

オブジェクト指向 エクササイズのススめ

12-A-6

菅野洋史/大村伸吾

株式会社オージス総研
オブジェクトの広場

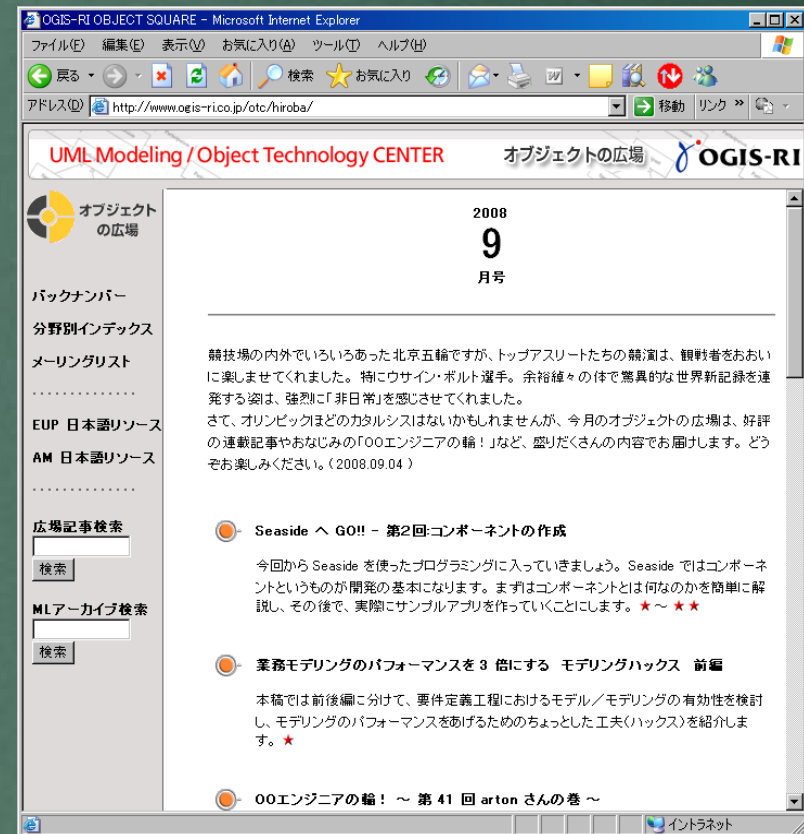
講演者のご紹介

- 株式会社オージス総研

オブジェクトの広場編集部

- 月間のオンラインマガジン

- ThoughtWorksアンソロジーを
翻訳しました!!



ThoughtWorks アンソロジー

- ThoughtWorks社コンサルタント
の

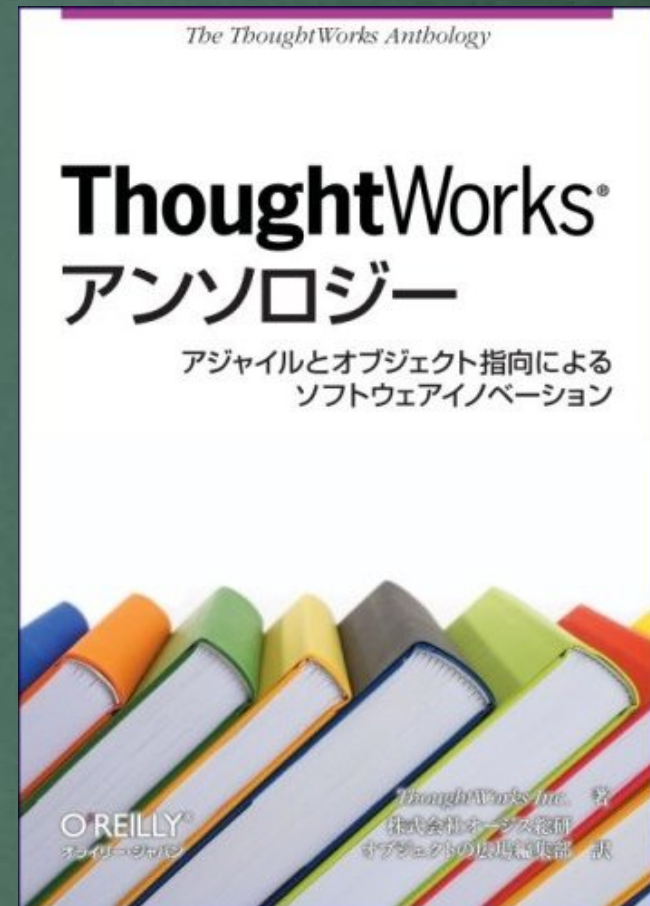
骨太なエッセイ集

- **様々な** ジャンルを収録

DSL、プログラミング、設計、
マネジメント、ビルド、デプロイ、テスト...

