オブジェクト・ゲーム
～コードを使わず最速でオブジェクト指向がわかる方法～

2006年 6月29日
NECシステムテクノロジー
第一産業ソリューション事業部
牛尾　剛

ゲリラモデリング

開発現場の制約の中で最速で
オブジェクト指向を身につける教育手法

CIO

・オブジェクト指向がわかった
・最新アーキテクチャが理解できようになった

プロセス改善／生産革新部門

・一般の技術者でもオブジェクト指向がわかってもらえるぞ
・オブジェクト指向ベースの手法が使えるようになるな

管理者

・はじめてオブジェクト指向とPJでどう使うかがわかった
・すぐには導入しないけど、事例が増えたら使ってもいいかも・・・

普通の技術者

・はじめてオブジェクト指向とPJでどう使うかがわかった
・アーキテクチャが違った見方ができるようになった

出来る技術者

・独学でわかりにくいところがわかった
・後輩に教えるときも有効だな
・時間がかかるオブジェクト指向感覚がよくわかった

ゲリラモデリング

オブジェクト・ゲーム
→ コードなしで通常のコースでは身につかない オブジェクト指向の基礎スキルが身につく手法

オブジェクト指向基礎
→ オブジェクト指向の基礎とメリットを体感できる

オブジェクト指向感覚
→ オブジェクト指向PJを体験しないと理解できない感覚を習得する

完成型
→ オブジェクト指向PJはどういう作りなのかを考え方含めて理解する

万人向け

開発者向け

オブジェクト指向モデリング概念
基礎知識
→ 書籍や教育コースではあまり説明されないが 実プロジェクトで必須になる知識の吸収

ゲリラモデリングを実施すると
通常の教育もずっと理解しやすくなる

一般的な教育

UMLによるオブジェクト指向
分析設計コース

アジャイル開発
プロセス

コンポーネントモデリング
／SOAコース

試行プロジェクト実施

1．オブジェクト・ゲームとは

オブジェクトゲームとは

現場でオブジェクト指向をわかってもらうには

なぜオブジェクト指向は難しいのか

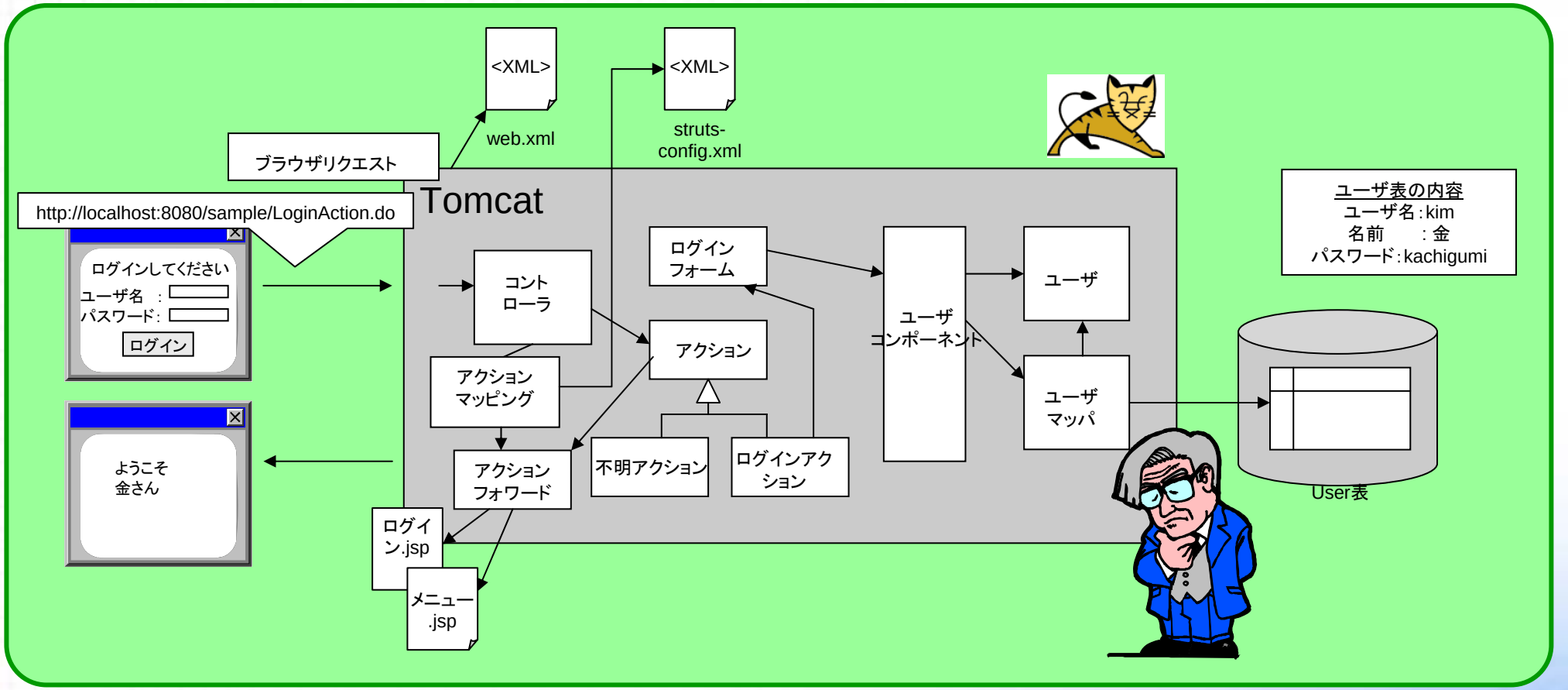
理論と体感	オブジェクト指向の考えや基礎的な理論の説明は単純だが メリットがわかるには体感(コーディング／デバッグ)が必要
完成型	オブジェクト指向の基本的な考えの中にはプロジェクトレベル の複雑なアプリケーションでしか体験できないことがある
前提知識の多さ	完成型を理解するためにはミドルウェアや言語など いろいろなことを理解が必要で大変

解決策

オブジェクト・ゲーム

- オブジェクト指向アプリケーションの動きや役割分担を理解するゲーム
- 皆さんにプログラムになってもらいます

こんなん理解できます？



オブジェクト・ゲームなら

前提知識なし4時間で理解OK！

本日のお断り

- 本来は少人数のワークセッションで完成型まで実施すると4時間かかります。
- 本日はオブジェクト・ゲームのやり方を理解して体感することに集中しましょう。
- やり方自体は簡単なので近くのオブジェクト指向を理解している人にやり方を教えてやってもらいましょう。

