

# オブジェクト指向

2018/7/3(火) プログラミングIV 第十二回  
福井大学 工学研究科 情報・メディア工学専攻  
長谷川達人



# 演習問題の解説

- HP上にアップロードした.
- 点数に**不服がある人**は個別に申し立ててください.
- 今回, 採点にさける時間が少なかったなので, 点数の不服があるかもしれません. ご指摘ください.
- 前回の解説を今から行います.

# 質問への回答

- サードパーティ製とは？
  - 第三者製ってこと。Java標準外が提供するという意味で。
- ステップイン/ステップオーバーがよくわからなかった。
  - プログラムの実行カーソルを一つ進めるだけ。
  - ただし、次がメソッドの場合、その中に入るのがインで、メソッド内をまとめて実行するのがオーバー。インで入った途中からやっぱオーバーしなくなったときには、ステップアウト。
- ステップインは標準APIの中にも入っていく？
  - 試してみるとよいが、F5を連打するとどんどん中に入っていく。
- テキスト出力すると情報漏えいの可能性は？対策ある？
  - 暗号化かなあ。着眼点はステキ。

# 質問への回答

- デバッガが搭載されていないことはある？
  - C言語はデバッガなしじゃなかった？
  - 統合開発環境には搭載されていることが多い。
- デバッガを使っても見つからないエラーは？
  - デバッガはあくまで人間のデバッグをサポートするツールなので、使いこなす人間がアホだとエラーも見つからない。ちゃんとプログラムが読める人が適切に使うと効率的になるツール。
- デバッグは就職後もよく使う？
  - プログラムに携わる以上は永遠にバグとの戦い（途中からはAIがやるかもしれないが）。

# 質問への回答

- デバッガ機能神では？なんで早く教えてくれなかったの？
  - 教える重要度に応じて順序が．．．
  - 確かに，デバッグ機能は神だと思う．ArrayListの次くらいに．
- デバッガがうまく使えない．．．
  - デバッガが使えない人はまずprintln()を用いたデバッグができるようになると良い（どの辺で何故エラーが出たのかをprintln()で探索する）．これを効率化するツールがデバッガだと思うと良い．
- デバッグをもう一度やるには？
  - たぶん，一度目のデバッグがちゃんと終了していないのだと思う．再生マーク付近に停止マークがあるので，ちゃんとデバッグを開始したら停止させるようにしよう．

# 質問への回答

- リファクタリングはよく使う？
  - eclipseのリファクタリングをよく使うかは謎だが、「リファクタリングしておいて」と先輩に頼まれるような、用語として使うことは多いと思う。
- 高速化は就職に向けてできておいたほうが良い？
  - 何事もできるに越したことはない。
  - 授業で触れたレベルは一般常識として知っておいてほしい。
- 現場のプログラムはコメントだらけだったりする？
  - する。ひどい。例えば初版から書き換えるたびに以下くらい書いたりする。  
// 20180626 XXの処理をYYに変更する[4行->6行]  
// 旧プログラム4行のコメントアウト  
新プログラム 6 行

# 質問への回答

- 今まで出てきたArrayList等の初期化の仕方をまとめてほしい.
  - ごめん. 答えてあげたいんだけど何がほしいのかよくわからなかった.
  - `ArrayList<E> list = new ArrayList<T>();`
- 最初から正しいところに書けばリファクタリング不要な気がする.
  - ところがどっこい, 仕様変更やスパゲッティなコードを, 後日きれいに読みやすくするという作業はよくある.
- 提出するコードに解説を書いたら良いことある?
  - 普段の演習課題のことだろうか? それならば残念ながら「ない」.
- 文字化けはeclipse内では直せない?
  - eclipseは一度開くとその文字コードに勝手に変換するような挙動を取ることがあるように見える. . . その辺微妙.

# 質問への回答

## 自由開発に関して

- 自由課題でサードパーティ製APIを使用してよいか？
  - 全然OK. 使える車輪はどんどん使おう.
- 自由課題でメインはJavaだが一部他の言語に任せてもよいか？
  - Javaの講義なので悩んだが, メインがJavaならOK. 基準は特にないから発表時にアピールして.
- 発表用の動画は自宅で撮影してきてもよいか.
  - 全然OK. 発表資料作成等々もなんでもOK.
- 自由課題の発表順は学籍番号順か？
  - 過年度生→正規年度生→編入生の順で学籍番号順 (学籍でソートするとそうなるので) .



# 本講義の概要

前半		後半	
第1回	基本文法の復習	第9回	標準ライブラリ
第2回	クラス~カプセル化の復習	第10回	ファイル入出力
第3回	抽象クラス, インタフェース	第11回	デバッグ, インポート, 高速化
第4回	ポリモーフィズム	第12回	<b>オブジェクト指向</b>
第5回	GUI 1	第13回	自由開発演習 1
第6回	GUI 2	第14回	自由開発演習 2
第7回	スレッド, 例外処理	第15回	自由開発演習発表会
第8回	ジェネリクス, コレクション	第16回	期末試験

何を開発するか, 少しずつ考えておくこと

# 本日の目標

## 概要

君たちはもうJavaの**基礎文法**を一通り学習した。しかし、Javaで重要なのは、基礎文法ができる前提で**オブジェクト指向**ができることである。改めてオブジェクト指向を学ぼう。