- オライリーさんブースで


絶賛販売中！



ところで
オブジェクト指向開発
していますか？



本当にオブジェクト指向?

- オブジェクト指向言語使えばオブジェクト指向開発でしょうか?
 - 処理を全部、ロジッククラスに持たせてませんか?
 - 継承やインタフェース使えばオブジェクト指向だと言ってますか?
 - Struts (ry
- 

それは
オブジェクト指向
では無い



オブジェクト指向でやるなら

責務を持ったオブジェクトがコラボレーションすることによって、複雑なシステムを構成する
といいね

- もちろん必ずしも、オブジェクトが銀の弾丸
なわけじゃない

オブジェクト指向が出来ない理由


- 以前の慣習から抜け出すのが難しい
- 一部の開発者だけOOを分かっていればいいという風潮がある(特に設計やアーキテクチャ)
- 実践的な教材/教育が少ない

ネックは人(教育)



そこで オブジェクト指向エクササイズ

オブジェクト指向プログラミングを強制的に身に
着けるためにハードなコーディング規約を実際
のプログラムに適用するエクササイズ



誰がやる？

- 開発の仕事には入って数年目の人
- ある程度、自分は出来ているという認識を持ってる人(の鼻っ柱を叩き折る)
- 最近、オブジェクト指向で開発していないなーというオブ厨の人



しばし、オブジェクト指向エクササイズの
内容説明を・・・



実際にためしてみる



「お題」

一見00っぽい感じだけど、エクササイズの観点で見るとダメダメな、あるツールをエクササイズのルールに従うように書き直す。



Amazon中古価格調査ツール

単純なコマンドラインツールを

リファクタリングします

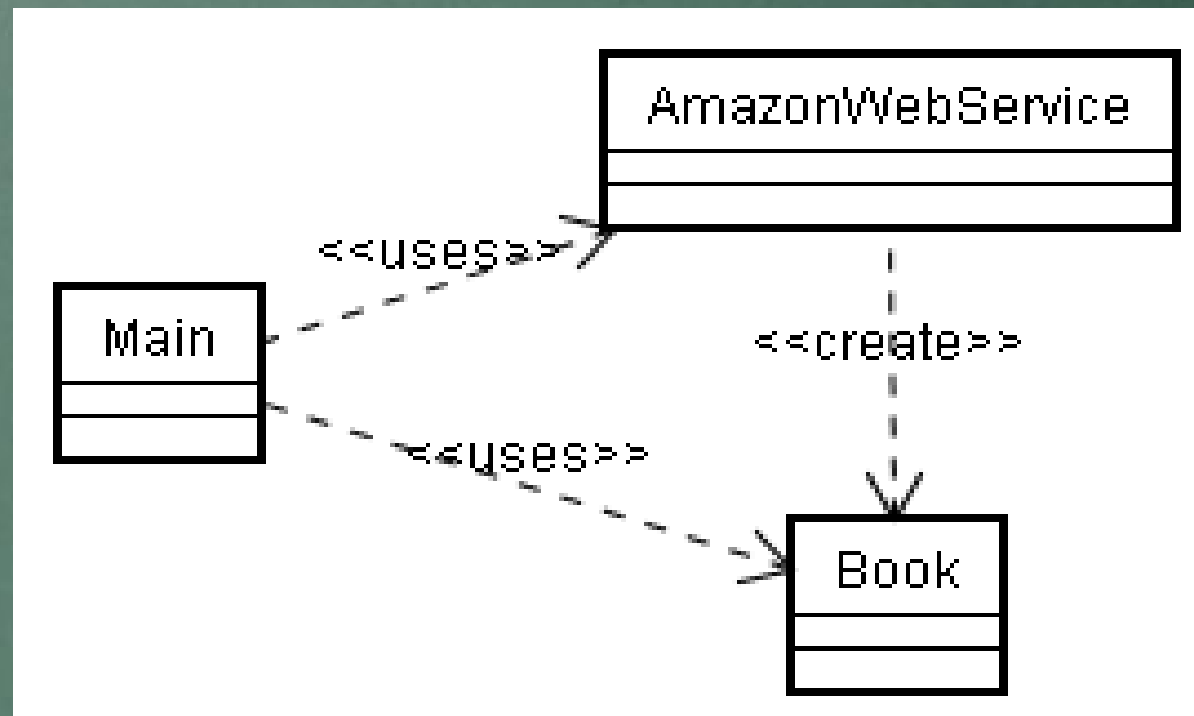
- 引数でASINを与えて起動
- AmazonマーケットプレイスをWebAPIで呼び出し、最も安い値段を検索する
- 標準出力する