アイスブレイク

- 自己紹介
- チームリーダー

オブジェクト指向の開発

オブジェクト指向型開発

とは

システムを現実世界にある「もの」(オブジェクト)としてとらえる方法

- 日常生活で「もの」を扱うときと同じような感覚でシステム構築ができる
- 実際のアプリケーションでは現実世界にない「もの」もある

説明は難しい

習うより慣れよう

- 多くのパターンにふれるとわかります
- オブジェクト・ゲームなら体感できます

2．オブジェクト・ゲームのルール

クラス

クラス

このゲームのプログラムの単位です。カードで表します。

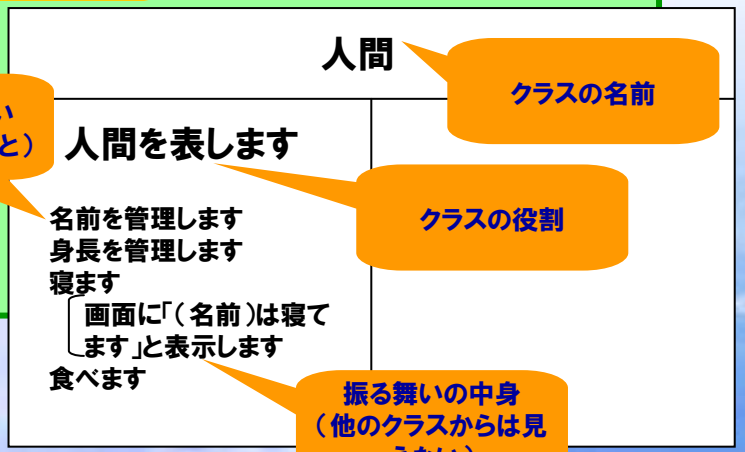
プログラムを役割分担ごとに分けたのがクラスです。

既存のアプリケーションやクラスが協調して１つの機能が動作します。

役割はもの・概念・その他で役割分担します。

例えば商品や従業員や受注などです。

その他はあとの演習で体験！



既存のアプリケーションや画面

既存のアプリケーションや画面やデータベースもカードにします。

アプリケーションの名前

Java

引数の名前のクラスを実行します

振る舞い（アプリケーションができること）

画面

牛尾が寝ています。

画面に表示する内容

オブジェクト化する

オブジェクト化する（newする）

クラスはオブジェクトにしないと動けません

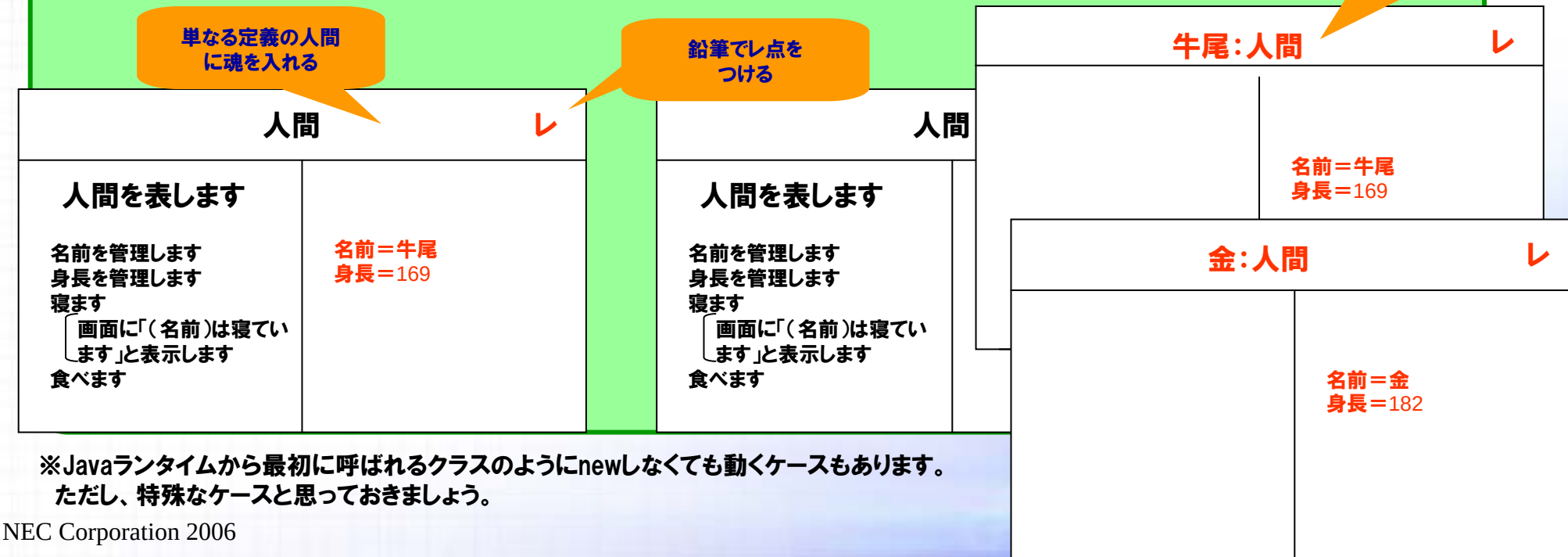
オブジェクトは一つのクラスから一つでも複数でも作れます

現実世界にある例でいくと
クラス：人間
オブジェクト：牛尾剛

単なる定義の人間
に魂を入れる

鉛筆でレ点を
つける

複数の時の
書き方

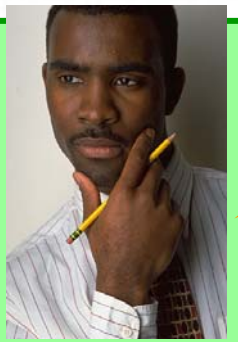


※Javaランタイムから最初に呼ばれるクラスのようにnewしなくても動くケースもあります。
ただし、特殊なケースと思っておきましょう。

クラスの使い方

クラス(オブジェクト)の使い方

- クラスは他のクラスから命令されないと動きません
- クラスは命令されると自分の役割分だけ働きます
- クラスはプログラム全体のことは意識しません
- 「○○を管理する」と書いていると○○のデータを保存および取得する役割があります
- 例えば人間に関することは人間クラスに書いています
- クラスは他のクラスの振る舞いだけ見えます。〔の中身や線の右側は見えてません



ルールが多くて面倒だけど、後でゲームをするとすぐおぼえられるよ

他のクラスから命令されないと動けない

人間	レ
人間を表します 名前を管理します 身長を管理します 寝ます 　画面に「(名前)は寝ています」と表示します 食べます	名前＝牛尾 身長＝169 この例だと名前と身長のデータを保存している

メッセージのパターン

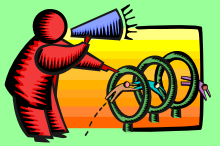
メッセージ(他のクラスへの命令)の方法

命令のみ行う場合

起立しろ！

自分の担当を指差す（ポイント）

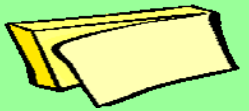
クラスの担当のひとの肩をたたいて命令します。



情報を渡して命令を行う場合

部長を役職として渡すので社員クラスください

他のクラスに渡したい情報を付箋に書いたのち渡しながら命令します。



クラス(のオブジェクト)を渡して命令を行う場合

②カード担当の人ごと渡す場合

①他のクラスにクラスカードごと渡して命令します。

②他のクラスにクラスカードをもった人ごと送って命令します。

①クラスに担当がない場合カードのみ

３．最初のプログラムの理解

ゲームのやり方

ゲームのやり方

- 鉛筆と消しゴムの準備 
- カードを並べる 
- 役割分担する 
- プログラムを実行する 
- シナリオを実行する 
- ゲームの結果とアプリケーションを比較する 

基本はカード毎に
1名。兼務あり。

実際のアプリケーション
を動かしてみる

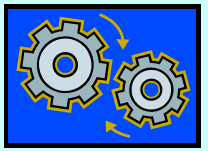
慣れるまでは同じ
シナリオを3回実行

解説を聞いて概念を
「体感」しよう

最初のプログラム

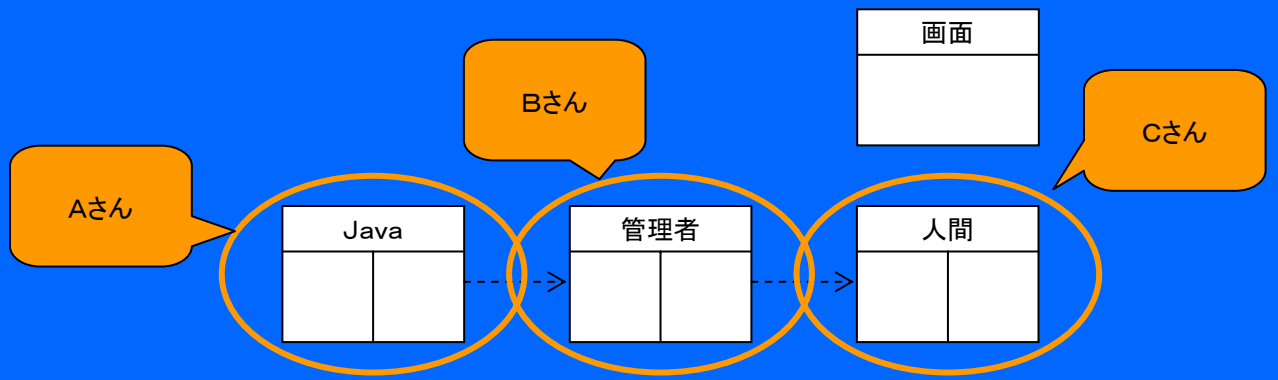
仕様と実行イメージ

人間クラスを動かしてみよう
クラスは他のクラスから命令されないと動けないので
「人間」に命令する役割の「管理者」クラスを作ってみた。



```
C:¥java Manager
牛尾が食べています。
牛尾が寝ています。
```

カードの並べ方と役割分担



※Javaから最初に命令されるクラスはnew不要です

カードを左図のように並べます

自分が受け持つカードを決めます

Javaカードの「実行する」を実行してみよう

※2人の時は「Java」と「管理者」を兼務および「人間」で行う

最初のプログラムの理解

Java

第一引数のクラスを実行します

画面

管理者

管理者を表します

実行する
- ・人間クラスを名前=牛尾 身長=169を渡してnewします。
- ・人間クラスに食べろと命令する
- ・人間クラスに寝ろと命令する

人間

人間を表します

名前を管理します
身長を管理します
寝る
- 画面に「(名前)は寝ています」と表示する

食べる
- 画面に「(名前)は食べています」と表示する

最初のプログラムの理解

クラスのカードをそのままプログラミング言語で書くと
プログラムが出来てしまいます。

ということは

- プログラミング言語でプログラムを書いたのと同じ効果が得られる
- プログラミング言語の知識なしでプログラム実行／デバッグ／仕様変更ができる
- プログラムを書くより早いので短時間に沢山体験できる
- クラスの役割を意識して実行できる

