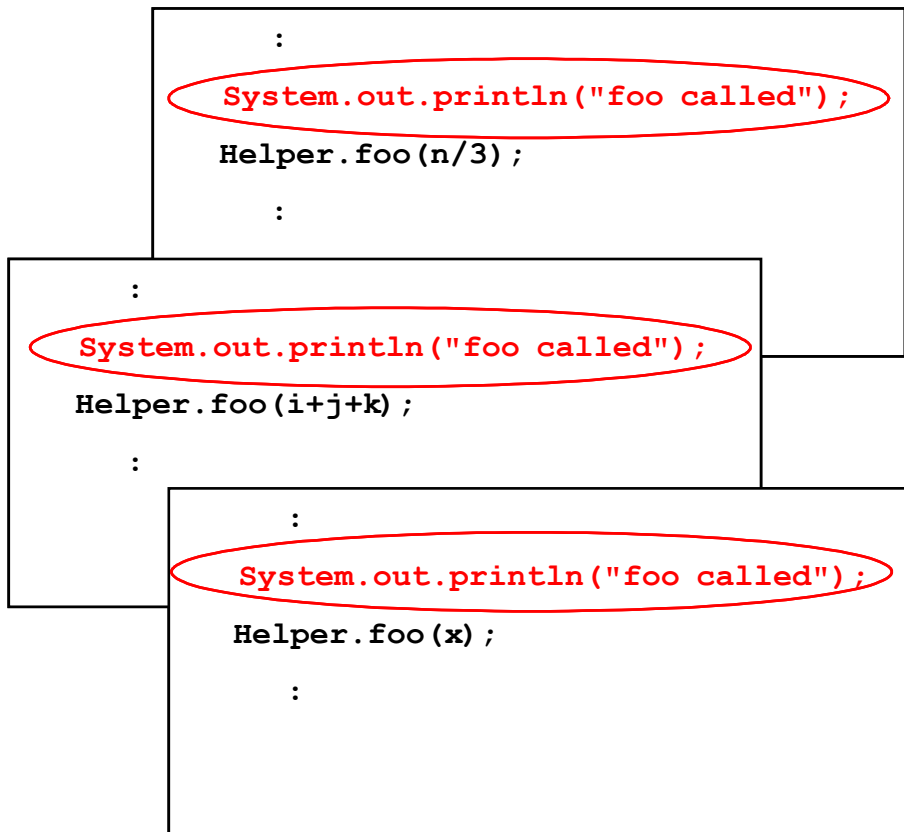
    public void printAllBooks() {
        for (Book book : books) {
            book.print();
        }
    }
}

// Aspect
class PrintAspect {
    void print(Book book) {
        book.print();
    }
}

// Advice
@Around("execution(* Library.find(..))")
public void aroundPrint(ProceedingJoinPoint joinPoint) throws Throwable {
    try {
        joinPoint.proceed();
    } finally {
        PrintAspect printAspect = new PrintAspect();
        printAspect.print(joinPoint.getArgs()[0]);
    }
}
```

# Codebeispiel 1

Problem: Vor jedem Aufruf von „foo“, erfolgt ein lokaler Aufruf



