Seasar Conference
2006 Autumn

今さら人には聞けないAOP入門

2006.11.12

エスエムジー株式会社

小森　裕介（komori@smg.co.jp）

はじめに

「えっ！？AOPって、もう『今さら聞けない』の？」

そんなことはない！

・・・と思います

でも、AOPが開発の中で
一般的になりつつあるのもまた事実・・・

そろそろ『知らない』って言えなくなってきたアナタに、
AOPの基礎を50分で伝授します！

はじめまして！

- **名前**：小森　裕介
- **Blog**：http://d.hatena.ne.jp/y-komori/
- **所属**：エスエムジー株式会社（http://www.smg.co.jp）

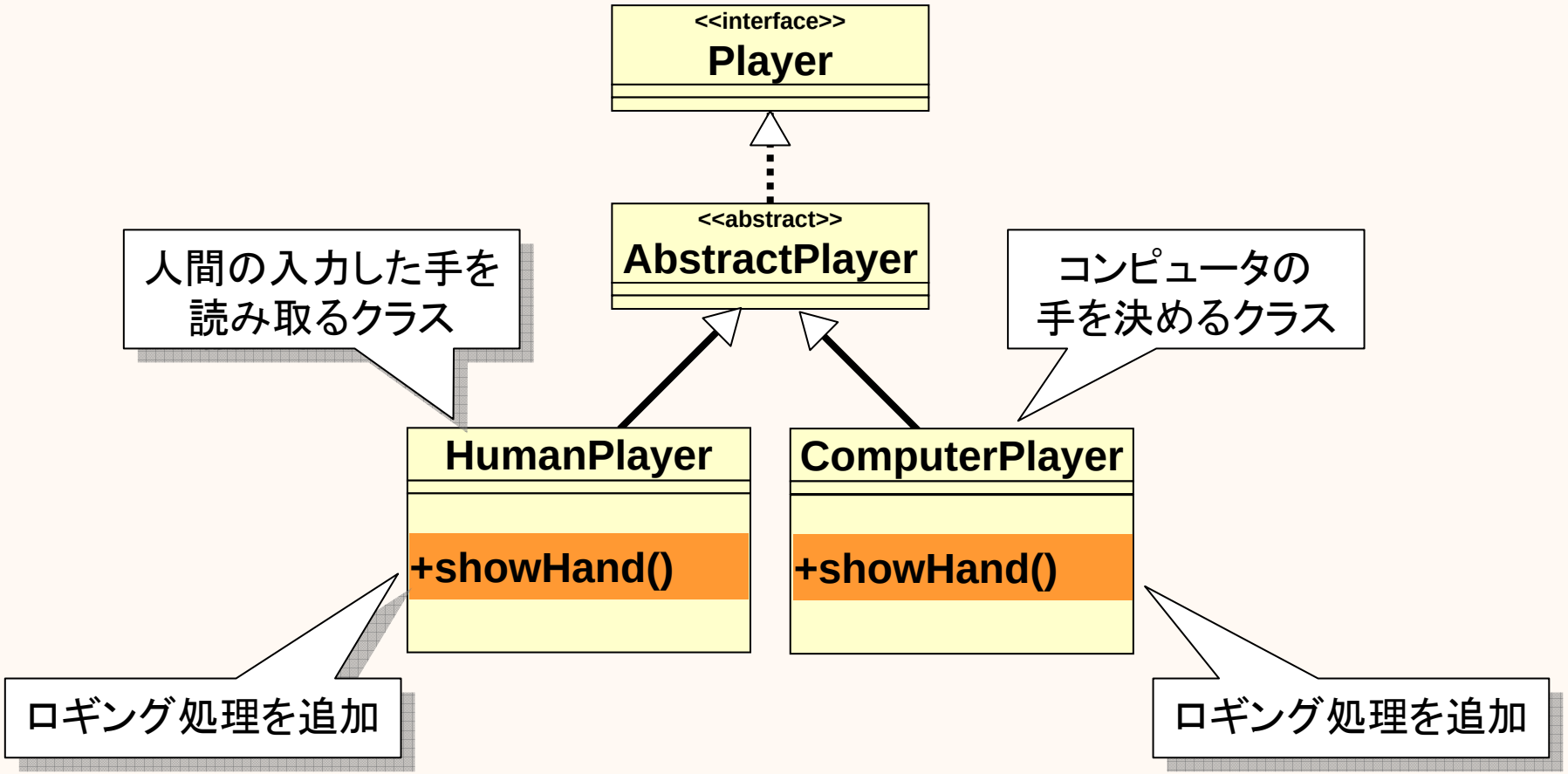
- **主な仕事**：
  - Javaによる集中監視制御システム設計・開発
  - Webアプリケーションシステムの設計・開発
  - 教育・各種執筆活動
    - 日経ソフトウェア「とことん作って覚える! Java入門」連載中
    - 「なぜ、あなたはJavaでオブジェクト指向開発ができないのか」
- **Seasar2とのかかわり**
  - S2Containerコミッタ、S2JMSコミッタ、S2JFaceコミッタ

AOPがなければ何が困る？

- 具体例で考えてみよう！
  - ジャンケンプログラムにロギング処理を追加したい！

Log4jでロギング処理を追加しよう！

- ジャンケンの手を決めるメソッドへロギング処理を追加



追加されたロギング処理

Before

```
public class ComputerPlayer extends AbstractPlayer {
  private Tactics tactics;

  public int showHand() {
    int hand = tactics.readTactics();
    return hand;
  }

  public void setTactics(Tactics tactics) {
    this.tactics = tactics;
  }
}
```