要は経験

適切なプログラムを沢山書いたのと
同じ効果を得られる

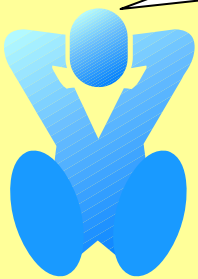
４．社長起立プログラム

社長起立プログラム　その1

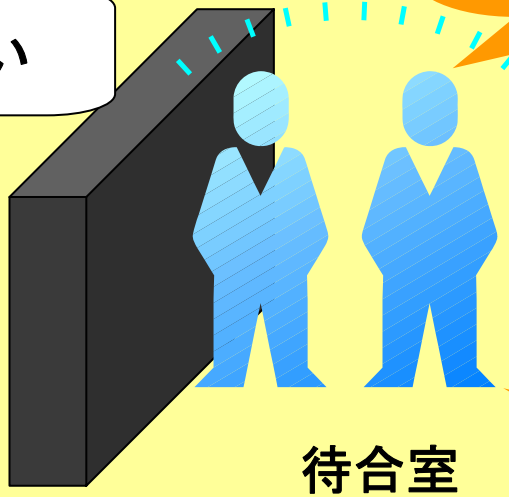
社長命令起立のイメージ図

社長は社員だと
思って起立命令を出す

立ちなさい



社長



待合室

実際に立つのは、
担当のAくん
主任のBさん
部長のCさん
でも担当も主任も部長も社員

立ち方はそれぞれ違う
担当：普通に立つ
主任：すばやく立つ
部長：だるそうに立つ

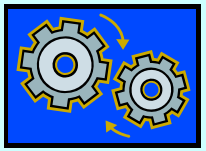
社長起立プログラム　その1

仕様と実行イメージ

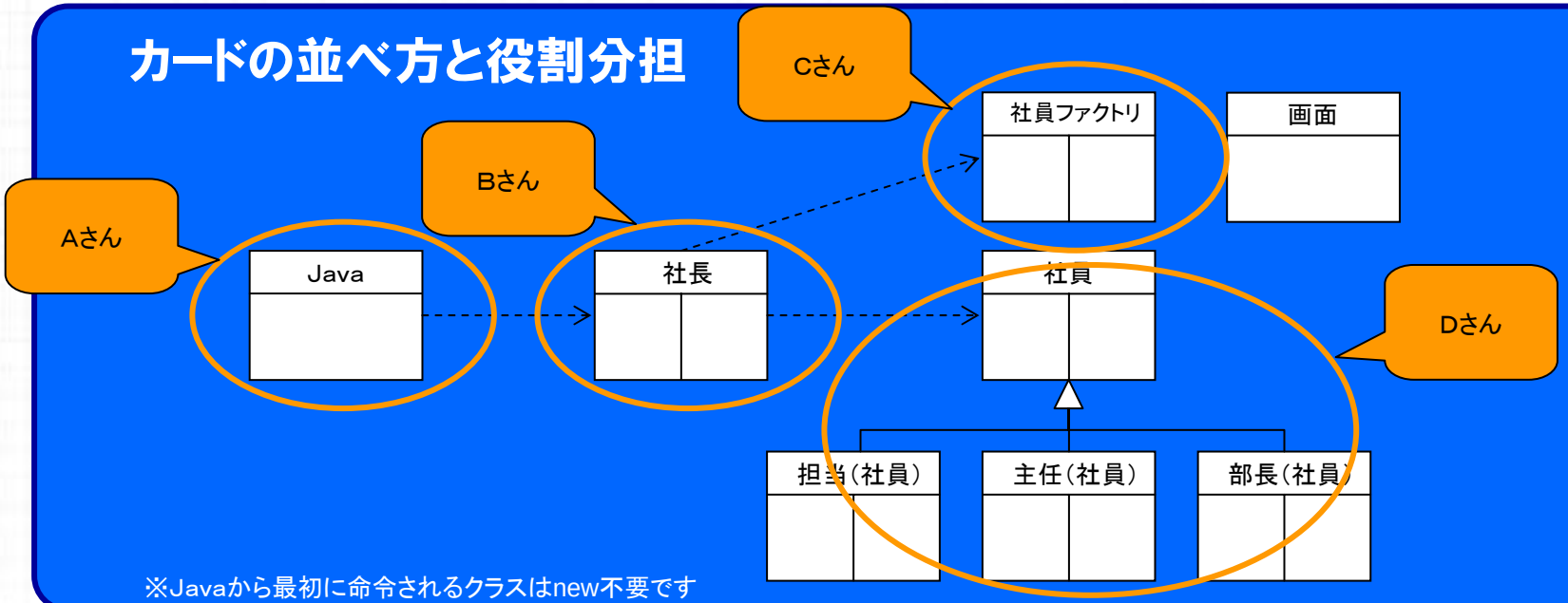
こんなプログラムを理解してみましょう。

```
C:¥java Shacho 担当
担当が普通に起立しました。
C:¥java Shacho 主任
主任がすばやく起立しました。
C:¥java Shacho 部長
部長がだるそうに起立しました。
```

```
C:¥java Shacho 課長
・・・実行結果はお楽しみ
```