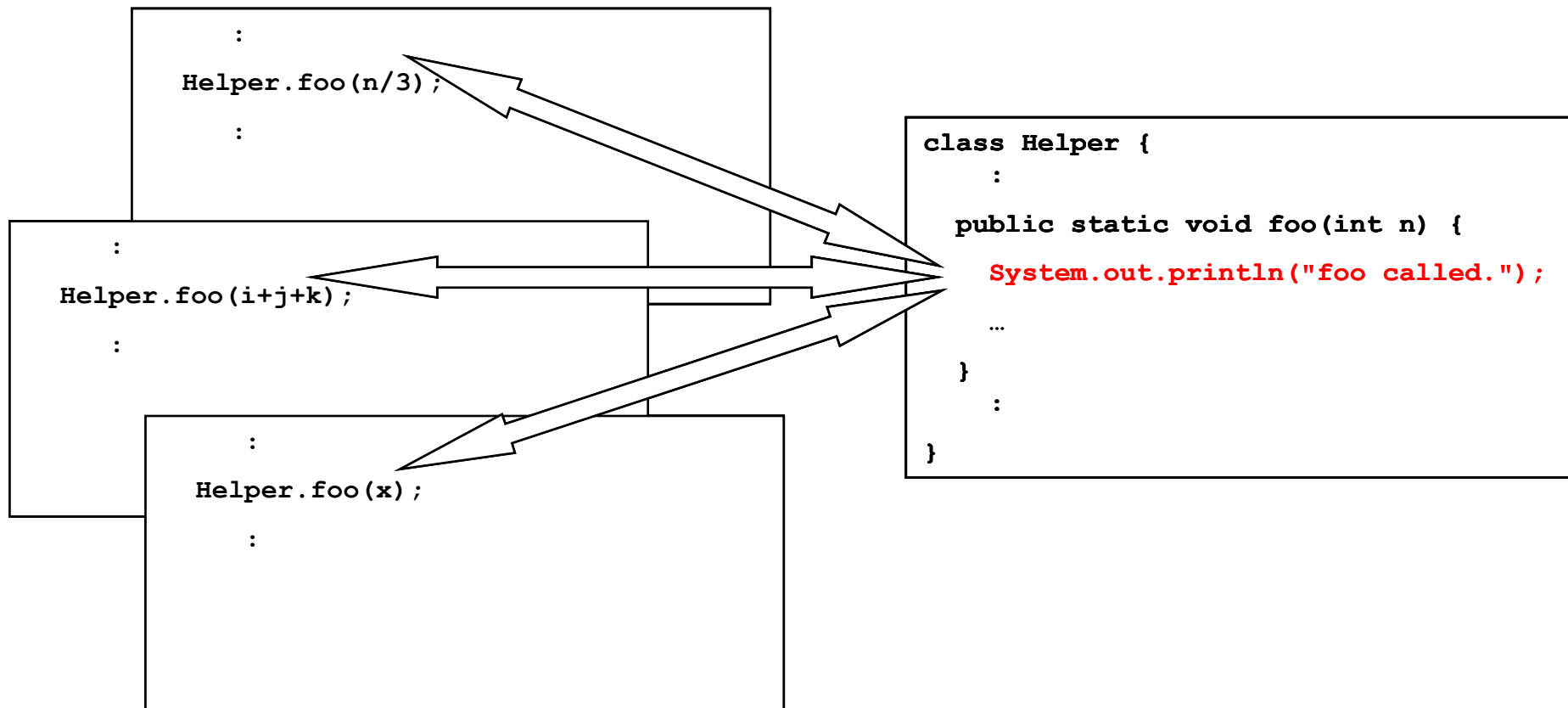
```
class Helper {
    :
    public static void foo(int n) {

        ...

    }
    :
}
```

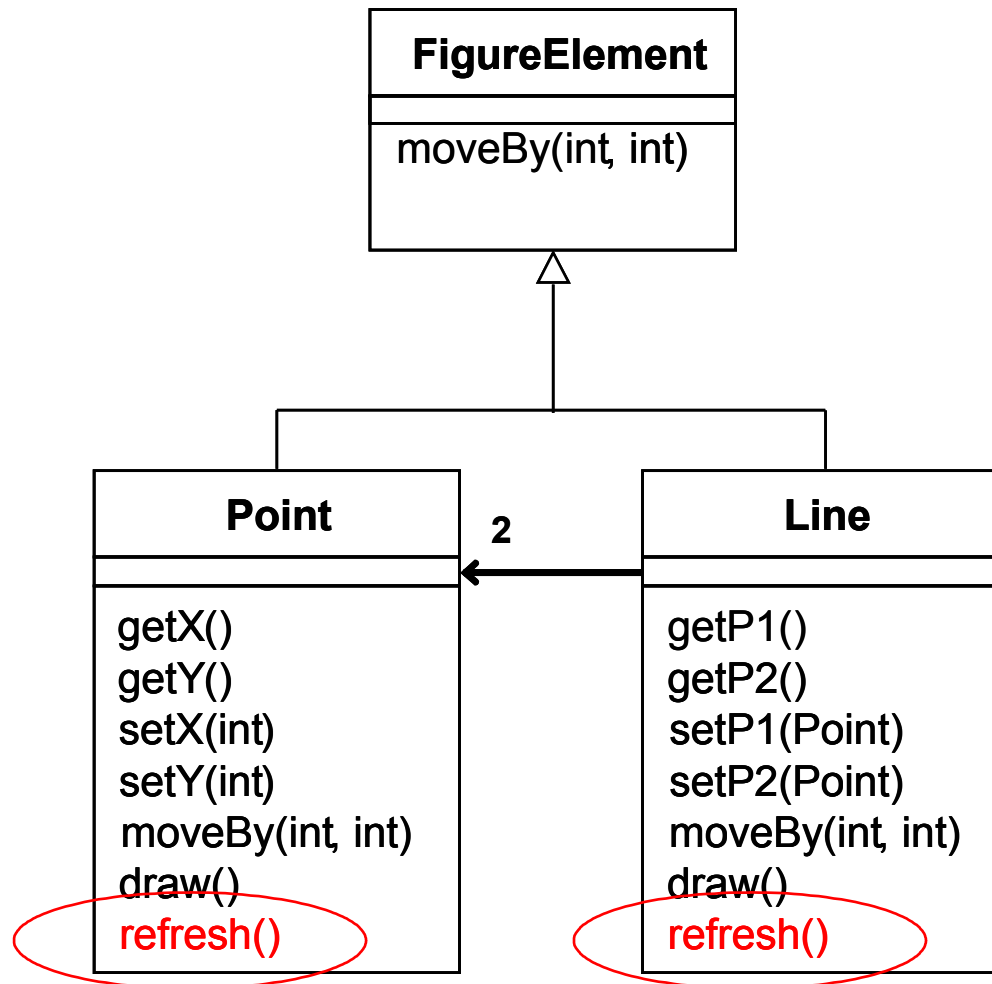
# Codebeispiel 1 – Lösung

⇒ Die Prozedur kann sich darum kümmern



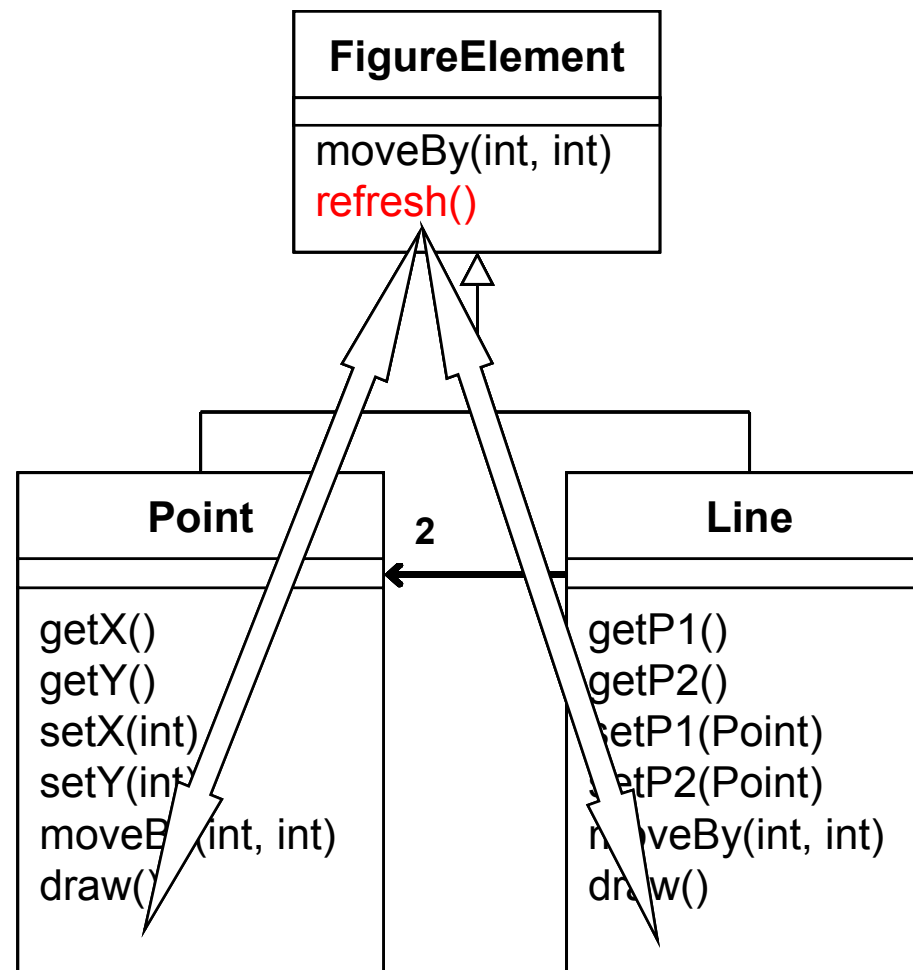
# Codebeispiel 2

Problem: Alle Unterklassen haben gleiche Methode



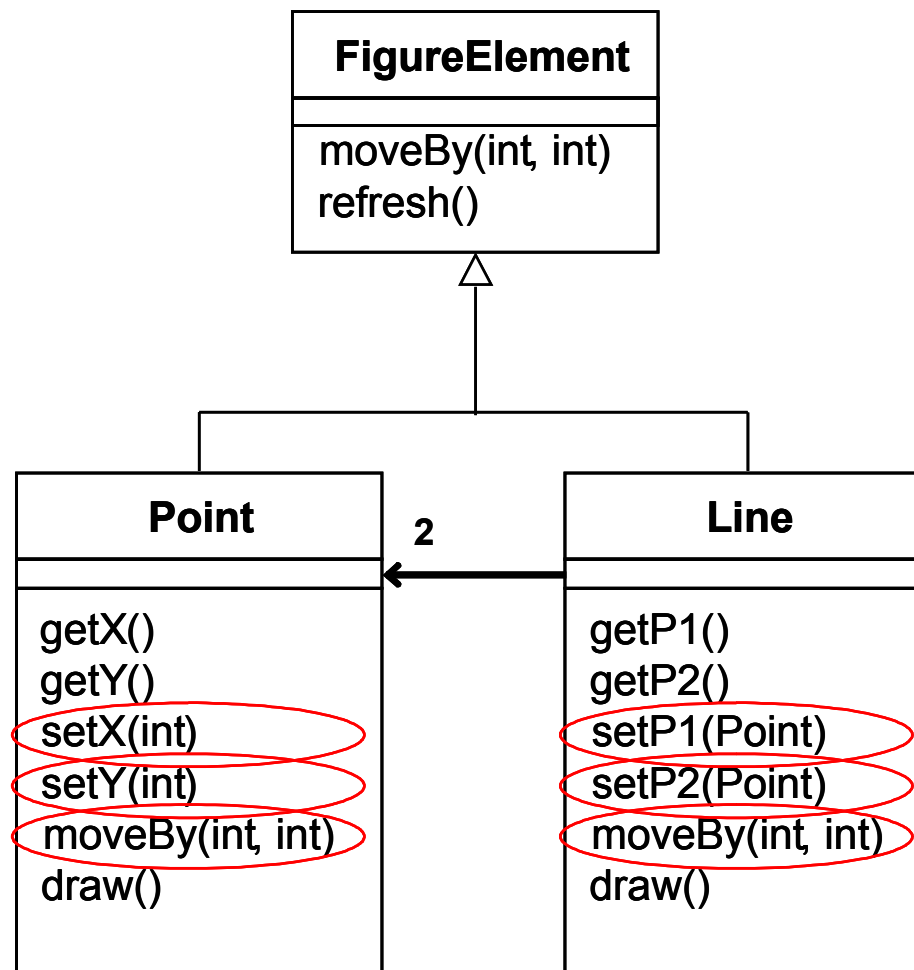
# Codebeispiel 2 – Lösung

⇒ Vererbung kann das lösen



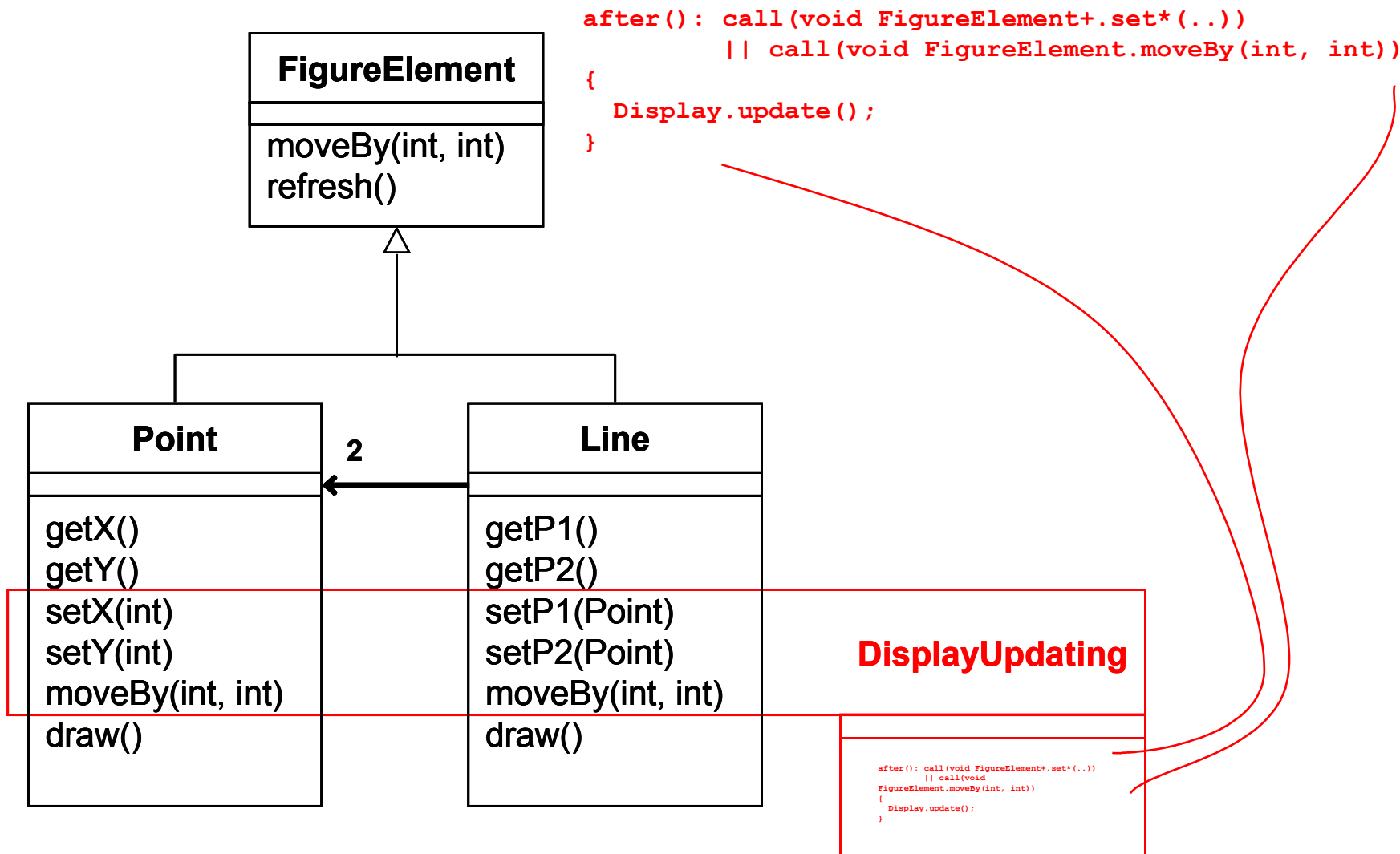
# Codebeispiel 3

Problem: Alle markierten Methoden führen am Ende aus: **Display.update()** ;