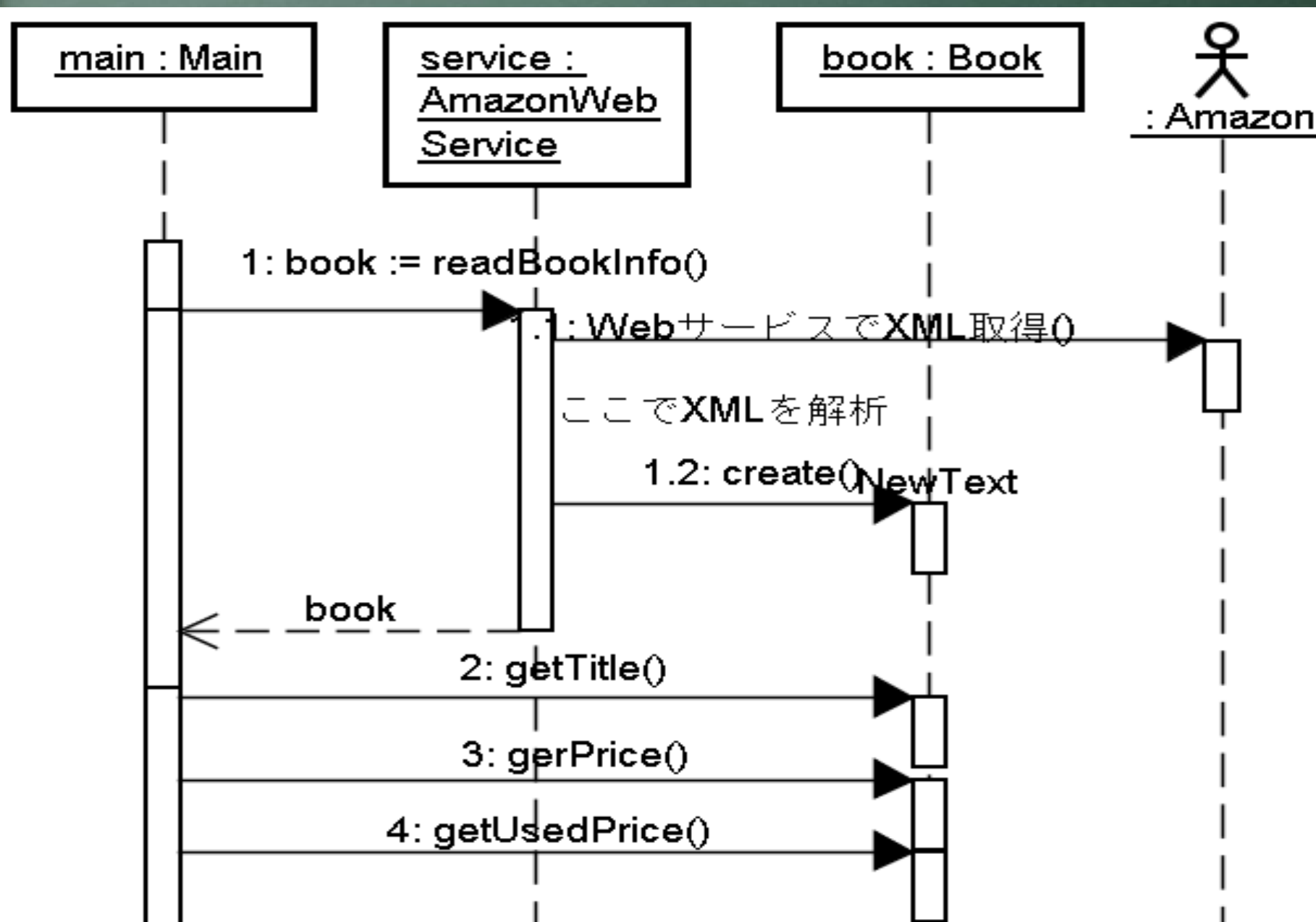
デモ



当初の構造



シーケンス図



ルールの適用

今回は、9つ全部は時間が足りないので、二つの視点でルール群に注目する

- 責務をクッキリさせる
- コラボレーションをバッチリさせる



責務クツキリに役立つルール

- すべてのエンティティを小さくすること
- すべてのプリミティブ型と文字列型をラップすること
- 一つのクラスにつきインスタンスは二つまでにすること?



Bookクラス

- クラス図
- 「インスタンス3つ」 「priceがint」



販売情報の抽出

- 「一つのクラスにつきインスタンスは二つまでにすること」
- (OfferSummaryのクラス図)