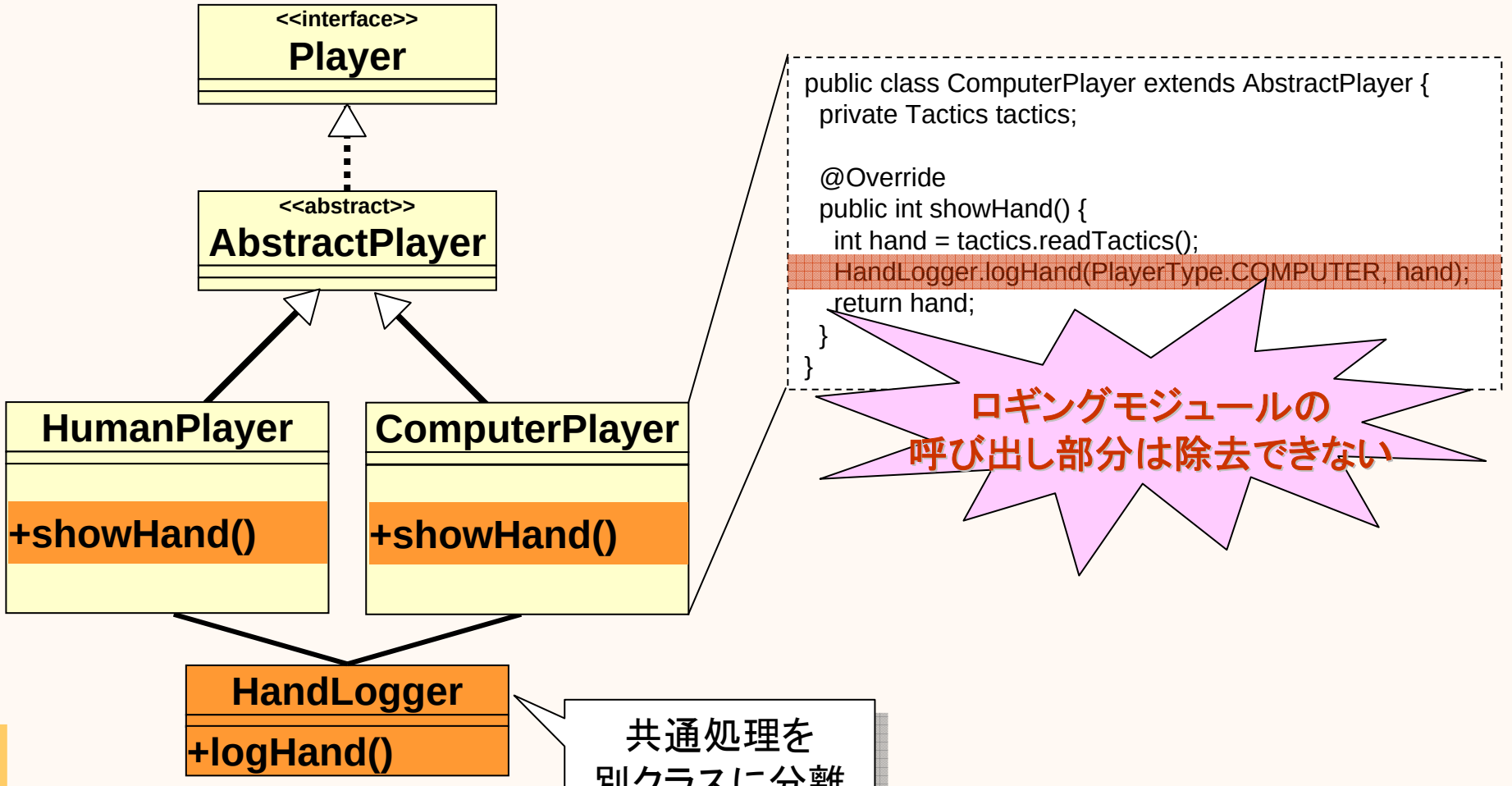
After

```
public class ComputerPlayer extends AbstractPlayer {
  private Tactics tactics;

  private static final Logger logger = Logger.getLogger(ComputerPlayer.class);

  public int showHand() {
    int hand = tactics.readTactics();
    logHand(hand);
    return hand;
  }

  public void setTactics(Tactics tactics) {
    this.tactics = tactics;
  }

  private void logHand(int hand) {
    switch (hand) {
    case Janken.STONE:
      logger.debug("コンピュータ:グー");
      break;
    case Janken.SCISSORS:
      logger.debug("コンピュータ:チョキ");
      break;
    case Janken.PAPER:
      logger.debug("コンピュータ:パー");
      break;
    }
  }
}
```