## 目標

作りたいシステムからクラスの設計が簡易的に実現できる。



なるほど

# 本日の提出課題

## 講義パート

課題を意識しながら  
講義を聞くと良い。

### 課題 1

本日の授業を聞いて、  
**よくわかった**と思う内容を2点簡潔に述べよ。

### 課題 2

本日の授業を聞いて、  
**質問事項**または**気になった点**を1点以上簡潔に述べよ。

### 課題 3

感想（あれば）

# オブジェクト指向

## 3大要素

継承

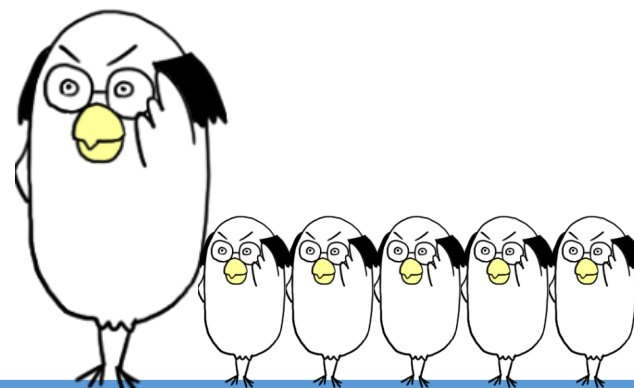
カプセル化

ポリモρφイズム

クラスとインスタンス

Javaの基礎文法

3大要素はオブジェクト指向開発を行う上でJavaに実装されている便利な機能である。



# オブジェクト指向

クラスとインスタンス

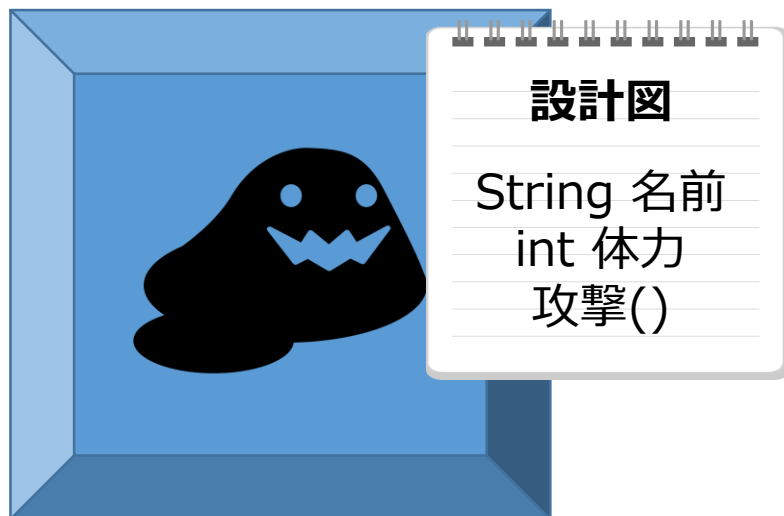
クラス

インスタンス

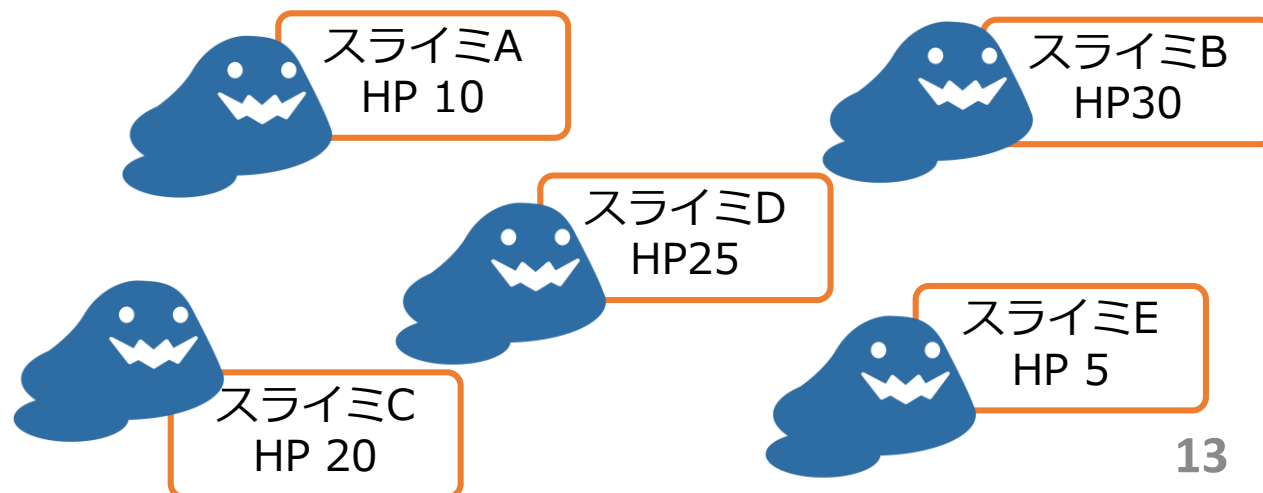
= 設計図や金型

= クラスをもとに作られた量産品

クラス（金型）



インスタンス（金型で量産）



# オブジェクト指向

## 継承

共通する部分を別のクラスにし，各クラスがそれを**継承**することで，同じ処理を一元管理することができる。



スライミ

String 名前  
int 体力  
攻撃()



デヨラン

String 名前  
int 体力  
攻撃()  
幻術()



キメラン

String 名前  
int 体力  
攻撃()  
飛行攻撃()