Moneyによるラップ

「すべてのプリミティブ型と文字列型をラップすること」

- Moneyのクラス図



最終的なBookクラス



利点

- 最初の時点では思いつかなかったクラスを抽出できた
- 「Book」が持っていた過剰な責務を適切に配分することができた。
 - MoneyのことはMoneyに



コラボレーションに関するルール

オブジェクトをより能動的にして、コラボレーションによって機能を実現する

関係するルール

- getter/setterを利用しない



なぜなら

getter/setterを定義したクラスの中身はスカスカになる

- 責務がgetter/setter利用側に寄ってしまう
- Tell, Don't Ask! 「たずねるな! 命じよ!」



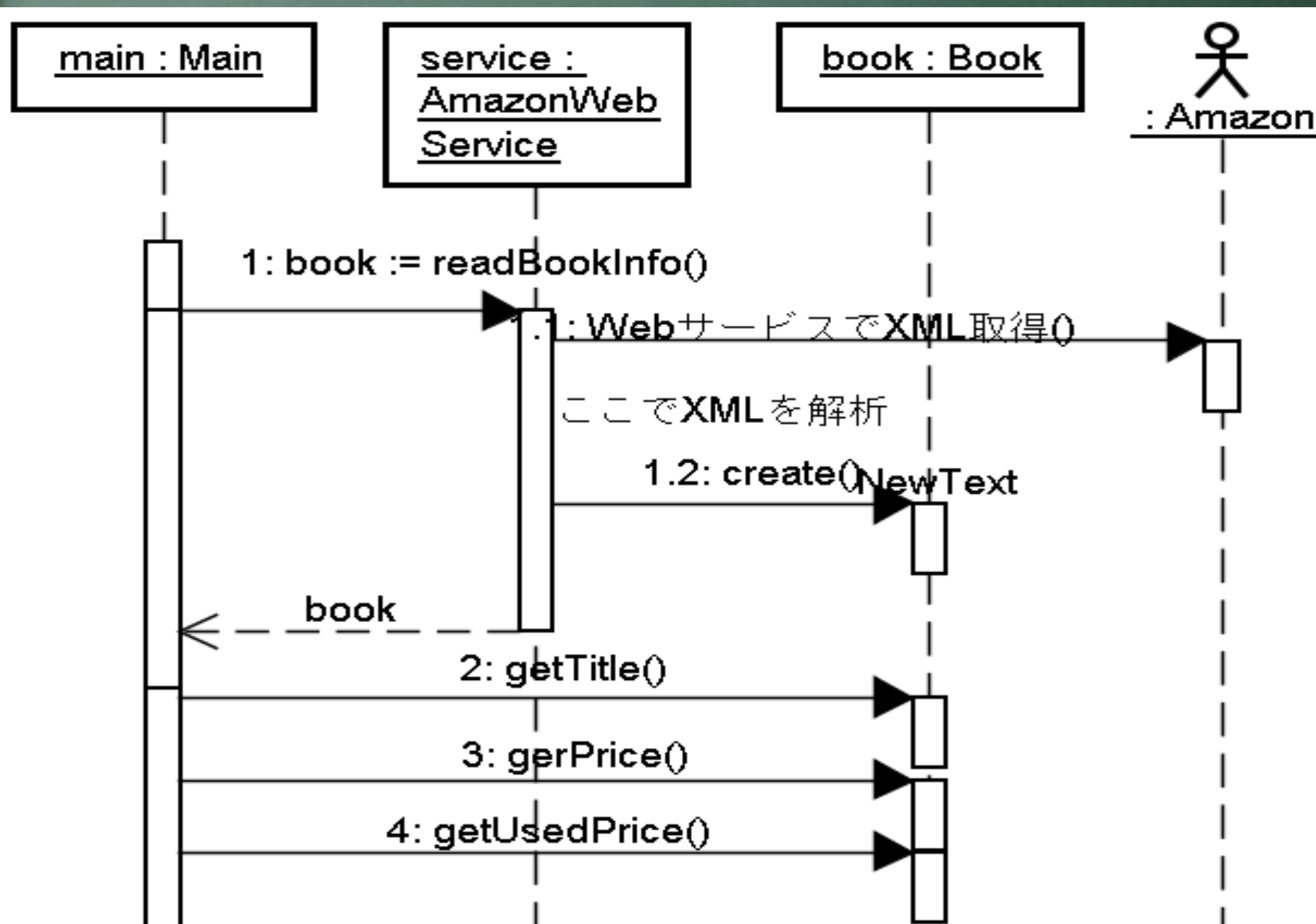
画面表示のとき・・・

Bookから情報取得しないとどうしようも無い

どうすればいい？



シーケンス図



考えられる戦略

- 妥協する(エクササイズ的にはNG)
- DTO的戦略
- ダブルディスパッチ戦略
- etc . .



DTO戦略

利用者(この例では画面)が要求する画面表示用オブジェクトをモデル側(この例ではBook)が生成して返却する




BookDTO

- 画面表示用のオブジェクトを、Bookクラスが
作って返却する



コード例

```
class Book {  
    public BookDTO represent() {  
        return  
            new BookDTO(  
                title, price(), usedPrice());  
    }  
}
```



Mainで表示する

```
AmazonWebService service =  
    new AmazonWebService();  
Book book = service.readBookInfo(args[0]);  