・・・追加されたロギング処理

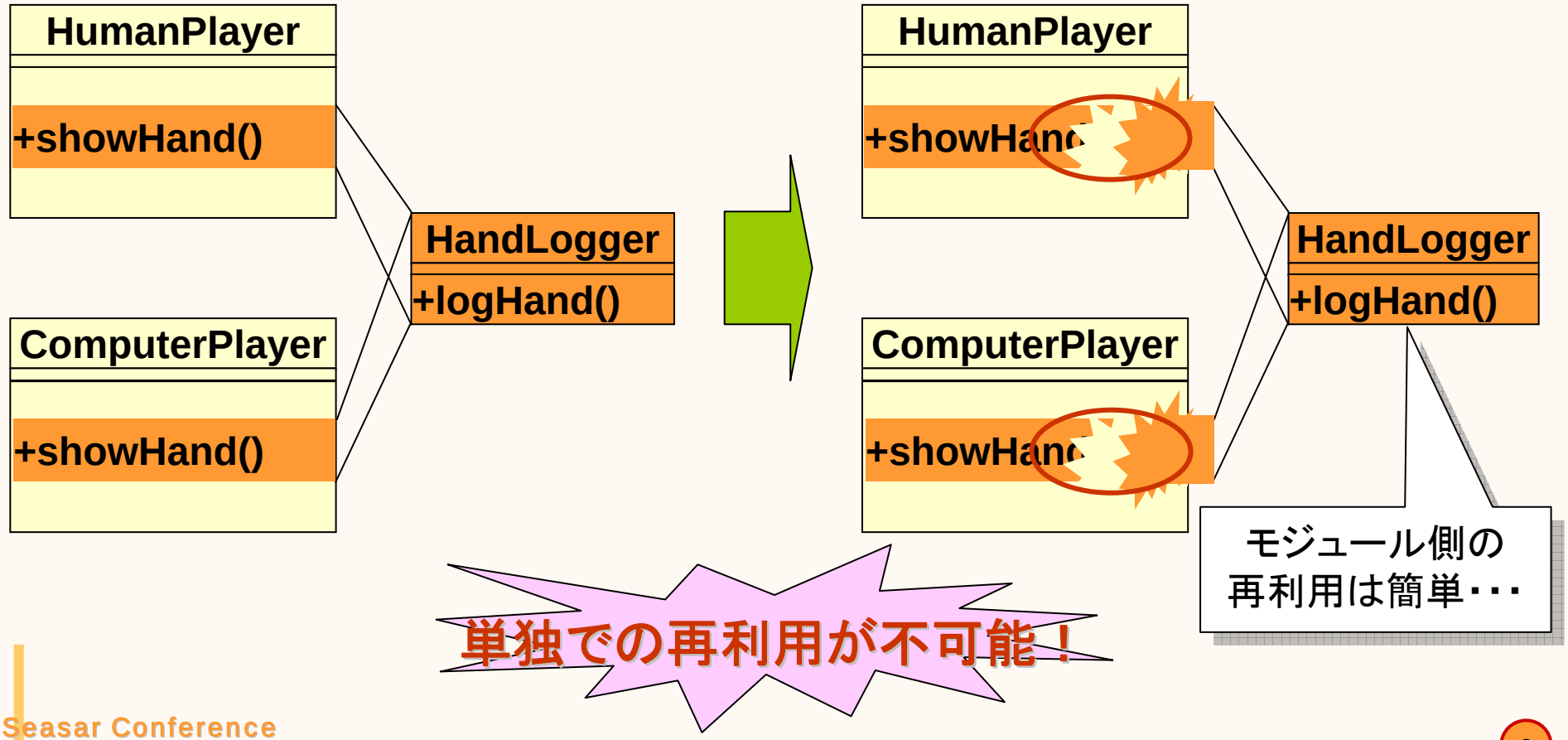
ロギング処理を共通化したい！

- ロギング処理に重複部分があるのでなんとかしたい



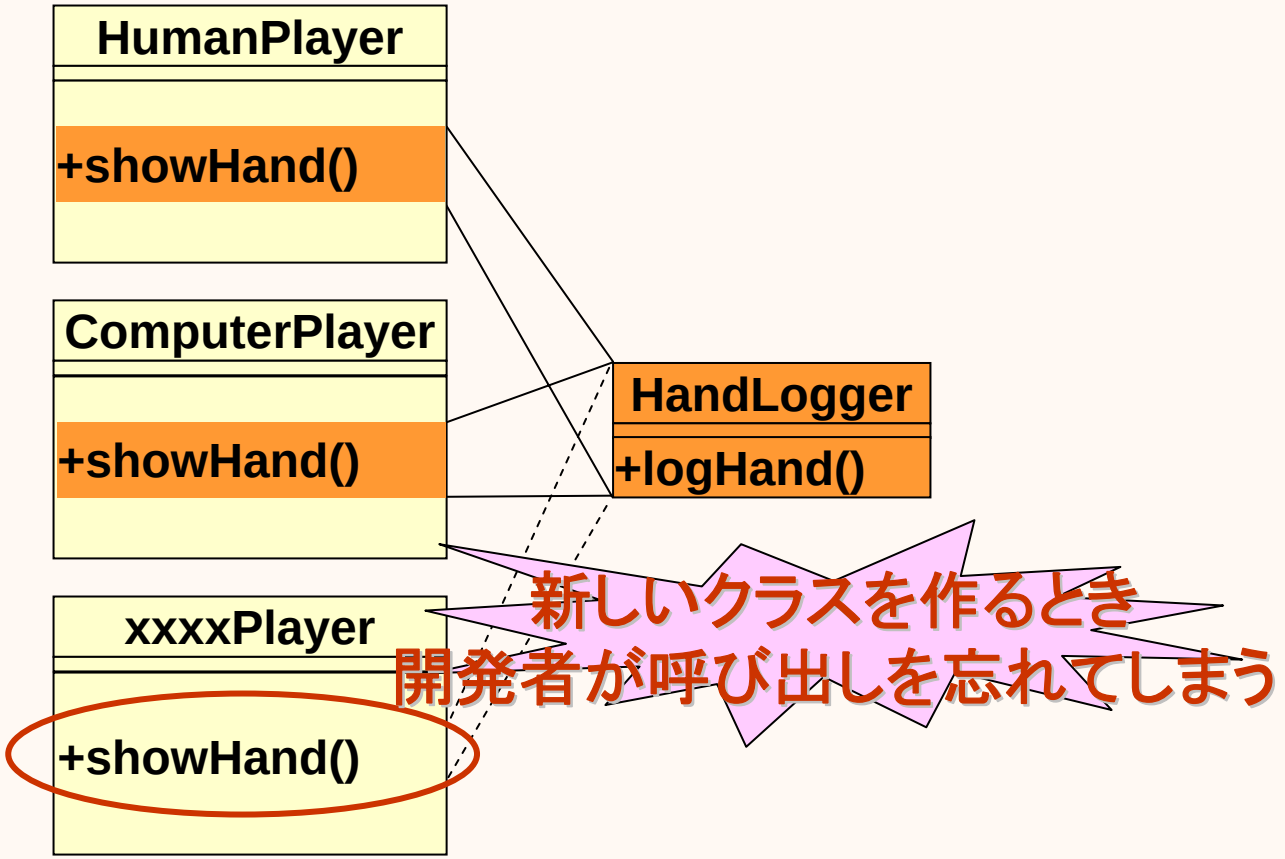
呼び出し部分が残ると何が問題か

- ロギングモジュールの呼び出し部が依存部分として残ってしまう
  - ➡ ロギング処理を削除しようとすると呼び出し元にも影響が出る



似たようなクラスを作成するときの問題

- 共通モジュールの呼び出しは開発者に依存してしまう

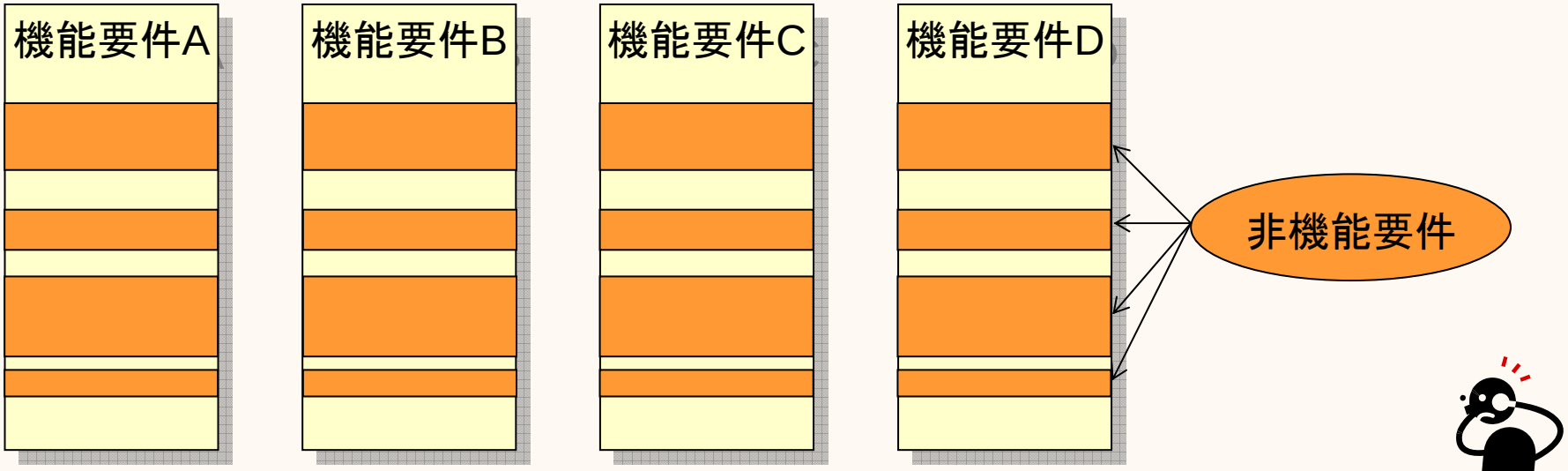


非機能要件はモジュール単独分離が難しい

- 非機能要件に関する処理は、全機能に影響するため、モジュールとして分離しにくい

➡ ロギング処理も非機能要件の一つ

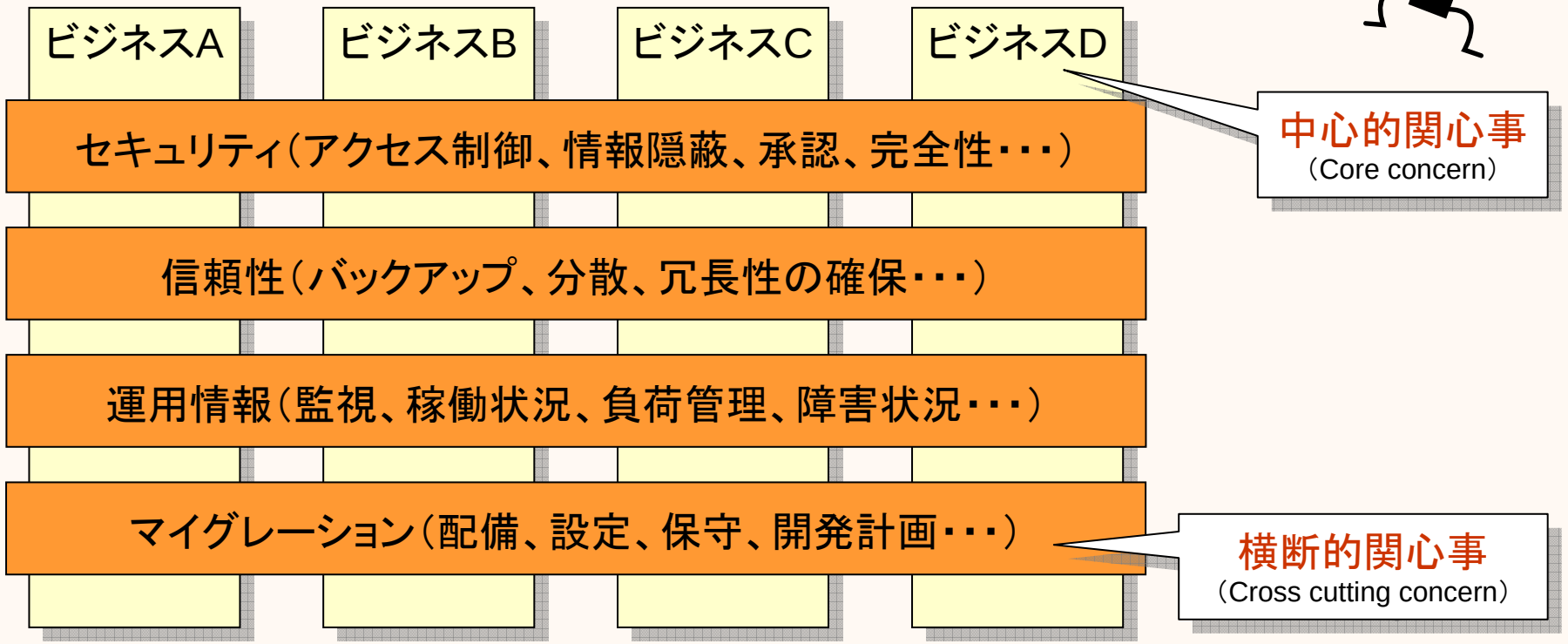
非機能要件・・・システムの本来の機能とは関係ないが、信頼性や保守性、使いやすさを向上させるための要件