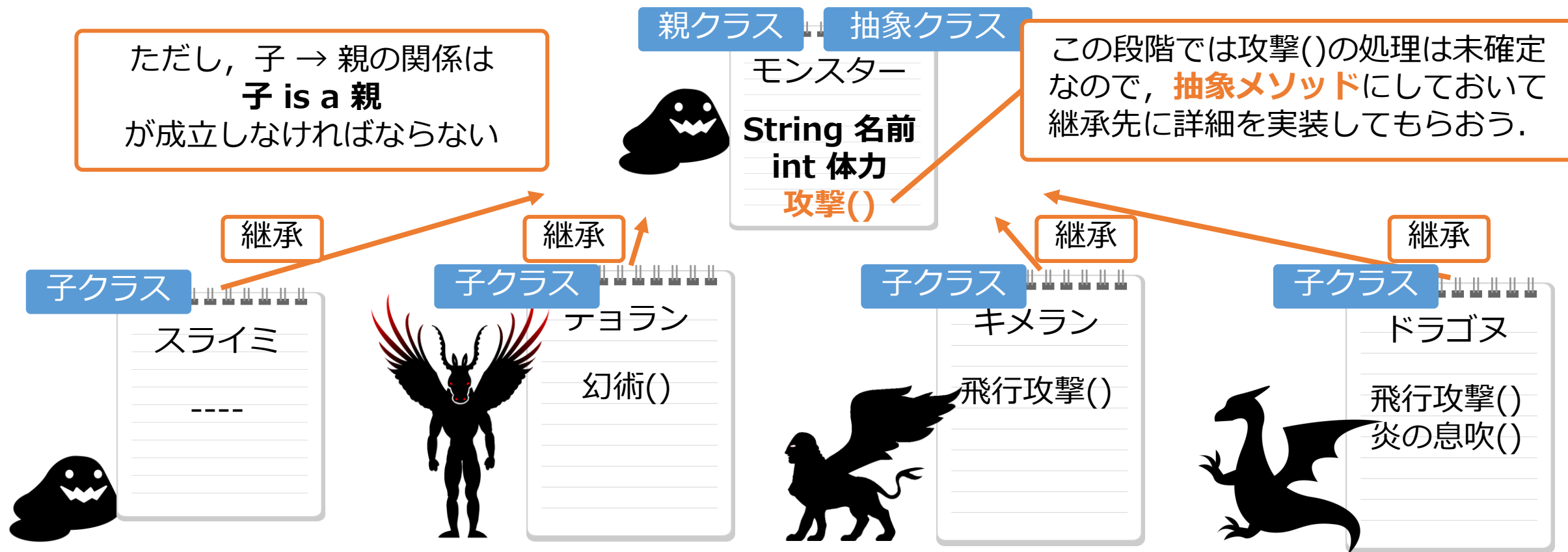
ドラゴヌ

String 名前  
int 体力  
攻撃()  
飛行攻撃()  
炎の息吹()

# オブジェクト指向

継承, 抽象クラス, 抽象メソッド

共通する部分を別のクラスにし, 各クラスがそれを**継承**することで, 同じ処理を一元管理することができる.



# オブジェクト指向

implements : インタフェース

共通する部分を別のクラスにし，各クラスがそれを**継承**することで，同じ処理を一元管理することができる。

親クラス 抽象クラス



モンスター  
String 名前  
int 体力  
攻撃()

よく見たら**飛行攻撃()**という機能も共通するものがある...

継承

継承

継承

継承

子クラス

スライミ

----



子クラス

デヨラン

幻術()



子クラス

キメラン

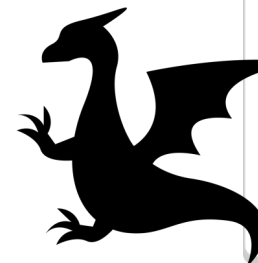
飛行攻撃()



子クラス

ドラゴヌ

飛行攻撃()  
炎の息吹()