BookDTO dto = book.represent();  
System.out.println(dto);
```



欠点

DTO自体が振る舞いをもてずエクササイズの目的
と離れていく

getter/setterとどう違うの?



ダブルディスパッチ戦略

- 画面表示用のオブジェクトをBookに潜入させる
- それを受け取ったBook側が、必要な値を表示オブジェクトに設定する



クラス図



シーケンス図



コード例:Book側



コード例:画面表示用オブジェクト



コード例:Main



効果

- 設計アイデアを引き出した
- 属性を列挙するという振る舞いがBook側に移動した
- 画面表示の振る舞い自体は、ConsoleFormaterが持っている
- 責務が配分され、オブジェクトのコラボレーションにより機能が実現されている

その他の設計案

- Bookクラス自体に表示能力を持たせるなど
 - Book.showで画面表示する

-



明確な正解は無い

その場の「制約」をどうバランスさせるかによって、判断することが重要

- このような設計時の判断の集積はアーキテクチャになり、チームのコモンセンスになる



それ以外に検討できるところ

- AmaozonWebService



AmazonWebService

AmazonWebServiceには責務が3つある

- Amazon本家へのアクセス
- XMLの解析
- Bookオブジェクトの生成



「エンティティを小さく!」

- AmazonWebService
- BookFactoryの分離
- WebResourceクラスの抽出



最終的なクラス図



演習のまとめ

- エクササイズが設計に対する議論を引き出した
- 責務が分割されて、オブジェクト間のコラボレーションにより機能が実現されるようになった。



最後に



エクササイズで壁を超えよう

- とにかくきついで
- あらゆるテクニック、知識を総動員しろ
- 議論を巻き起こせ

ソフトウェアは人が作る!!



スペシャルサンクス