各機能に共通する処理をモジュール化する方法はないの？

アスペクト指向は関心事の分離から

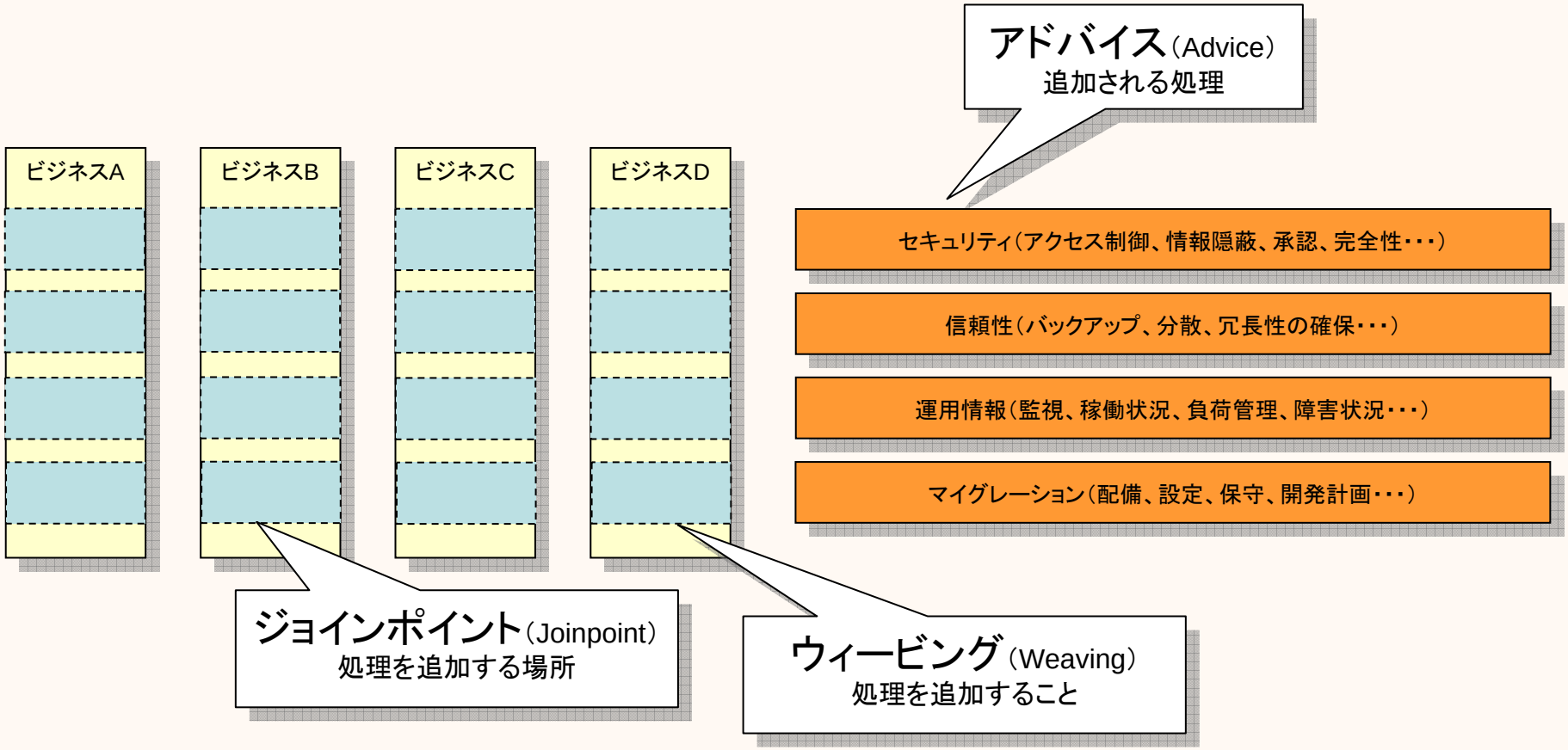
- システムを2種類の要件に分け、横断的関心事を分離するのがアスペクト指向の考え方