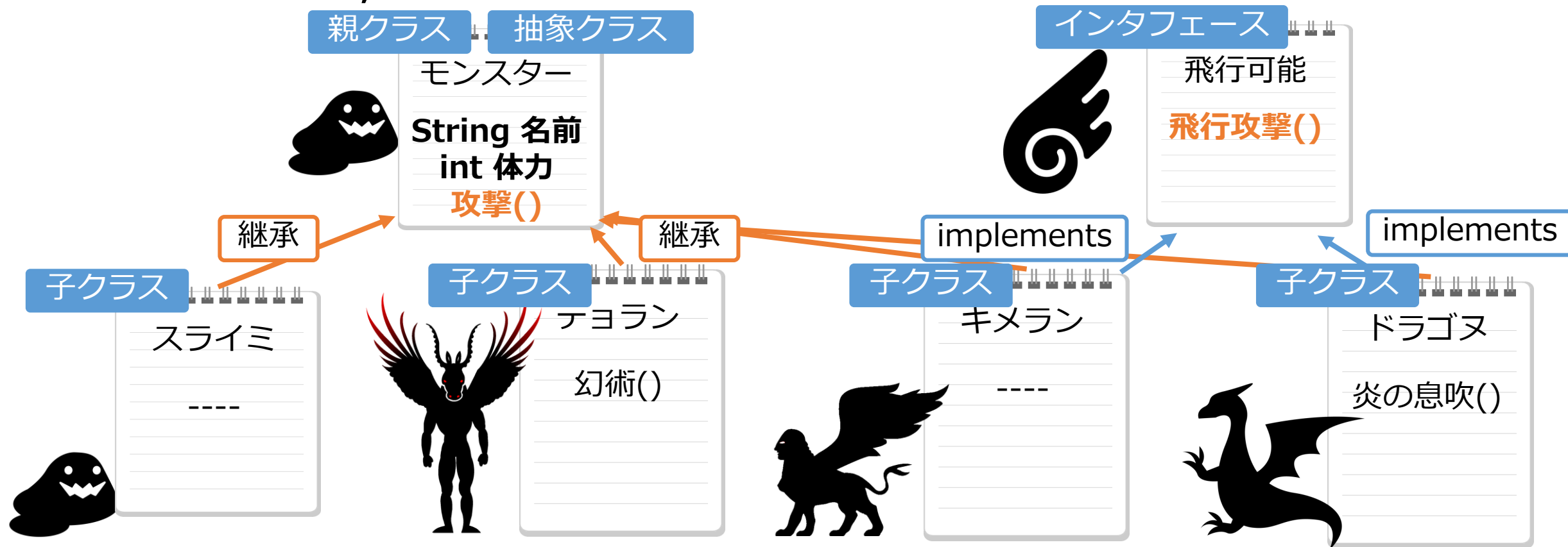




# オブジェクト指向

implements : インタフェース

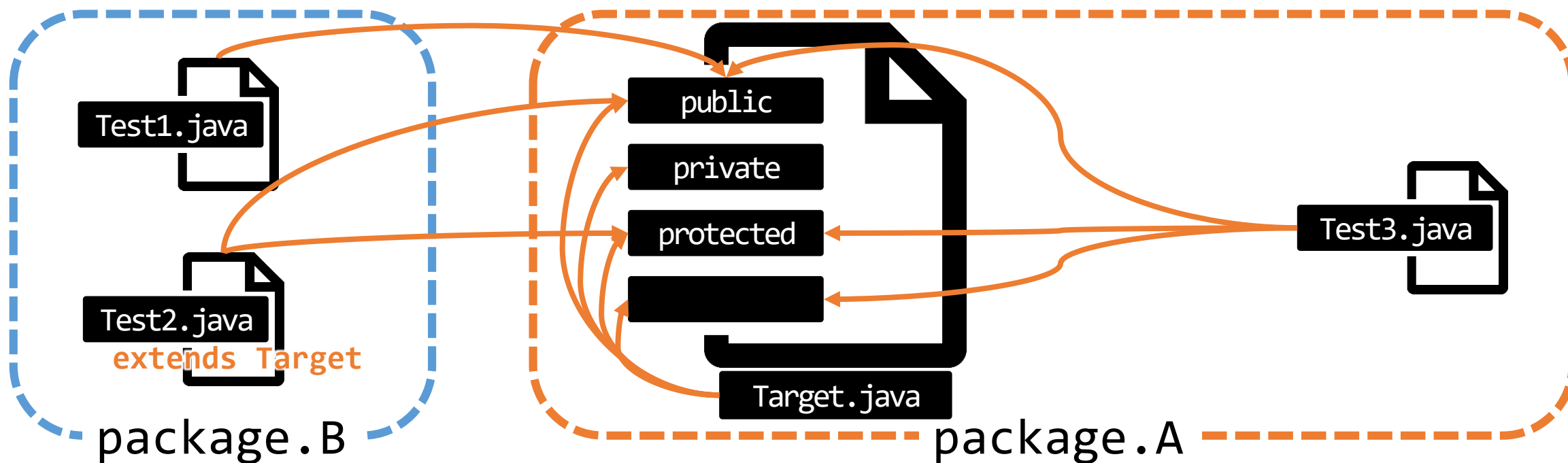
共通する部分を別のクラスにし，各クラスがそれを**継承**することで，同じ処理を一元管理することができる。



# オブジェクト指向

カプセル化：クラス、フィールド、メソッドに対する4種類のアクセス権

<b>public</b>	: すべてのクラスからアクセスできる.
<b>private</b>	: 自身のクラスからのみアクセスできる.
<b>protected</b>	: 自分を継承したクラスからor 同じパッケージに属するクラスからアクセスできる.
何も書かない	: 同じパッケージに属するクラスからアクセスできる.