カードの並べ方と役割分担



※Javaから最初に命令されるクラスはnew不要です

※2人の時は「Java」と「管理者」を兼務および「人間」で行う

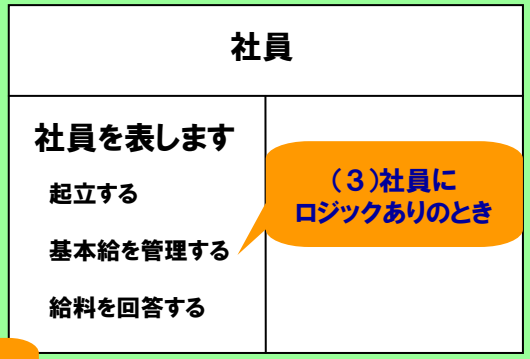
※人数が多いときは担当・主任・部長でそれぞれ一人でもよい

ゲームのルール

ルール：継承

例えば現実の世界の例でいくと
担当・主任・部長は社員であるという
関係にあるクラスで適用される。

担当は社員だし、
主任も社員だし、
部長も社員だ。

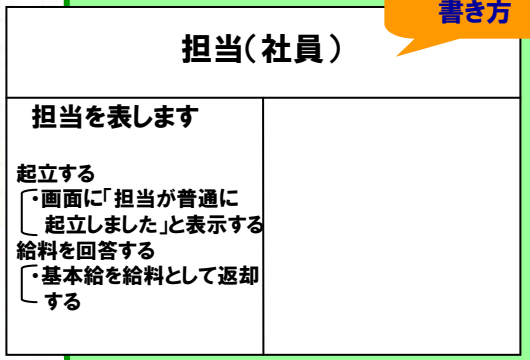


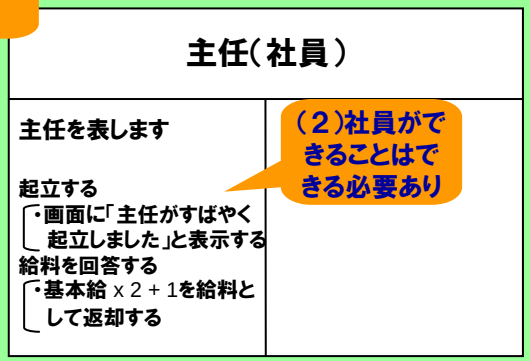
（１）担当・主任・部長は社員を継承している

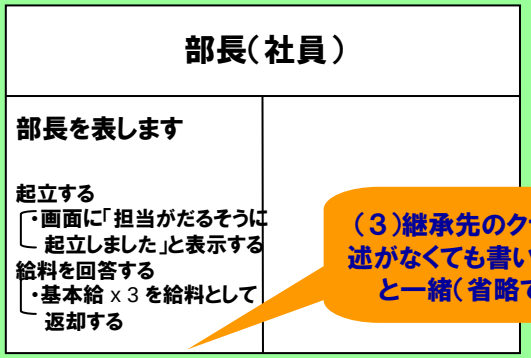
（２）社員ができることは担当・主任・部長ができないといけない

（３）社員に振る舞いのロジックが書いているときは社員を継承したクラスは記述が不要

（１）継承の書き方







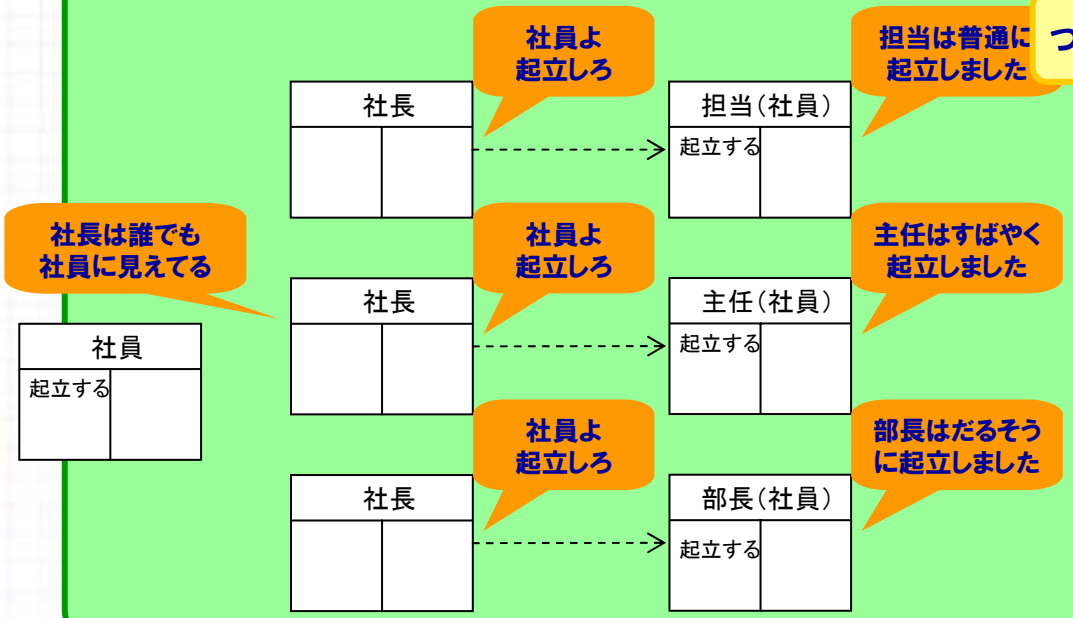
ゲームのルール

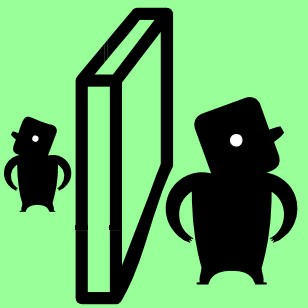
ルール：ポリモーフィズム（多態性）

継承ルールが適用されたカードは
ポリモーフィズムルールが使える。

担当・主任・部長は他のクラスから社員として扱うことができる。

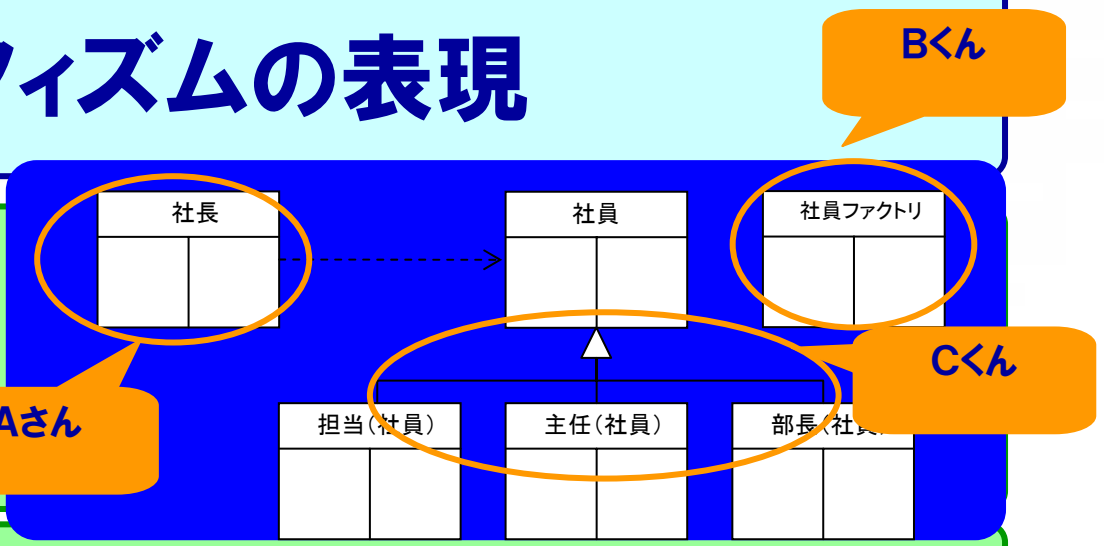
つまり継承先のクラスは継承元のクラスとして扱える。





ポリモーフィズム

ポリモーフィズムの表現

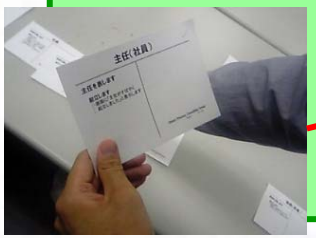


①ポイント

②主任をnew

社員ファクトは主任をnewします

③主任（社員）を返却



社員担当の人は「社員」カードをポイントし、「主任」カードを自分だけ見えるように

「社員」担当ごと「社長」に返却
他のメンバーは「社員」カードしか見えません

ゲームのルール

ルール：カプセル化

寝ろ

管理者

人間

人間を表します

名前を管理します
身長を管理します
寝る
　画面に「（名前）は寝ています」と表示する
食べる
　画面に「（名前）は食べています」と表示する

名前＝牛尾
身長＝169

仕様変更

食べろ

管理者

人間

人間を表します

名前を管理します
身長を管理します
寝る
　画面に「（名前）は寝ています」と表示する
食べる
　画面に「人間は食べています」と表示する

人間

人間を表します

名前を管理します
身長を管理します
寝る
　画面に「（名前）は寝ています」と表示する
食べる
　データベースを番号＝1で検索して結果を返す

データベース

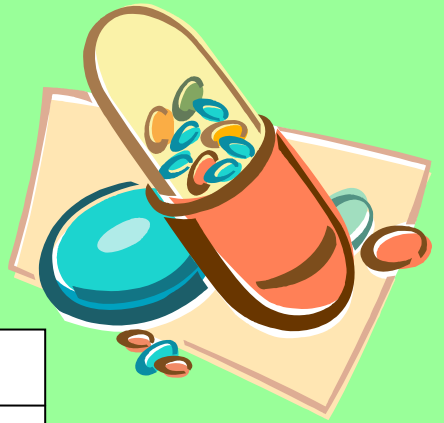
番号1　もぐもぐもぐ
番号2　むしゃむしゃ

中身が変わっても意識しなくていい設計にすること

カードは他のカードの〔の中身や線の右側の内容を意識しません。

ただ、どんな振る舞いをしてくれるだけを知っていて命令します。

クラスは自分の役割だけ果たして命令したら後のことはしらない。



NEC

社長起立プログラム　その1

Java

第一引数のクラスを実行します

実行時例外を捕まえてメッセージを画面に出します。

画面

社長

社長を表します

実行する
- ・社員ファクトリをnewする
- ・社員ファクトリに第二引数を役職として渡して社員クラスをもらう
- ・社員に起立しろと命令する

社員ファクトリ

社員をnewする役割

役職にあった適切な社員クラスを返却
- ・担当の時は担当クラスをnewして返却
- ・主任の時は主任クラスをnewして返却
- ・部長の時は部長クラスをnewして返却