【出典】『Seasar2で学ぶDIとAOP』（arton著、技術評論社）p20

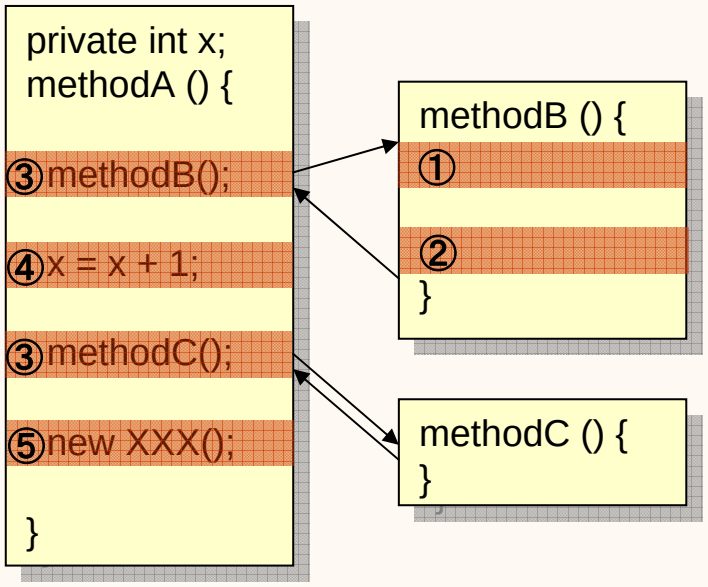
ジョインポイントとアドバイス

- アスペクト指向では、機能の中にジョインポイントを定義し、アドバイスをウィービングする



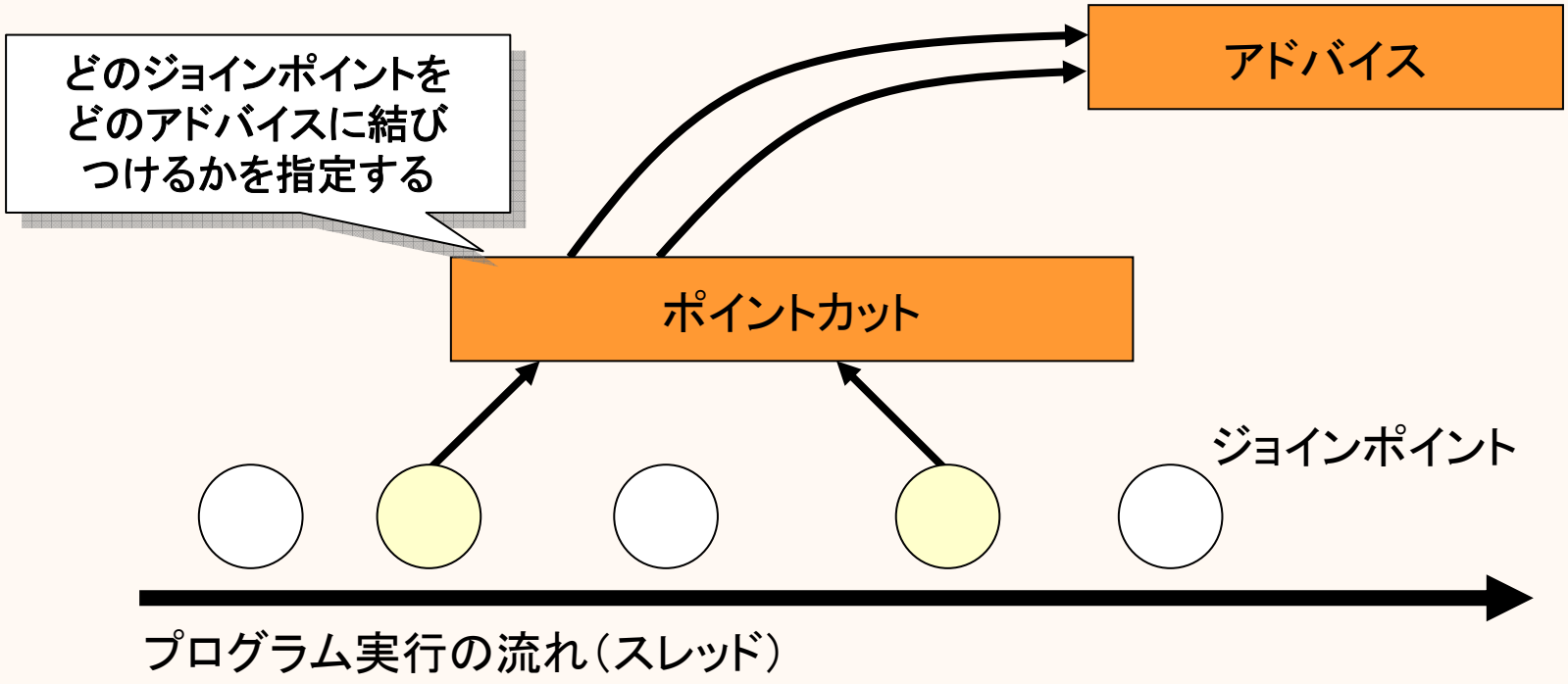
どのようなジョインポイントがあるか？

- ジョインポイント（Joinpoint）はアドバイスを挿入可能なプログラム上の位置
  - ① 呼び出されたメソッドの開始点
  - ② 呼び出されたメソッドの終了点
  - ③ メソッドの呼び出し地点
  - ④ フィールドの参照、更新地点
  - ⑤ クラスの生成地点



ジョインポイントとアドバイスを結びつける ポイントカット

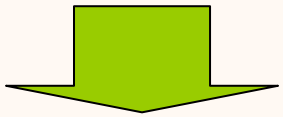
- どのジョインポイントにどのアドバイスを結びつけるかはポイントカット(Pointcut)で指定する