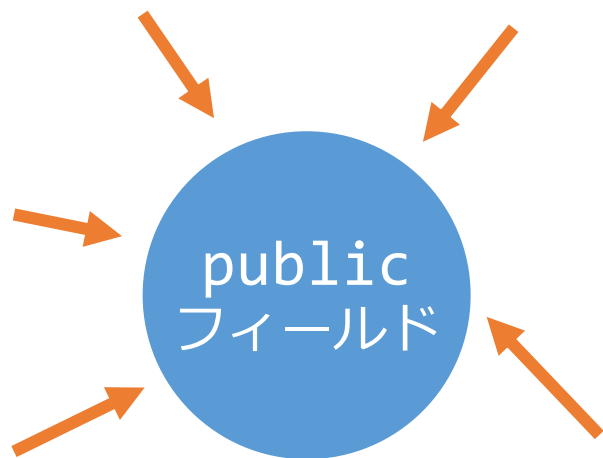


# オブジェクト指向

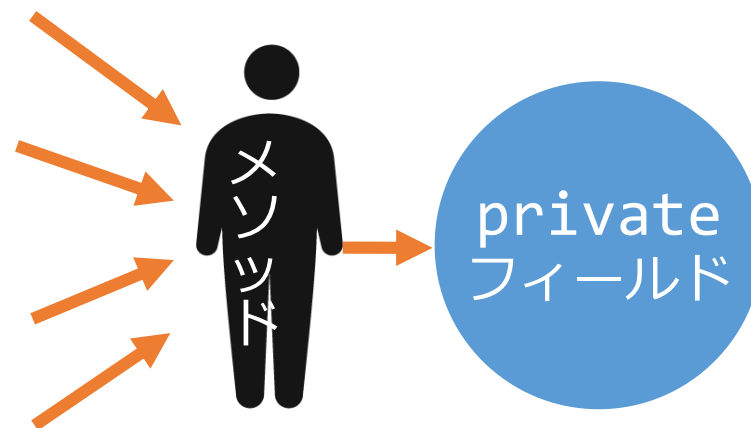
## カプセル化

**カプセル化**： フィールドやメソッドを隠蔽すること

4種類のアクセス権を正しく設定することで誤使用を減らせる。  
拡張性を考えてprivateをprotectedにすることもある。



どこからでも改変されうる  
→デバッグ時箇所の特定制が難しい



メソッドが仲介する  
→デバッグ時にメソッドで対応できる

# オブジェクト指向

## ポリモーフィズム

**ポリモーフィズム**： 型をあえてザックリと捉える方法

型をザックリ捉えることで、色々細やかな違いを気にせずに、  
だいたいこんな感じでしょ、とオブジェクトを操作できる。



とりあえず全部モンスターやろ。  
attack()しとけば攻撃するんやろ。



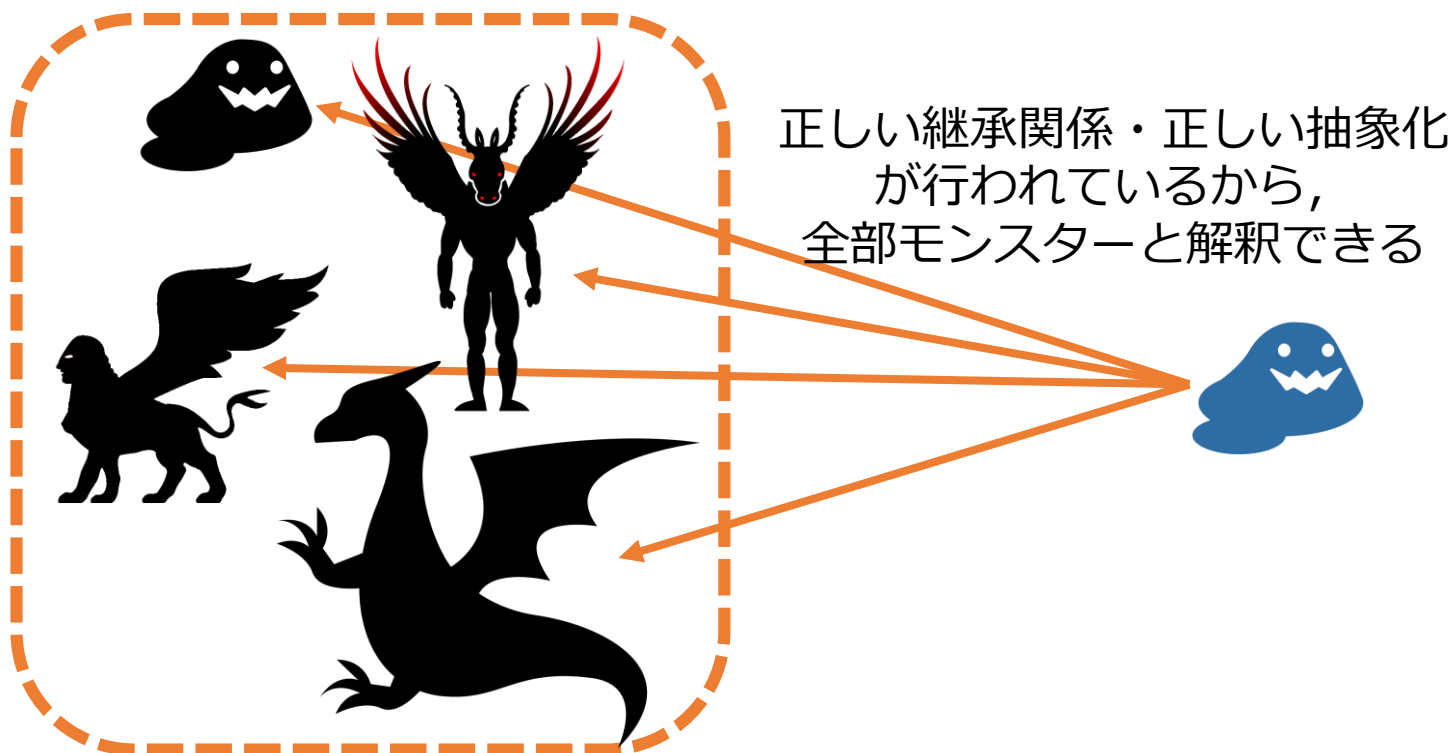
なんか色んな種類おるようやけど、  
おっちゃんにはわからんわあ～

# オブジェクト指向

## ポリモーフィズム

**ポリモーフィズム**： 型をあえてザックリと捉える方法

型をザックリ捉えることで、色々細やかな違いを気にせずに、  
だいたいこんな感じでしょ、とオブジェクトを操作できる。



ポリモーフィズムが活躍するためには、設計時に正しい継承関係を検討しなければならない。

共通部分やis-aの関係をうまく抽象化しておくことで後の開発者がポリモーフィズムの恩恵を受けることができる。

# どうでもいい話

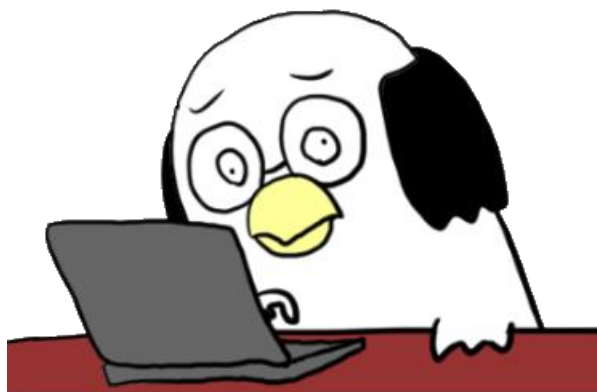
## お便りコーナー

- 医療SEとは？今までやった一番のミスは？
  - 電子カルテとか病院にあるシステム全般を開発したり導入したりする仕事。電子カルテ導入だと数億のプロジェクトとかざら。
  - データ移行途中で勝手に止めたこと（データ飛んでたら首も飛んでた）。
- 学生のうちにしておくこととかある？
  - なんでも良いから、自分の一つ上のハードルにチャレンジすること。 **IPA 未踏事業**とか**ソフトウェアコンテスト**とか、**起業**とか。学生のうちなら失敗しても死なない（おっさんになったらそんなリスク踏むのが辛くなる）。
- 学生のうちにアプリやゲーム作っておいたほうが就活有利？
  - 多少は？資格もそうだけど、何もやってない同じレベルの人と並んだ時に、それで差をつける程度にしかない。けど、作りたいものを作っているという情熱をぶつけられると採用担当は揺らぐと思う。

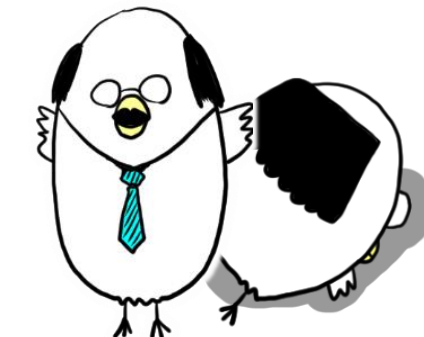
# オブジェクト指向

クラスを設計してみよう

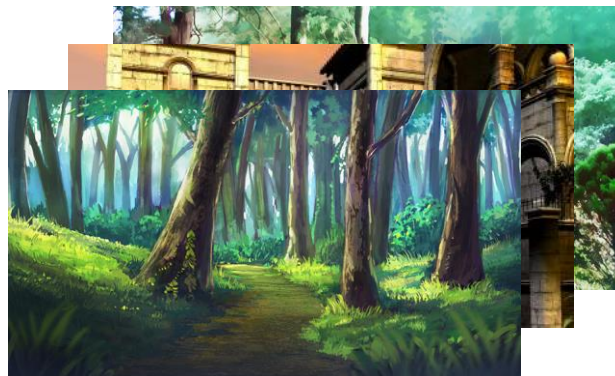