# Codebeispiel 3 – Lösung

⇒ mit Hilfe von AOP



# Crosscutting

- Statisch:  
zu bereits  
existierenden Typen  
neue Operationen  
hinzufügen
- Dynamisch:  
Aufruf von Apekten  
an definierten Stellen

# dynamisches Crosscutting

Join points: z.B. Methodenaufruf, Feldzugriff, Klassen- / Objektinitialisierung

Pointcut = join points + [Rückgabewerte] + [Methoden] + [Parameter]

z.B.: `receptions(void Point.setX(int))`

Advice: `wann(Parameter) : pointcuts { Code }`

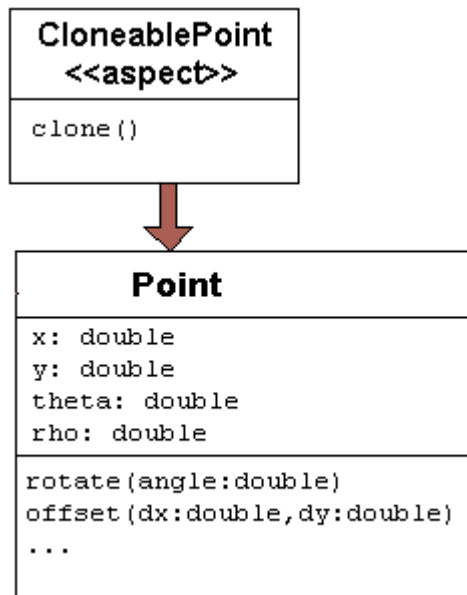
# dynamisches Crosscutting – Bsp. 1

```
Pointcut moves():  
    receptions(void FigureElement.incrXY(int, int)) ||  