適切な社員クラスが無い場合は実行時例外に「該当する役職はありません」とセットして投げる

社長起立プログラム　その1

社員	
社員を表します 起立する	

部長（社員）	
部長を表します 起立する ・画面に「部長がだるそうに起立しました」と表示する	

担当（社員）	
担当を表します 起立する ・画面に「担当が普通に起立しました」と表示する	

主任（社員）	
主任を表します 起立する ・画面に「主任がすばやく起立しました」と表示する	

社長起立プログラム　その1

実行時例外（例外）	
実行時の例外を表す エラーメッセージを管理する	

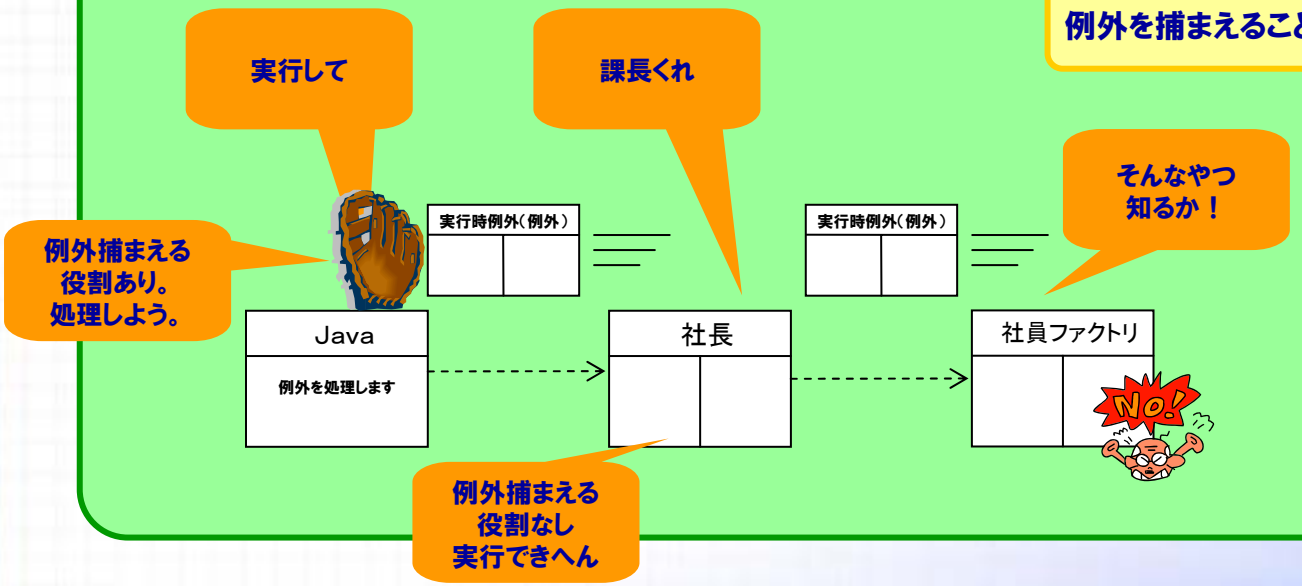
ゲームのルール

ルール：例外

プログラムは実行できなかったり問題が発生した場合、クラスは例外を投げることができる。

例外は仕事をお願いされたクラスに投げられます。

例外を捕まえることができるところが例外を処理します。



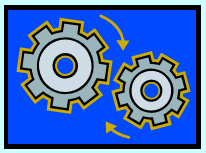
社長起立プログラム　その2

仕様と実行イメージ

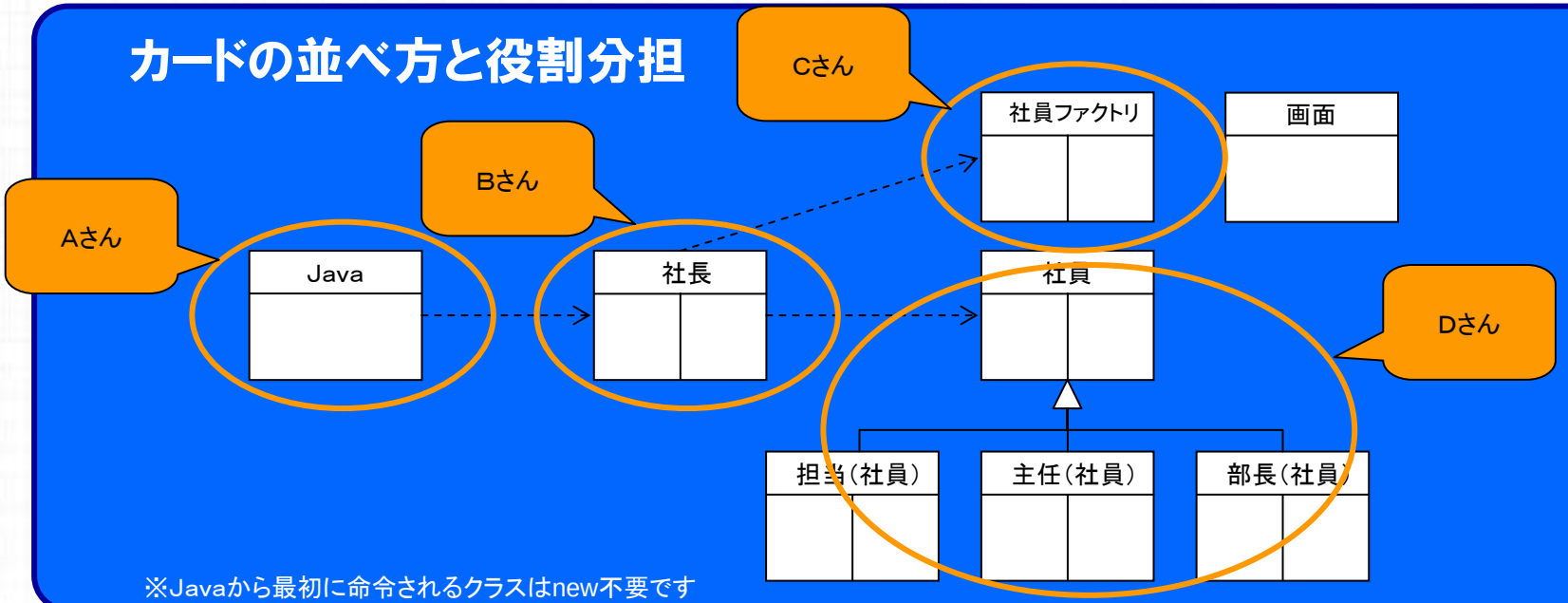
仕様追加してみよう

担当の場合の給料計算
　給料＝基本給
主任の場合の給料計算
　給料＝基本給 x 2 + 1
部長の場合の給料計算
　給料＝基本給 x 3

```
C:¥java Shacho 担当 100
担当が普通に起立しました。
社員の給料は100円です。
C:¥java Shacho 主任 100
主任がすばやく起立しました。
社員の給料は201円です。
C:¥java Shacho 部長
部長がだるそうに起立しました。
社員の給料は300円です。
```



カードの並べ方と役割分担

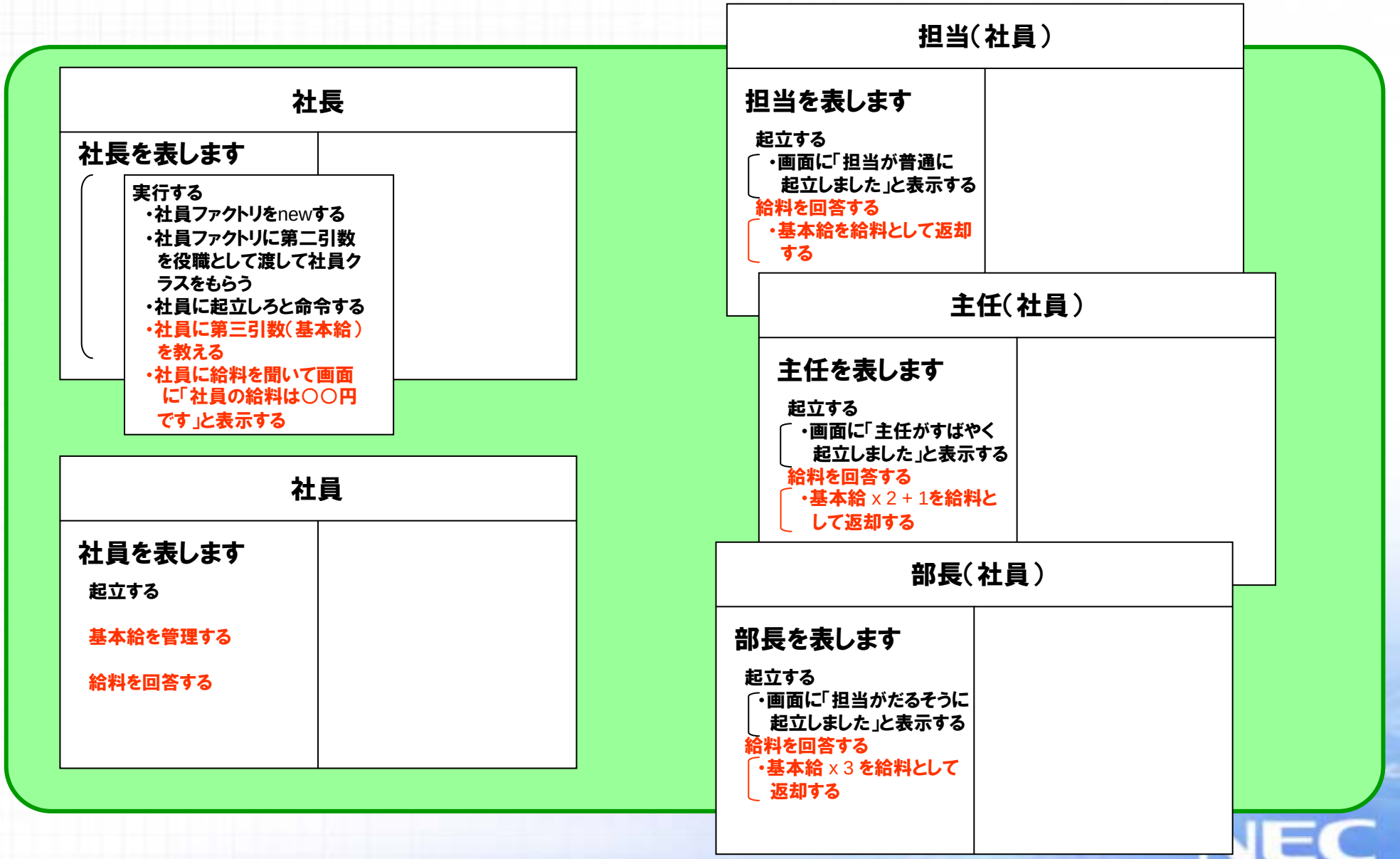


※Javaから最初に命令されるクラスはnew不要です

※2人の時は「Java」と「管理者」を兼務および「人間」で行う

※人数が多いときは担当・主任・部長でそれぞれ一人でもよい

社長起立プログラム　その2



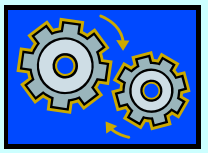
社長起立プログラム　その3

仕様と実行イメージ

クイズ：仕様を追加しよう
どうすればいい？

担当の場合の給料計算
　給料＝基本給
主任の場合の給料計算
　給料＝基本給 x 2 + 1
部長の場合の給料計算
　給料＝基本給 x 3
取締役の場合の給料計算
　給料＝基本給 x 4 + 2

```
C:¥java Shacho 担当 100
担当が普通に起立しました。
社員の給料は100円です。
C:¥java Shacho 主任 100
主任がすばやく起立しました。
社員の給料は201円です。
C:¥java Shacho 部長 100
部長がだるそうに起立しました。
社員の給料は300円です。
C:¥java Shacho 取締役 100
取締役がそれなりに起立しました。
社員の給料は402円です。
```