私がいろんなステージを  
ギューンと駆け巡るような  
ゲームが開発したいな



開発物をイメージ



ハトヤマ先生が



いろんなステージを  
どんなオブジェクト  
があるだろう

## ハトヤマクラス

int 体力  
飛ぶ()  
投げる()

## ステージクラス

String 名前  
int[][] マップ配置  
描画()

クラスを設計

# オブジェクト指向

クラスを設計してみよう

第四回の演習課題（Twitter風アプリ開発）を元に，クラスを設計する練習をしてみよう．

投稿の一覧をTimelineと呼ぶ



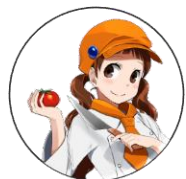
ハトヤマ先生 @Hatoyama

Twitterはこんな感じでつぶやくアプリ  
[2018/07/03 23:50:00]



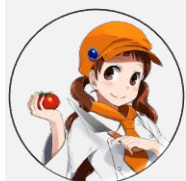
ハトヤマ先生 @Hatoyama

私はボッチなので一人でつぶやき続けます.  
[2018/07/03 23:51:00]



フリー素材 @FreeMaterial

@Hatoyama ドンマイです.  
[2018/07/03 23:54:00]



ハトヤマ先生 Retweeted ↕

フリー素材 @FreeMaterial  
@Hatoyama ドンマイです.  
[2018/07/03 23:54:00]

←つぶやき 1 件のことを**Tweet**と呼ぶ

名前 @ID

Tweetの内容  
[日時]

という書式で表示される

←返信 1 件のことを**Reply**と呼ぶ

←Tweet等の再投稿を**Retweet**と呼ぶ



# オブジェクト指向

クラスを設計してみよう


クラス設計を行うときには、このシステムの概要を一度文章に起こしてみると良い。

「Twitter風アプリでは、**ユーザ**が**Tweet**を**つぶやいたり**、**Reply**を**つぶやいたり**、**Retweet**を**つぶやいたり**することができ、つぶやきの一覧を**TimeLine**上で**確認する**ことができる。」

次に、クラスの候補として**名詞**を、メソッドの候補として**動詞**を抽出しよう。

# オブジェクト指向

クラスを設計してみよう

	<b>ユーザ</b>	<b>: Client</b>
	Tweetをつぶやく	: tweet()
	Replyをつぶやく	: reply()
	Retweetをつぶやく	: retweet()

   	<b>タイムライン</b>	<b>: TimeLine</b>
	確認する	: getTweets()

	<b>ツイート</b>	<b>: Tweet</b>
	<b>リプライ</b>	<b>: Reply</b>
	<b>リツイート</b>	<b>: Retweet</b>