【出典】『アスペクト入門』(千葉 滋 著、技術評論社)p47

実際にどうやって実現するの？

- AOP（Aspect-Oriented Programming：アスペクト指向プログラミング）はどうやって実現するの？

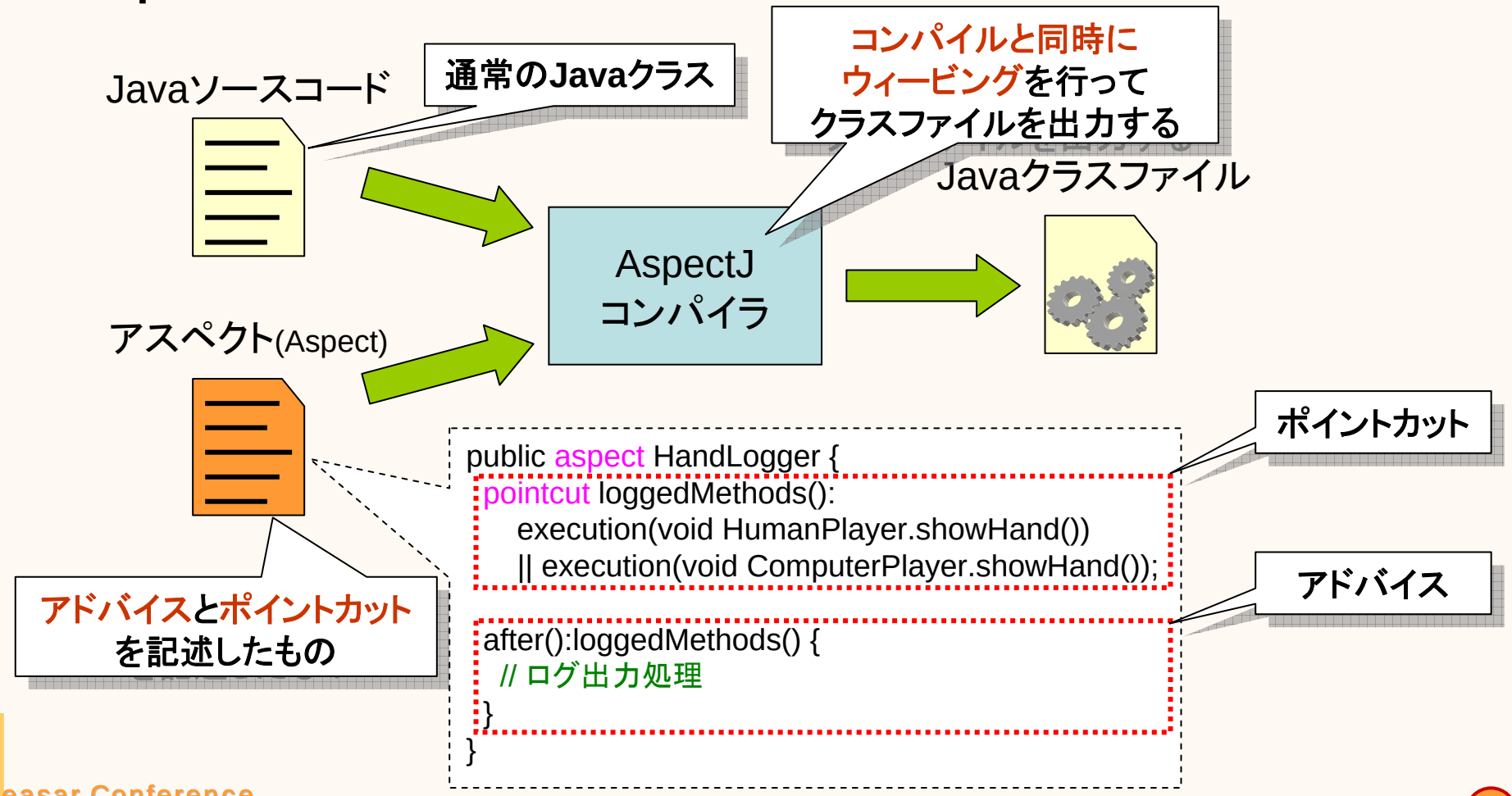


AOPを利用するには2種類の方法がある

1. 専用コンパイラを利用する方法
2. DIコンテナを利用する方法

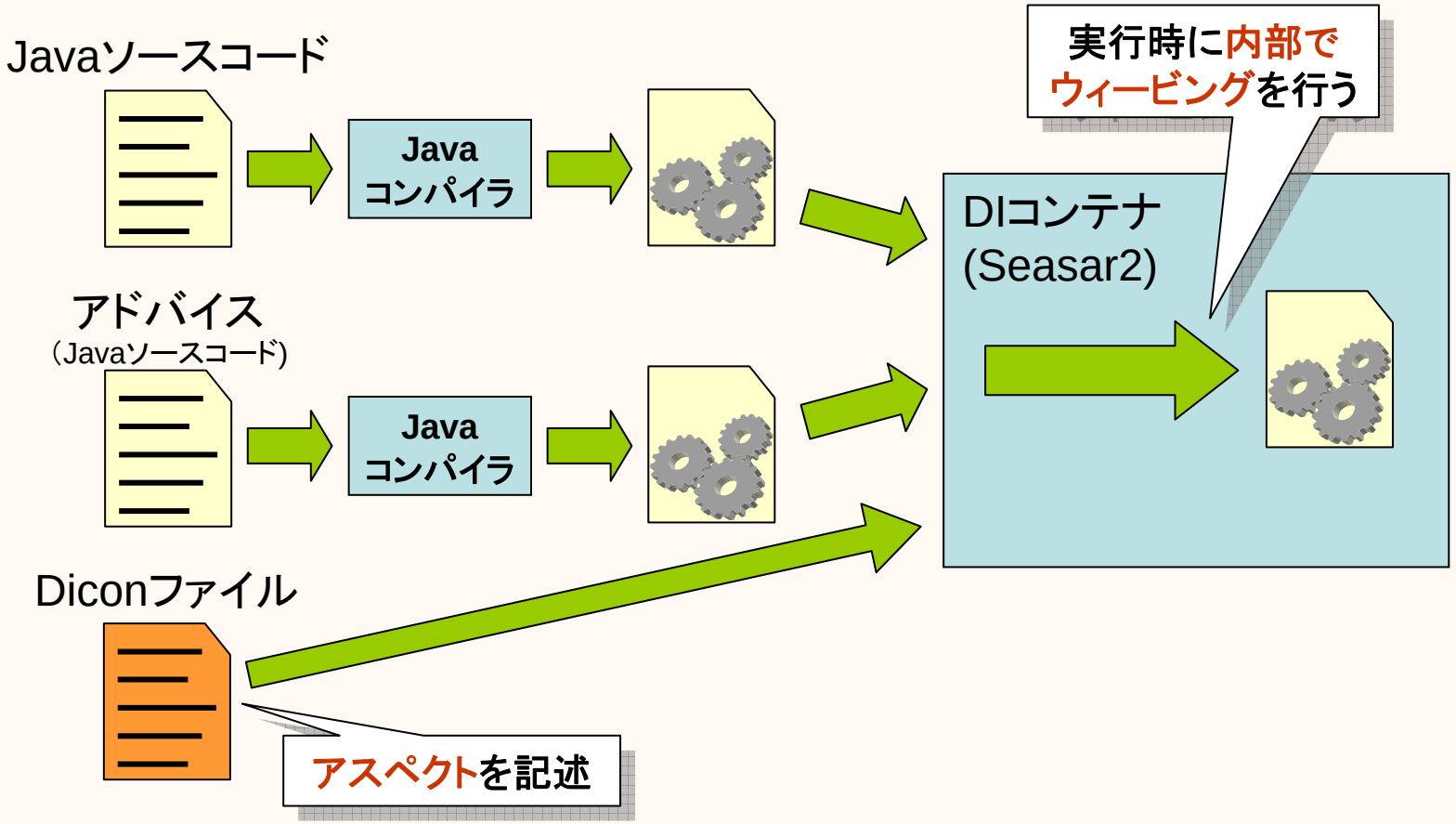
専用コンパイラを利用する方法

- AspectJによるAOPの実現



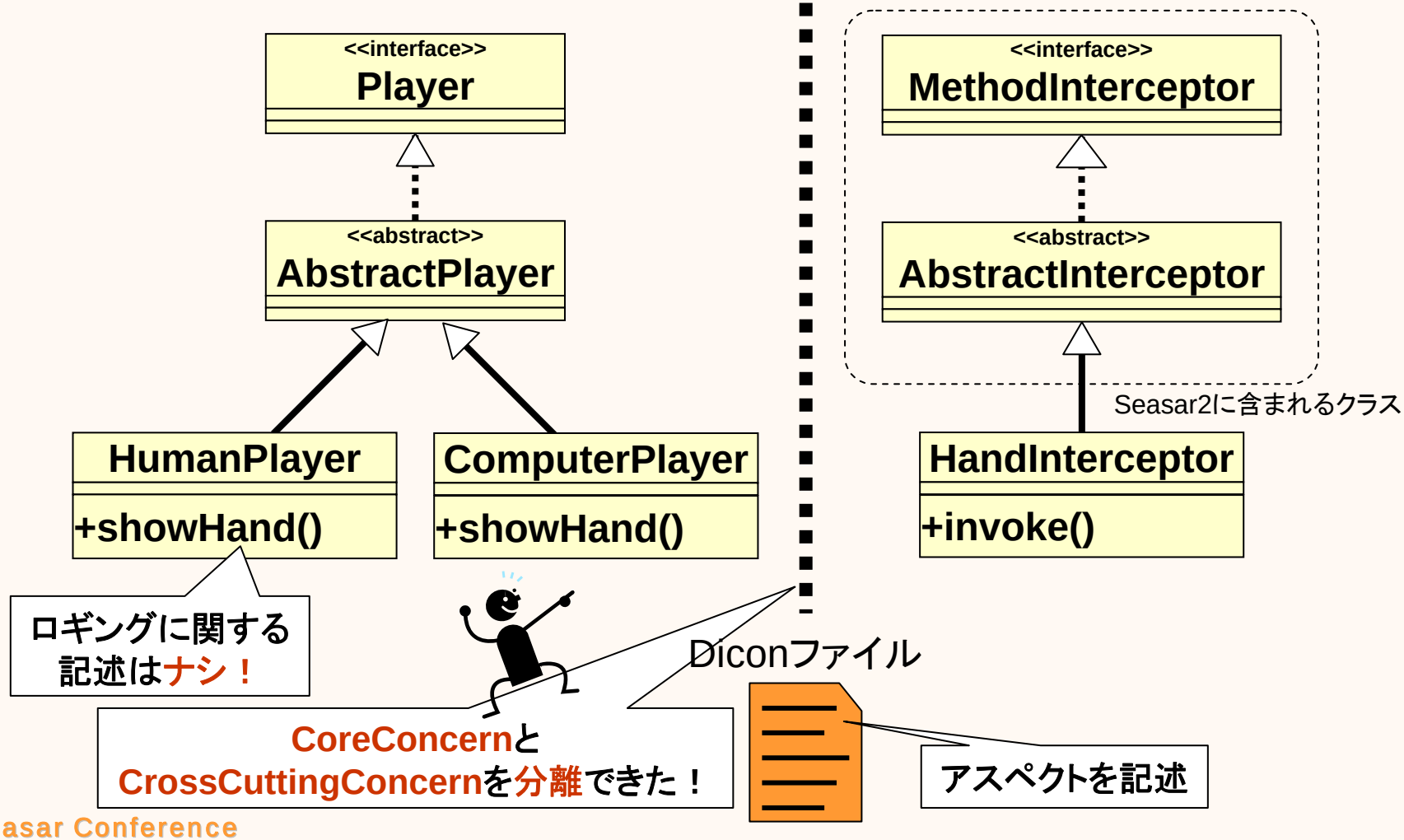
DIコンテナを利用する方法