カードの並べ方と役割分担

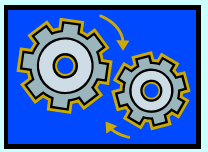
社長起立プログラム　その3

仕様と実行イメージ

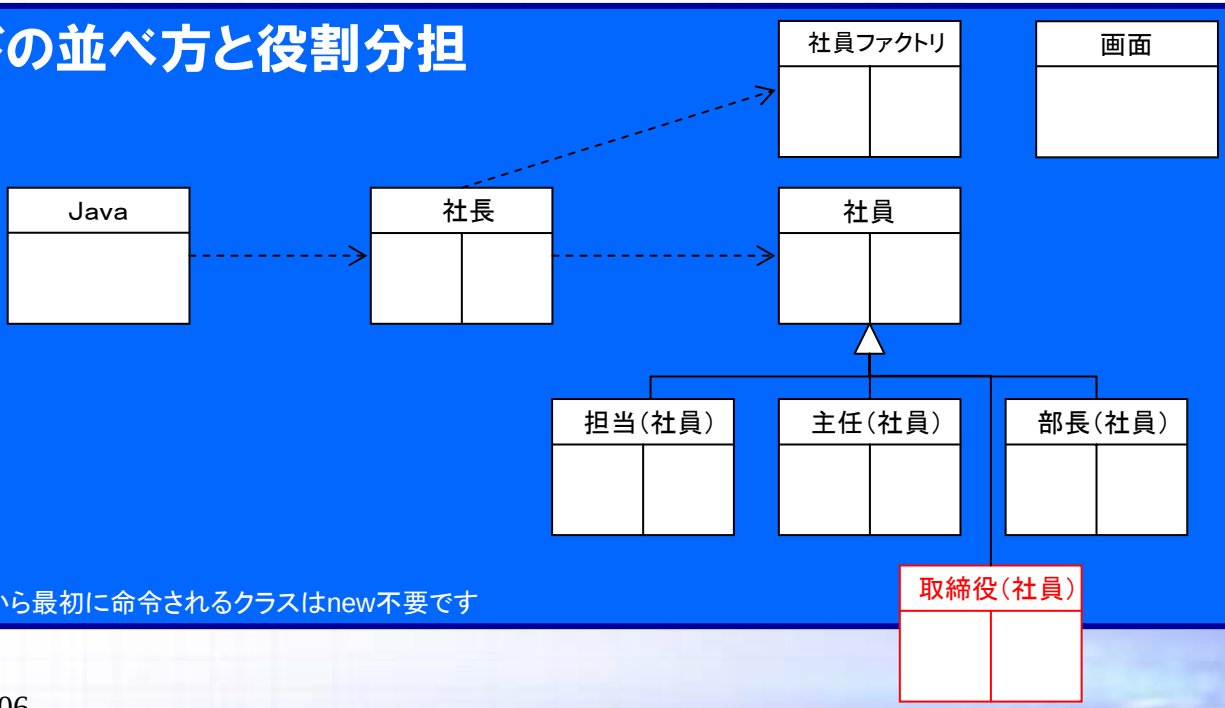
こうですね

担当の場合の給料計算
　給料＝基本給
主任の場合の給料計算
　給料＝基本給 x 2 + 1
部長の場合の給料計算
　給料＝基本給 x 3
取締役の場合の給料計算
　給料＝基本給 x 4 + 2

```
C:¥java Shacho 担当 100
担当が普通に起立しました。
社員の給料は100円です。
C:¥java Shacho 主任 100
主任がすばやく起立しました。
社員の給料は201円です。
C:¥java Shacho 部長 100
部長がだるそうに起立しました。
社員の給料は300円です。
C:¥java Shacho 取締役 100
取締役がそれなりに起立しました。
社員の給料は402円です。
```



カードの並べ方と役割分担



※Javaから最初に命令されるクラスはnew不要です

社長起立プログラム　その3

社員ファクトリ

社員をnewする役割

役職にあった適切な社員クラスを返却
- 担当の時は担当クラスをnewして返却
- 主任の時は主任クラスをnewして返却
- 部長の時は部長クラスをnewして返却
- 取締役の時は取締役クラスをnewして返却

適切な社員クラスが無い場合は実行時例外に「該当する役職はありません」とセットして投げる

取締役（社員）

取締役を表します

起立する
- 画面に「取締役がそれなりに起立しました」と表示する

給料を回答する
- 基本給 x 4 + 2 を給料として返却する

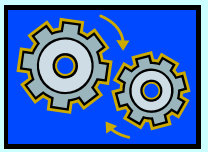
社長起立プログラム　その4

仕様と実行イメージ

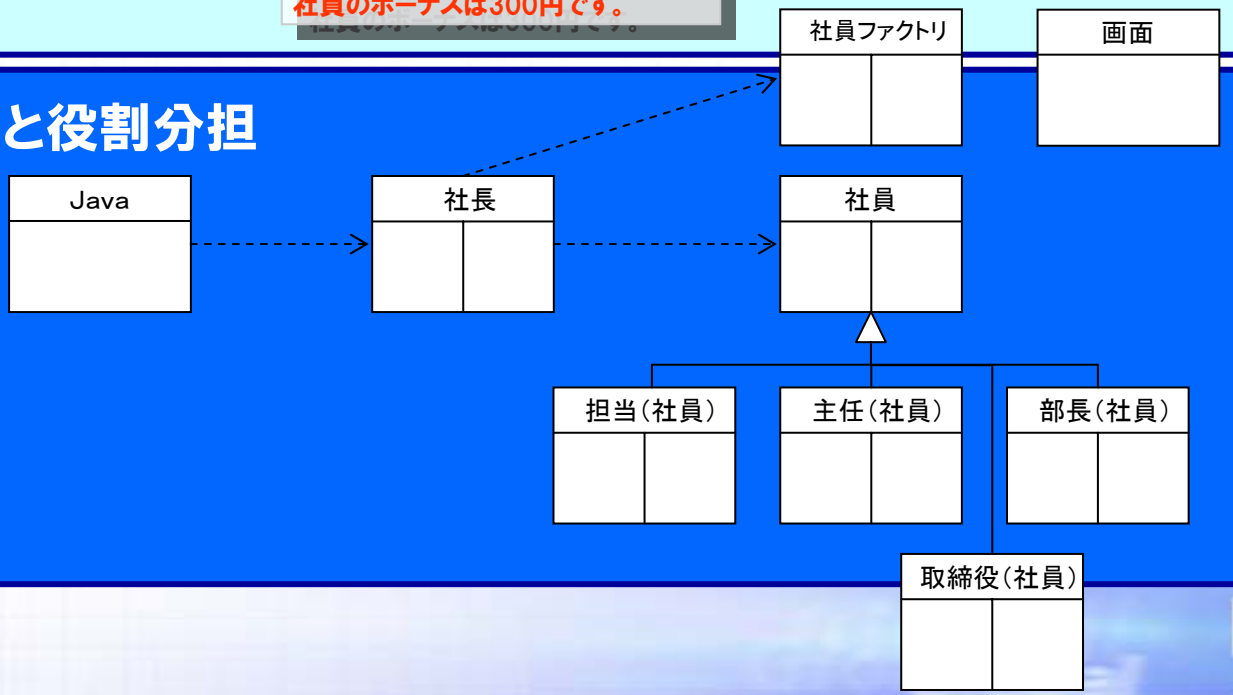
仕様を追加しよう

ボーナス計算　全社員
　ボーナス　＝　基本給 x 3

```
C:¥java Shacho 担当 100
担当が普通に起立しました。
社員の給料は100円です。
社員のボーナスは300円です。
C:¥java Shacho 主任 100
主任がすばやく起立しました。
社員の給料は201円です。
社員のボーナスは300円です。
C:¥java Shacho 部長 100
部長がだるそうに起立しました。
社員の給料は300円です。
社員のボーナスは300円です。
C:¥java Shacho 取締役 100
取締役がそれなりに起立しました。
社員の給料は402円です。
社員のボーナスは300円です。
```



カードの並べ方と役割分担



社長起立プログラム　その4

社員

社員を表します

起立する
基本給を管理する
給料を回答する
ボーナスを回答する
・基本給 x 3をボーナスとして返却する

担当でも主任でも部長でもボーナスは基本給x3なので、「社員」というものはボーナスx3と考えられる。（つまり社員に書くべき）

社長

社長を表します

実行する
・社員ファクトリをnewする
・社員ファクトリに第二引数を役職として渡して社員クラスをもらう
・社員に起立しろと命令する
・社員に第三引数（基本給）を教える
・社員に給料を聞いて画面に「社員の給料は○○円です」と表示する
・社員にボーナスを聞いて画面に「社員のボーナスは○○円です」と表示する

5．まとめ

まとめ

オブジェクト・ゲームの効能

- コードなしでオブジェクト指向の基礎がわかりメリットが体感できる
- 複雑なアーキテクチャがとてもわかりやすくなる
- プロジェクトレベルの完成型アプリケーションを短時間で経験できる
- プロジェクトに参加しないとわからないオブジェクト指向の基礎や感覚がわかる
- 短時間で優れた設計のパターンとメリットをたくさん経験できる
- 概念やパターンを「実例」から学ぶことができる

まとめ

オブジェクト指向習得のハードルを大幅に下げます

OPCG公認カード

オブジェクト・ゲーム硬式カード

OPCG公認
硬式オブジェクト・ゲームカード

印刷用硬式カード
L！FE情報カード
A6　無地

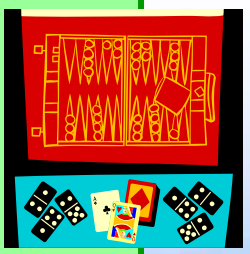
手書き用硬式カード
L！FE情報カード
A6　横罫

発展途上です

ゲームアイデアがあれば教えてね！

いろんな人の意見で発展中です。

引数ヲ付箋デ表現セヨ	チェンジビジョン勤務 平鍋　健児さん
カードはPPTで作ってA6で印刷ね	大阪在住 永野＆駒井さん
ポリモーフィズムは片手で隠す	NECシステムテクノロジー 勤務　岸本さん　原案
Newしたらプログラム域に付箋はる／色は重要	大阪在住 渡辺幸三さん