TimeLineはつぶやきの一覧を保有しているので、**包含関係**(**has-a**の関係)にある。  
TimeLine has a 一覧 of Tweet

# オブジェクト指向

クラスを設計してみよう

各クラスが何を保有しているのか（フィールド）を考えていこう。



ハトヤマ先生 @Hatoyama

Twitterはこんな感じでつぶやくアプリ  
[2018/07/03 23:50:00]



ハトヤマ先生 @Hatoyama

私はボッチなので一人でつぶやき続けます。  
[2018/07/03 23:51:00]



フリー素材 @FreeMaterial

@Hatoyama ドンマイです。  
[2018/07/03 23:54:00]



ハトヤマ先生 Retweeted

フリー素材 @FreeMaterial  
@Hatoyama ドンマイです。  
[2018/07/03 23:54:00]

Tweetは以下の要素を持っていそうである。

← ユーザ名 @ユーザID

つぶやき内容  
[投稿日時]

Replyは以下の要素を持っていそうである。

← ユーザ名 @ユーザID

@返信先ユーザID つぶやき内容  
[投稿日時]

Retweetは以下の要素を持っていそうである。

← ユーザ名 @ユーザID

Retweet対象の投稿情報

# オブジェクト指向

クラスを設計してみよう

各クラスが何を保有しているのか（フィールド）を考えていこう。



ハトヤマ先生 @Hatoyama  
Twitterはこんな感じでつぶやくアプリ  
[2018/07/03 23:50:00]



ハトヤマ先生 @Hatoyama  
私はボッチなので一人でつぶやき続けます。  
[2018/07/03 23:51:00]



フリー素材 @FreeMaterial  
@Hatoyama ドンマイです。  
[2018/07/03 23:54:00]



ハトヤマ先生 Retweeted   
フリー素材 @FreeMaterial  
@Hatoyama ドンマイです。  
[2018/07/03 23:54:00]

## Tweet

String username;  
String userId;  
String text;  
Date submitted;

## Reply

String username;  
String userId;  
String targetId;  
String text;  
Date submitted;

## Retweet

String username;  
String userId;  
Tweet target;  
Date submitted;

ここ、ReplyやRetweet  
かもしれないし...

共通箇所多すぎ?  
抽象化できそう。

# オブジェクト指向

クラスを設計してみよう

各クラスが何を保有しているのか（フィールド）を考えていこう。



ハトヤマ先生 @Hatoyama  
Twitterはこんな感じでつぶやくアプリ  
[2018/07/03 23:50:00]



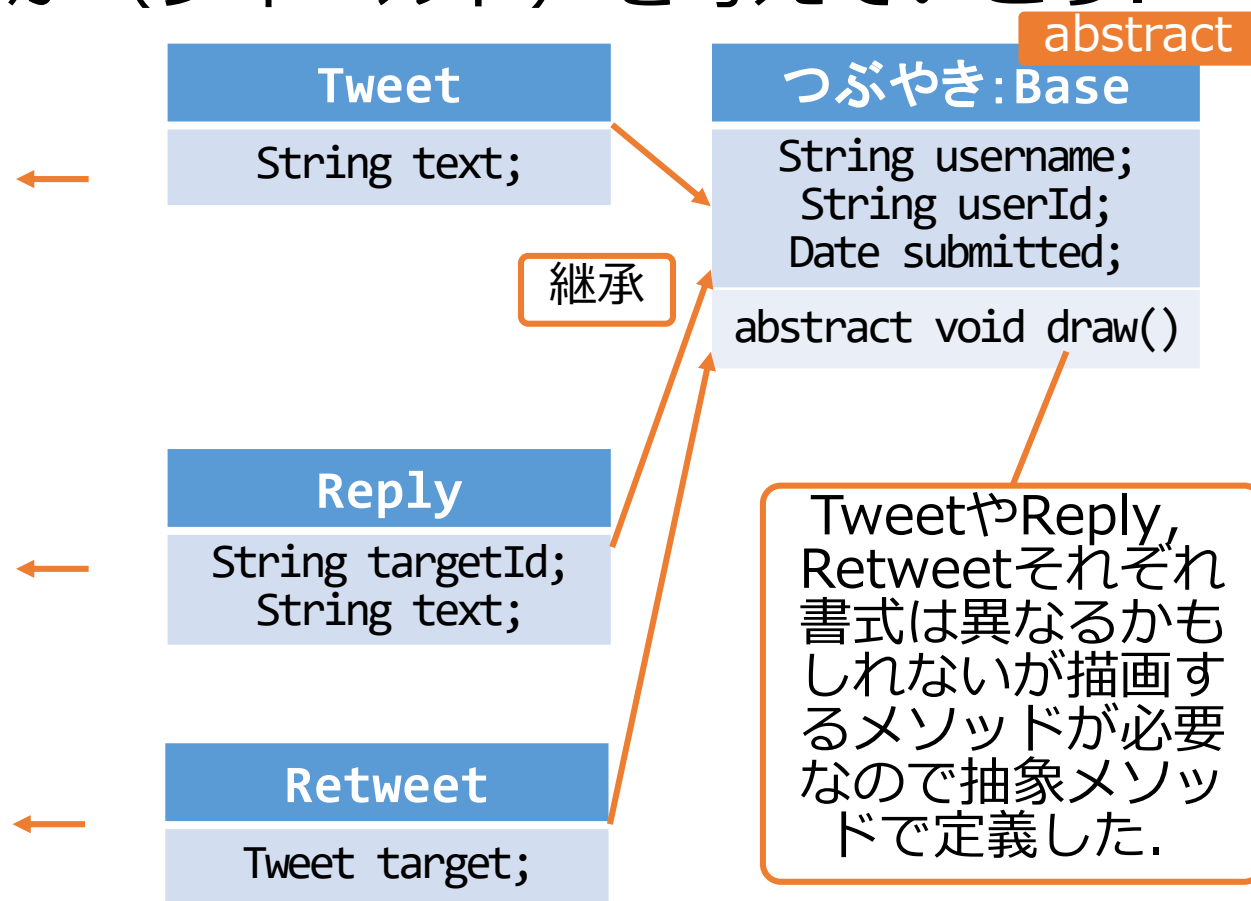
ハトヤマ先生 @Hatoyama  
私はボッチなので一人でつぶやき続けます。  
[2018/07/03 23:51:00]