    receptions(void Line.setP1(Point)) ||  
    receptions(void Line.setP2(Point)) ||  
    receptions(void Point.setX(int)) ||  
    receptions(void Point.setY(int));  
  
after(): moves() {  
    flag = true;  
}
```

# dynamisches Crosscutting – Bsp. 2

```
before(Point p, int nval):  
    receptions(void p.setX(nval)) {  
        System.out.println("x value of " + p +  
                           " will be set to " + nval + ".");  
    }
```

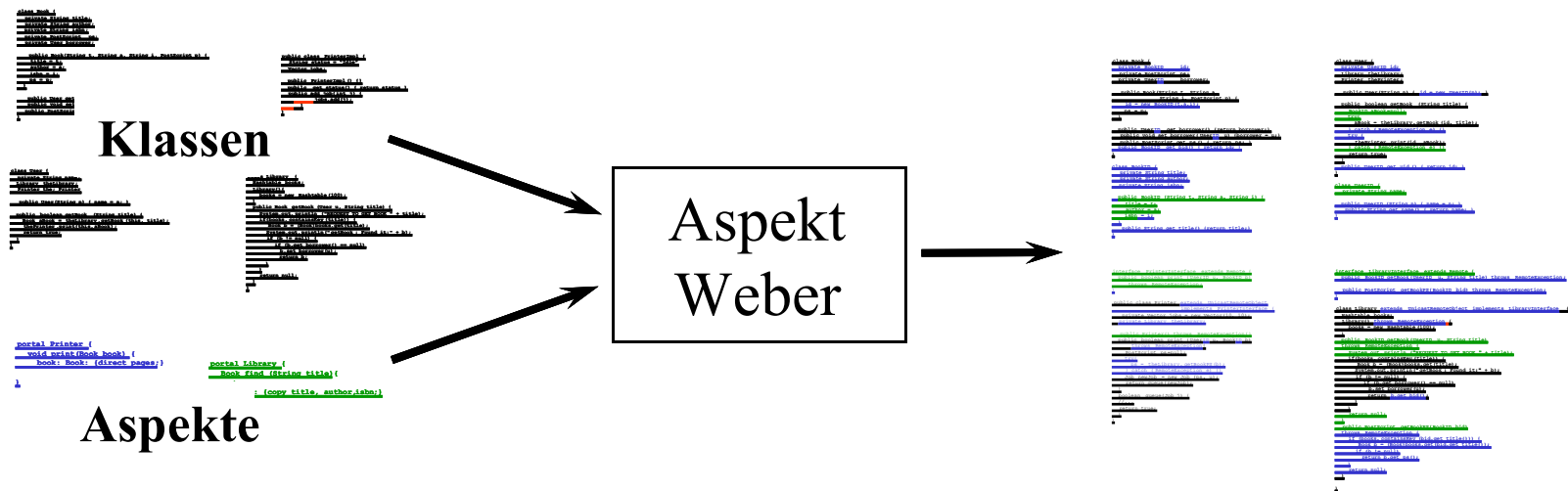
# statische Crosscutting – Bsp.



```
public aspect CloneablePoint {
    declare parents: Point implements Cloneable;