- DIコンテナ(Seasar2)によるAOPの実現



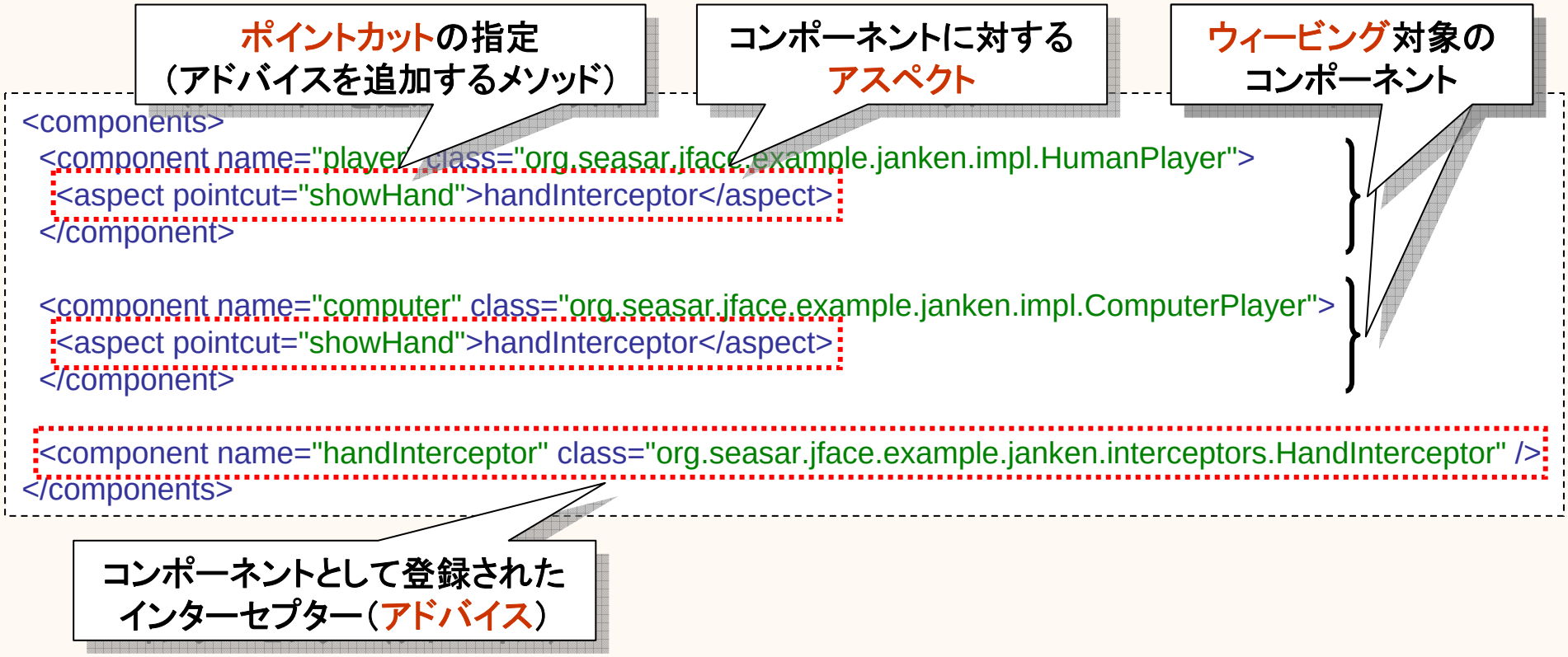
S2AOPを体感してみよう！

- 「アスペクトでジャンケン」のロギング処理をS2AOPで書き直す



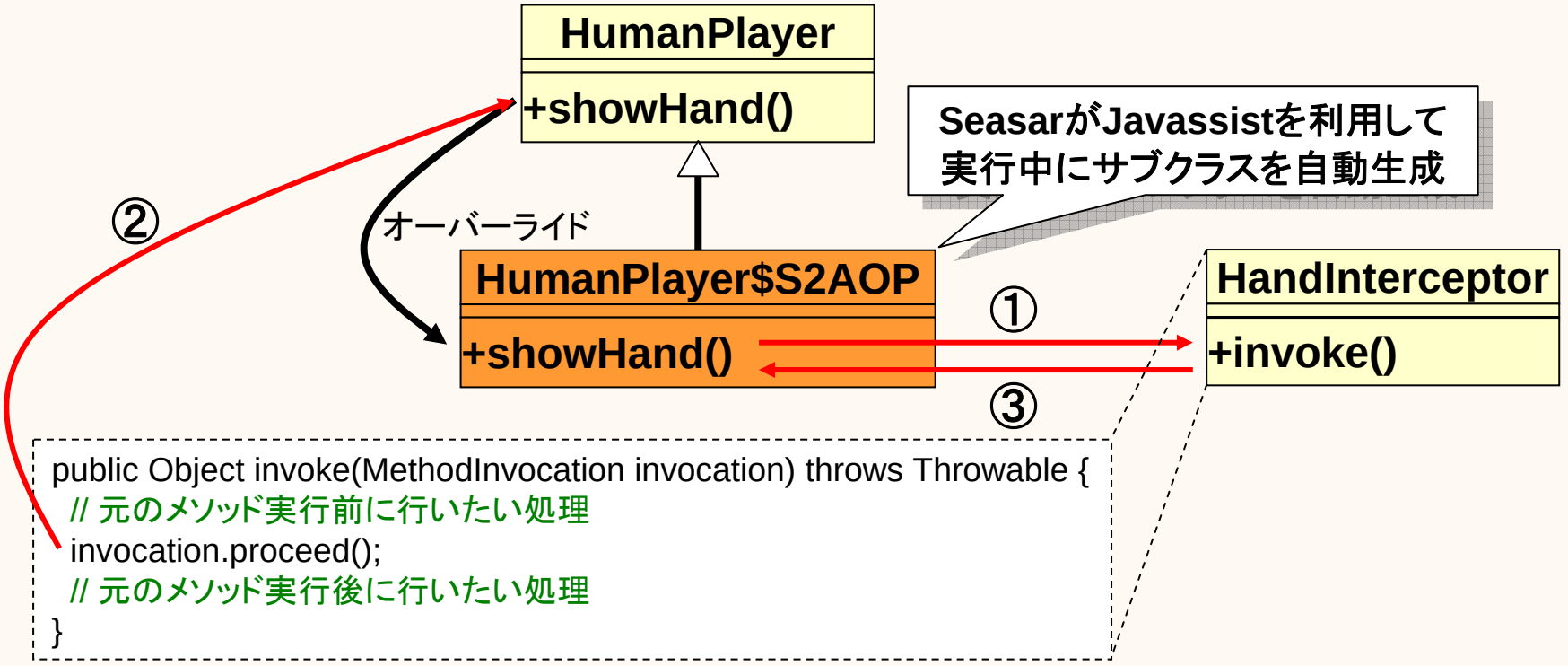
S2AOPでのDiconファイルの記述

- S2AOPではXML（Diconファイル）にアスペクトを記述する



S2AOPの仕組み

- ちょっとだけS2AOPの仕組みを紹介しましょう・・・



More Info! Javassist：国産のバイトコートエンジニアリングライブラリ
http://www.csg.is.titech.ac.jp/~chiba/javassist/