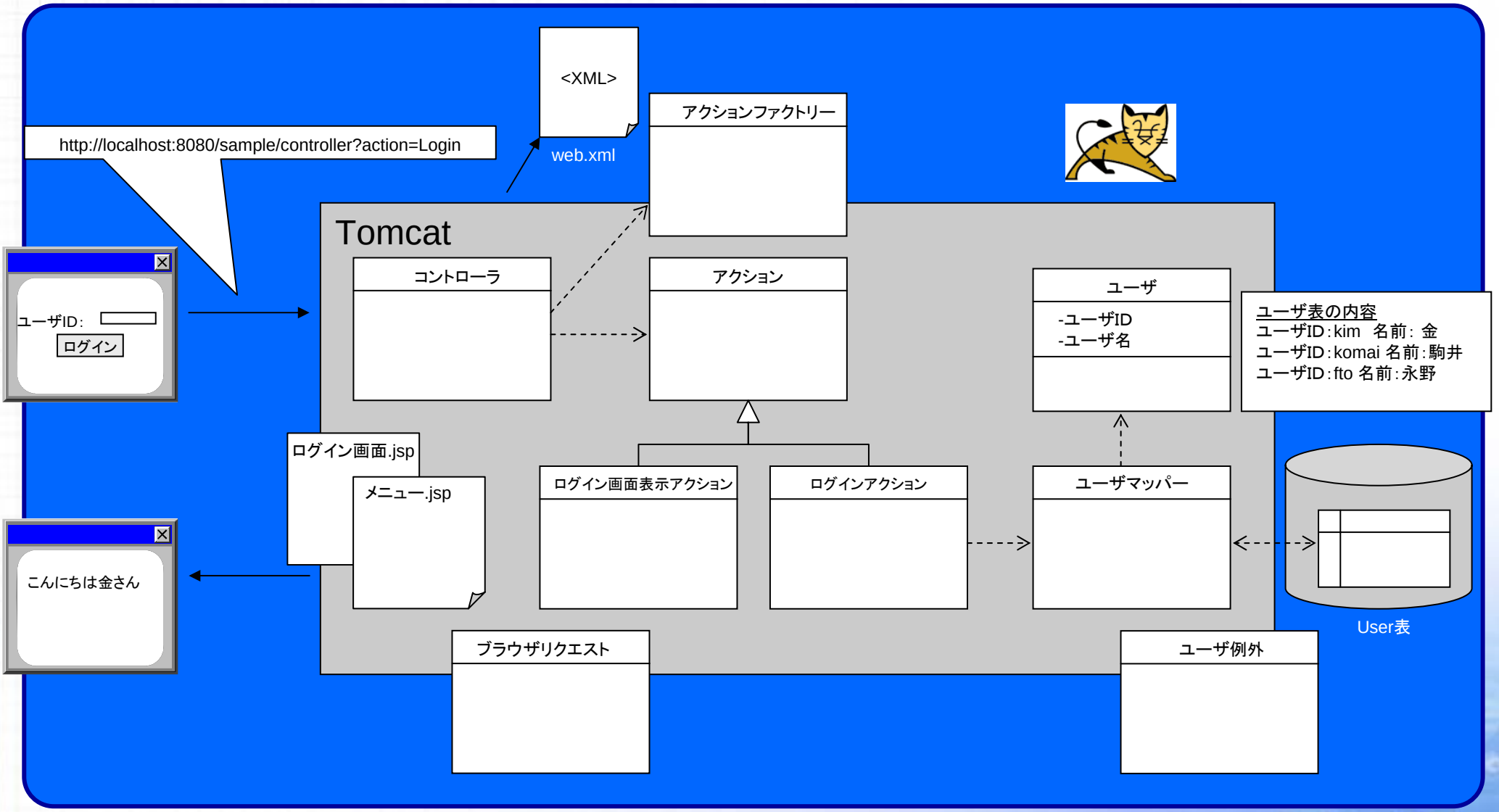
ご体感ありがとうございました。

オブジェクト・ゲームに関するお問い合わせは牛尾まで。
メールアドレス：t-ushio@bp.jp.nec.com

NEC

付録．本格的なＷｅｂアプリケーション
ＰＪレベルのＷｅｂアプリケーション

本格的なWebアプリケーション



本格的なWebアプリケーション

TOMCAT

アプリケーションサーバ

・ブラウザで入力したデータとURLに書かれたデータをブラウザリクエストに格納する
・設定ファイル「web.xml」を見てURLに対応するクラスにブラウザリクエストをわたして実行する

ブラウザリクエスト

ブラウザからのリクエストを表すクラス

ブラウザ情報を管理する

名前＝オブジェクトのセットを管理する

コントローラ

すべてのリクエストを受け付けて割振る役割

実行する

・ブラウザリクエストにactionという名前で格納されたデータをアクションファクトリーに渡してアクションをもらう
・ブラウザリクエストを渡してアクションを実行する

設定ファイル「web.xml」

/controllerというURLが要求されたら「コントローラ」クラスを実行すること

本格的なWebアプリケーション

アクション

ボタンを押す、URLと入力するなどの動作を表す

実行する

アクションファクトリー

アクションをnewして返却する役割

アクションを返却する
- データがviewLoginFormの場合ログイン画面表示アクションを返す
- データがLoginの場合ログインアクションを返す

ログイン画面表示アクション（アクション）

URLを入力してログイン画面を出すアクション

実行する
- ログイン画面.jspに制御を移す

ログインアクション（アクション）

ログインボタンを押したアクションを表す

実行する
- ブラウザリクエストからユーザIDを取得する
- ユーザマッパをnewする
- ユーザマッパからユーザIDに該当するユーザクラスをもらう
- ユーザクラスをブラウザリクエストに「user」という名前で格納する
- メニュー画面.jspに制御を移す

ユーザ例外を処理する
- ユーザ例外を受け取ったときはログイン画面.jspに制御を移す

※「○○.jspに制御を移す」の記述がある場合、TOMCATがアクションからブラウザリクエストをjspにわたして、jspを実行します。

本格的なWebアプリケーション

ユーザ

ユーザを表す

ユーザIDを管理する
ユーザ名を管理する

データベース

User表は以下のデータを持っている

ユーザID:kim　　ユーザ名: 金
ユーザID:komai　ユーザ名:駒井
ユーザID:fto　　ユーザ名:永野

ユーザマッパー

ユーザに関するデータベースアクセスを行う

ユーザIDに該当するユーザを返却する
- データベースをユーザIDで検索して、データをユーザクラスに格納する
- ユーザクラスを返却する

ユーザIDに該当するユーザが無い場合は「ユーザ例外」にを投げる

ユーザ例外(例外)