    public Object Point.clone() throws
        CloneNotSupportedException {
        // we choose to bring all fields up to
        //date before cloning.
        makeRectangular();
        makePolar();
        return super.clone();
    }
}
```

# Aspekt Weber



# Aspekt weben: Arten

- Statisch

Durch Präprozessor  
oder Compiler

☹ mehr Speicherverbrauch

- Dynamisch

Zur Laufzeit

☹ Laufzeitumgebung verbraucht  
Speicher und CPU-Zeit

# Literatur

- (1) Zeitschrift: IX 5/1999 S. 74 – 78
- (2) Vortrag: „An Aspect-Oriented Design Approach“ von Mehmet Aksit
- (3) Vortrag: „Use Cases and Aspects Working Seamlessly Together“ von Ivar Jacobson (Firma Rational)
- (4) Vortrag von Gregor Kiczales über AOP
- (5) Vortrag: „The fun has just begun“ von Gregor Kiczales

# Literatur

- (6) Internetseite: „aspect-oriented software development“ (<http://www.aosd.net/>)
- (7) Paper: ECOOP 2001 - An Overview of AspectJ (0.8)  
(<http://www.parc.com/groups/csl/projects/aspectj/index.html> )
- (8) Internetseite: aspectj project  
(<http://www.eclipse.org/aspectj/>)
- (9) Internseite: Basic Techniques - Examples  
(<http://dev.eclipse.org/viewcvs/indextech.cgi/~checkout~/aspectj-home/doc/progguide/ch03s03.html#sec:RolesAndViews>)