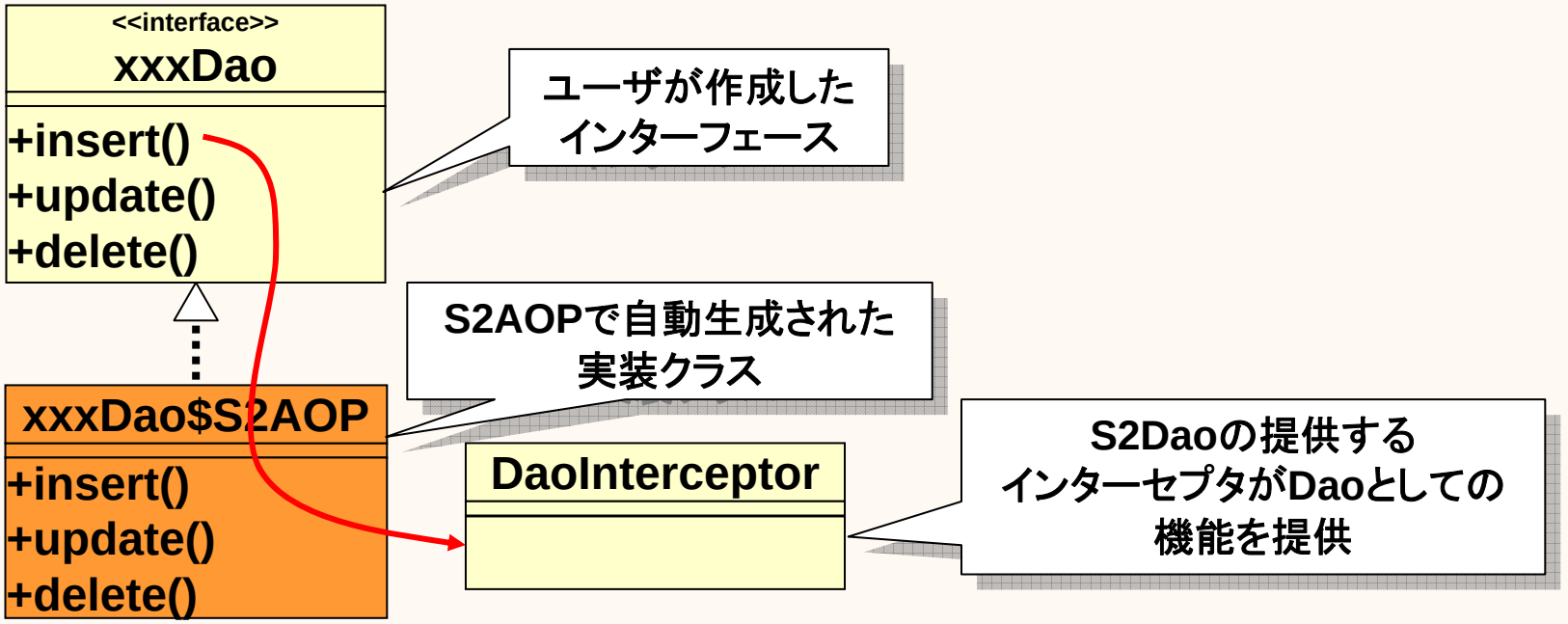
Seasar2の提供するインターセプター

- TraceInterceptor
  - メソッド呼び出しのトレースを自動出力
- ThrowsInterceptor
  - 例外のキャッチを一括処理
- ToStringInterceptor
  - toString()メソッドを自動処理
- MockInterceptor
  - モックを使ったテストを支援
- DelegateInterceptor
  - メソッド呼び出しを別コンポーネントへ委譲
- SyncInterceptor
  - メソッド呼び出しを同期化

More Info! http://s2container.seasar.org/ja/aop.html

S2DAOとAOP

- S2DAOもAOPによって実現されている
  - インターフェースしか用意していないのに処理が行われるのはどうして？？

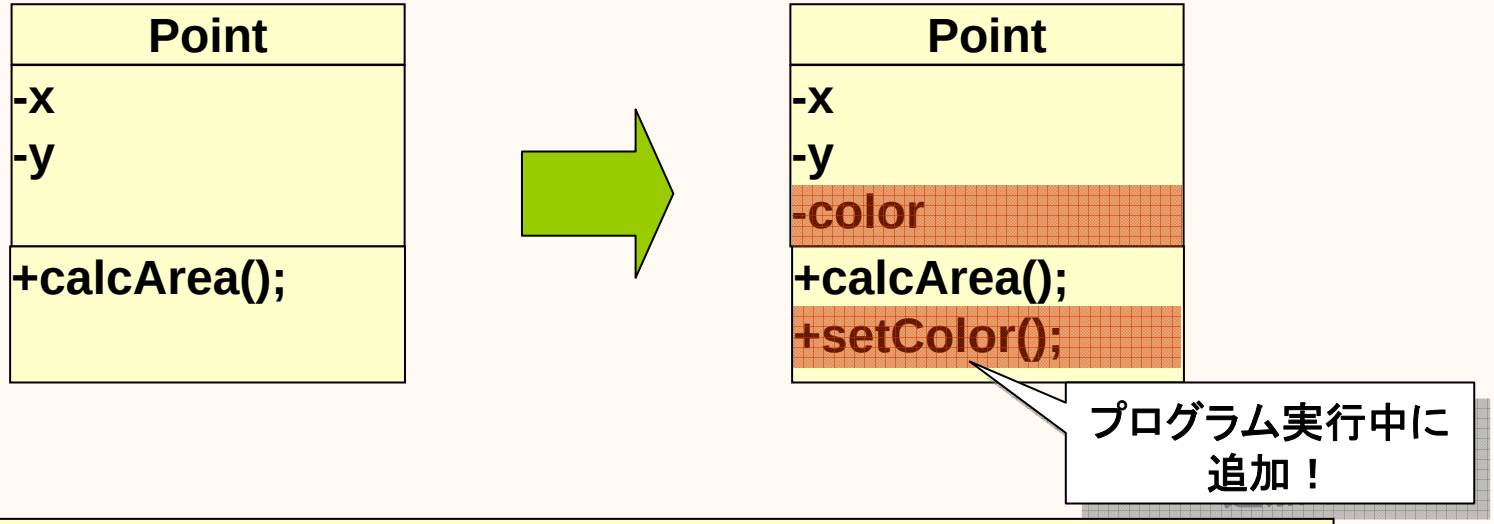


アスペクト指向でソースコード生成を
伴わないスマートなDAOが可能に！

More Info! http://s2dao.seasar.org/ja/s2dao.html

既存のクラスを拡張するインタータイプ宣言

- インタータイプ宣言とは
  - 既存のクラスの静的構造を変更するアスペクトの機能
- Seasar2.4で提供される新機能InterType
  - 既存のクラスに対してメソッドやフィールドを自由に追加可能

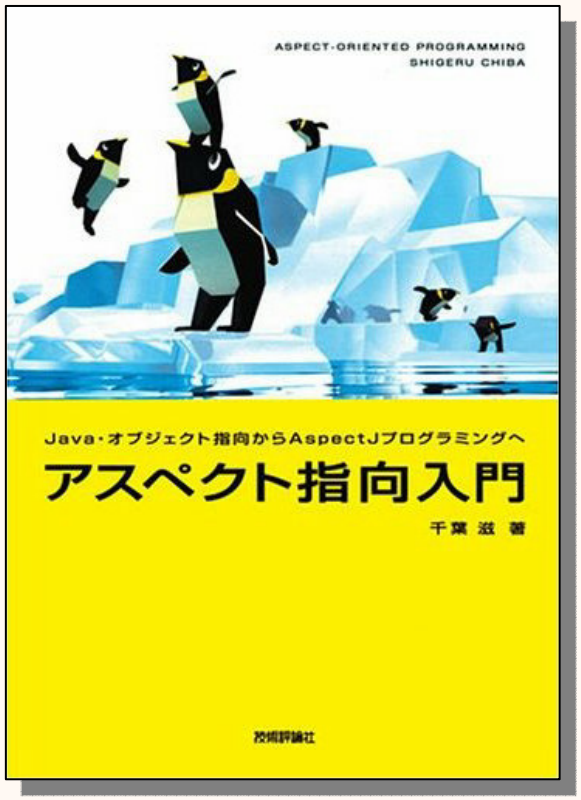


ソースコード自動生成に替わる手段として利用可能！

More Info! http://s2container.seasar.org/ja/aop.html#OriginalInterType

アスペクト指向をさらに知りたい方へ

- 『アスペクト指向入門』-Java・オブジェクト指向からAspectJプログラミングへ
  - 著:千葉 滋
  - 価格:￥2,480＋税
  - 出版社:技術評論社
  - ISBN:4-7741-2581-4



ご質問について

スピーカーブースで
お待ちしています
気軽にお越しください

ご静聴
ありがとうございました