ユーザに関する例外を表す

本格的なWebアプリケーション

ログイン画面.jsp

ログイン画面を表示する役割

メニュー画面.jsp

メニュー画面を表示する役割

ブラウザリクエストの「ユーザ」の名前で格納されているオブジェクトのユーザ名をつかって、「こんにちは○○さん」と表示する

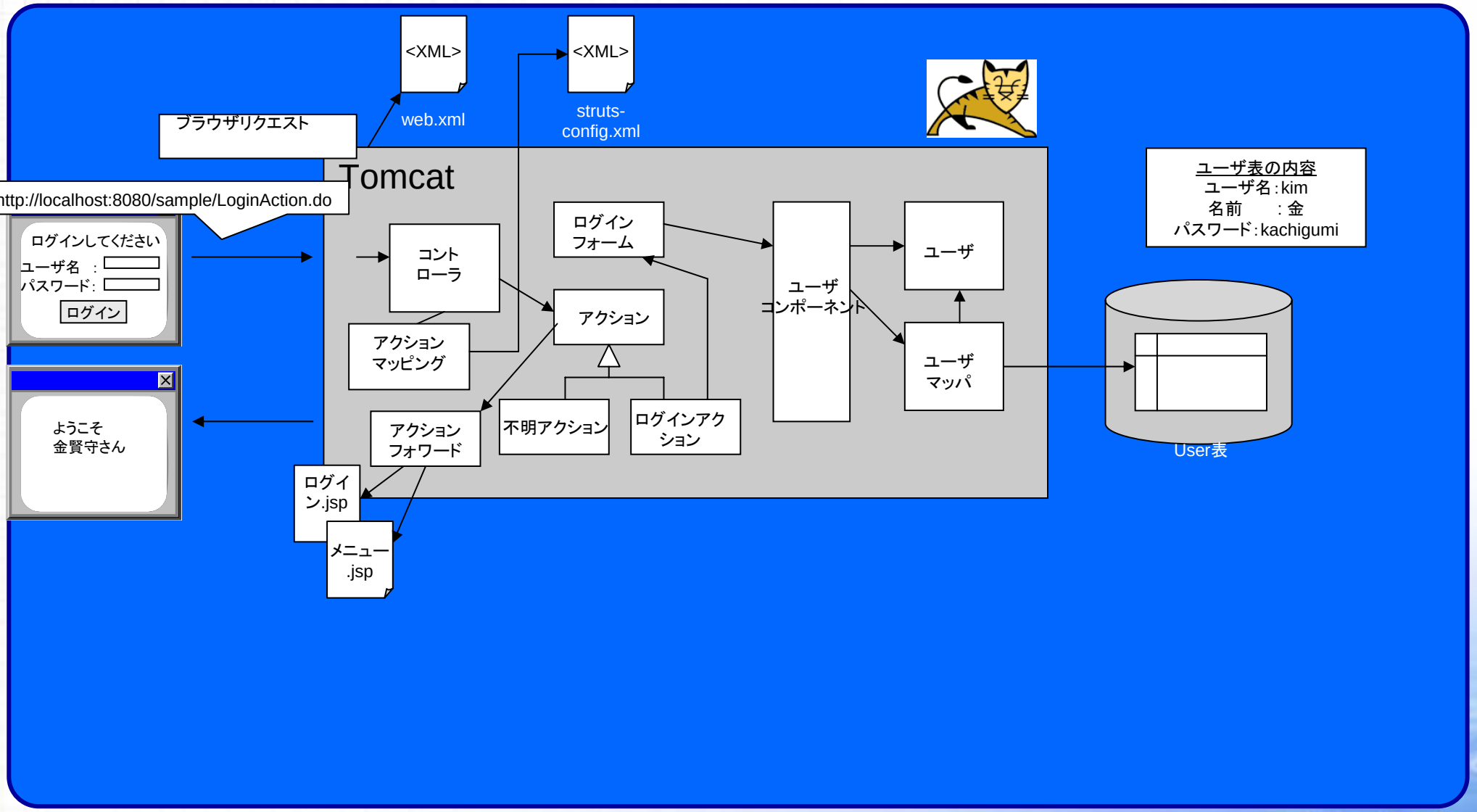
本格的なWebアプリケーション（隠しカード）

不明アクション（アクション）	
該当アクションなしのときのアクション 実行する ・アクションなし.jspに制御を移す	

アクションなし.jsp
アクションが見つからないときの画面の役割 「要求されたページがみつかりません。』と表示する

アクションファクトリー	
アクションをnewして返却する役割 アクションを返却する ・データがviewLoginFormの場合ログイン画面表示アクションを返す ・データがLoginの場合ログインアクションを返す	該当するアクションが無い場合は「不明アクションを返却する

PJレベルのWebアプリケーション



PJレベルのWebアプリケーション

TOMCAT

アプリケーションサーバ

・ブラウザで入力したデータとURLをブラウザリクエストに格納する
・設定ファイル「web.xml」を見てURLに対応するクラスを実行する

ブラウザリクエスト

ブラウザからのリクエストを表すクラス

ブラウザ情報を管理する

名前＝オブジェクトのセットを管理する

コントローラ

すべてのリクエストを受け付けて割振る役割

実行する
・アクションマッピングをnewする
・ブラウザリクエストからURLを取得する
・アクションマッピングにURLを渡して該当するアクション名を教えてもらう
・アクションマッピングにURLを渡して該当するアクションフォーム名を教えてもらう
・アクションとアクションフォームをnewする
・ブラウザリクエストに格納されたフォームのデータをアクションフォームに格納する
・アクションを実行しアクションフォワードを受け取る。その際ブラウザリクエストとアクションマッピングとアクションフォームをアクションに渡す
・アクションフォワードからフォワード先のJSP名を取得して制御を移す

設定ファイル「web.xml」

○○.doというURLが要求されたら「コントローラ」クラスを実行すること

PJレベルのWebアプリケーション

ログインフォーム（アクションフォーム）

ログイン画面関連のデータを管理するクラス

ユーザコードを管理する
ユーザ名を管理する
パスワードを管理する
ログインする
- ユーザサービスをnewする
- ユーザサービスにユーザコードとパスワードを渡してユーザクラスをもらう
- ユーザクラスのユーザ名をこのクラスのユーザ名にセットする

設定ファイル「struts-config.xml」

LoginActionの場合は以下のようにすること
- アクションフォームはログインフォームを使用する
- アクションはログインアクションを使用する
- フォワード文字が「menu」の場合はメニュー.jsp
- フォワード文字が「login」の場合はログイン.jsp

LoginViewの場合は以下のようにすること
- アクションフォームはログインフォームを使用する
- アクションは不明アクションを使用する
- フォワード文字が「success」の場合はログイン.jsp

アクション

ボタンを押す、URLと入力するなどの動作を表す

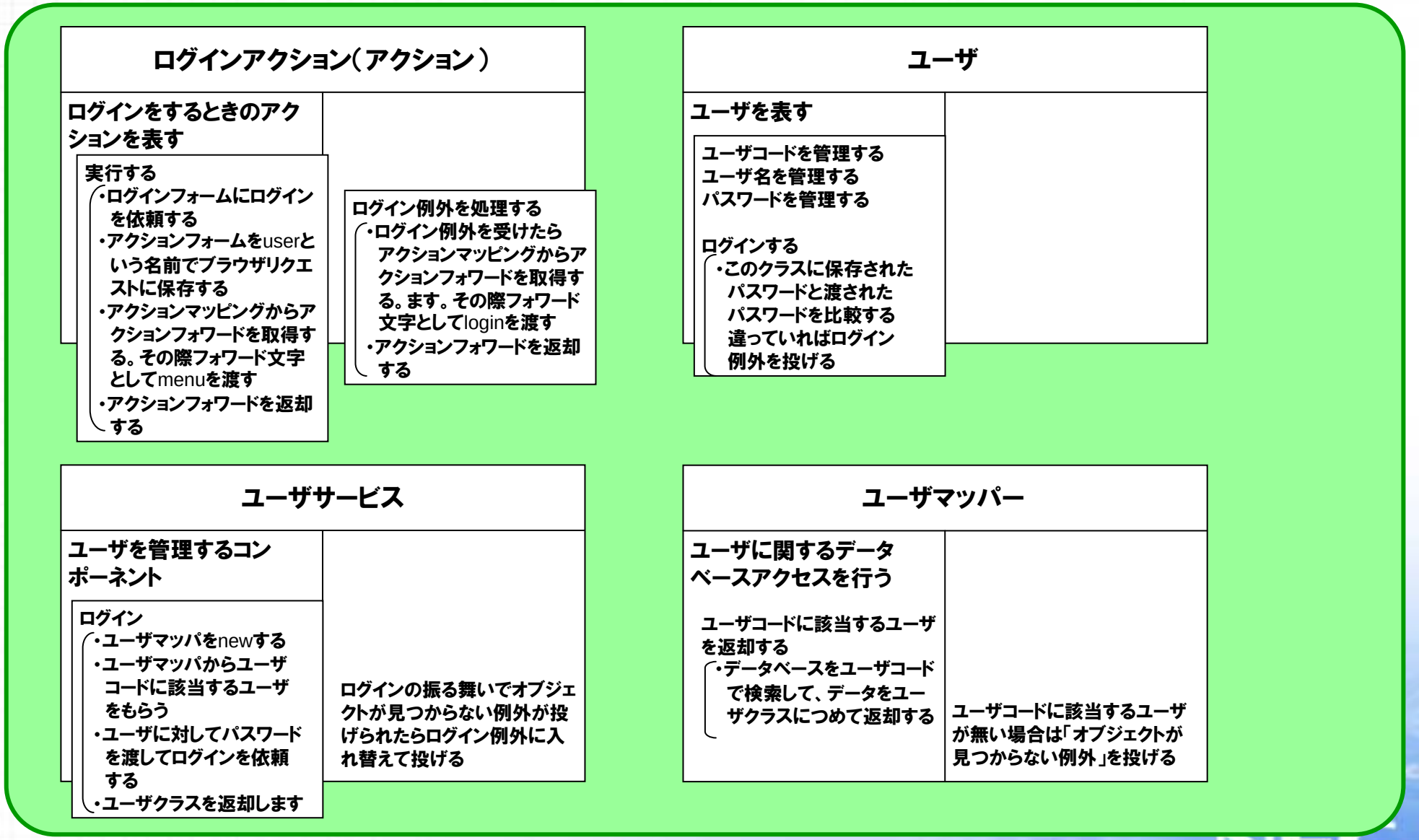
実行する

不明アクション（アクション）

なにもしないアクションを表す

実行する
- アクションマッピングからアクションフォワードを取得する。その際フォワード文字としてsuccessを渡す
- アクションフォワードを返却する

PJレベルのWebアプリケーション



PJレベルのWebアプリケーション

データベース
User表は以下のデータを持っている ユーザコード：kim　ユーザ名：金 賢守　パスワード：kachigumi ユーザコード：komai ユーザ名：駒井 聡　パスワード：20 ユーザコード：fto　ユーザ名：永野 太輔　パスワード：mickey

ログイン例外（例外）	
ログインに関する例外を表す	

オブジェクトが見つからない例外（例外）	
オブジェクトが見つからないときに投げる例外	

PJレベルのWebアプリケーション

ログイン画面.jsp

ログイン画面を表示する役割

アクションフォームのユーザコードを表示する
アクションフォームのパスワードを表示する

メニュー画面.jsp

メニュー画面を表示する役割

ブラウザリクエストの「ユーザ」の名前で格納されているオブジェクトの名前をつかって、「いらっしゃいませ○○さん」と表示する

アクションマッピング

マッピングの情報を管理する役割

該当するアクションを答える
- struts-config.xmlを参照して該当するアクションを答える

該当するアクションフォームを答える
- struts-config.xmlを参照して該当するアクションフォームを答える

アクションフォワードを返却する
- アクションフォワードをnewする
- struts-config.xmlを参照してフォワード文字に該当するJSP名をアクションフォワードに格納する
- アクションフォワードを返却する

アクションフォワード

フォワード情報を管理する役割

遷移先のJSP名を管理する