フリー素材 @FreeMaterial  
@Hatoyama ドンマイです。  
[2018/07/03 23:54:00]



ハトヤマ先生 Retweeted   
フリー素材 @FreeMaterial  
@Hatoyama ドンマイです。  
[2018/07/03 23:54:00]



# オブジェクト指向

クラスを設計してみよう

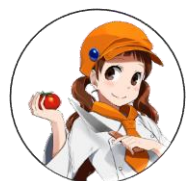
各クラスが何を保有しているのか（フィールド）を考えていこう。



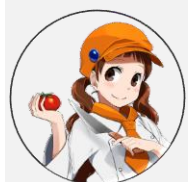
ハトヤマ先生 @Hatoyama  
Twitterはこんな感じでつぶやくアプリ  
[2018/07/03 23:50:00]



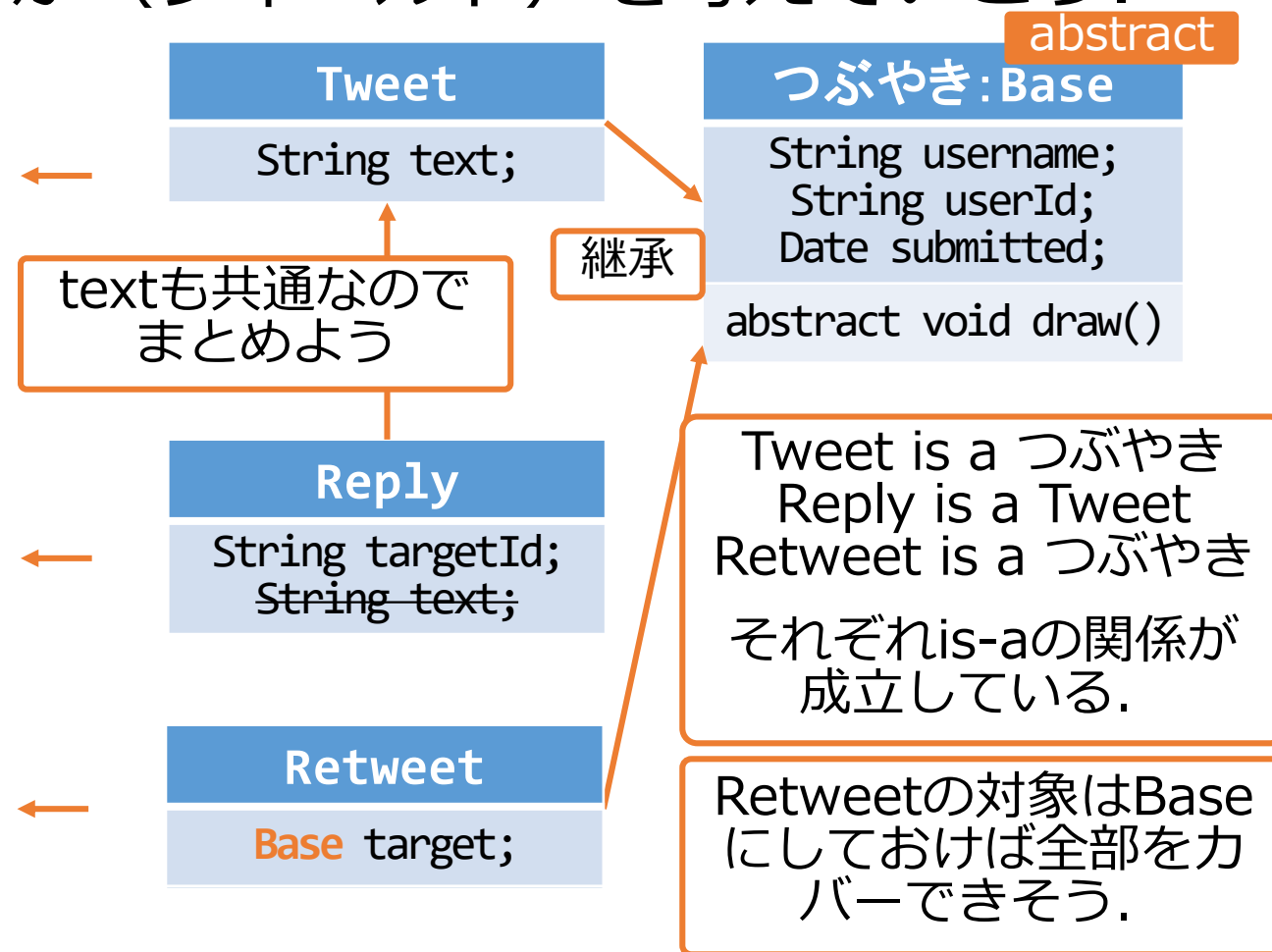
ハトヤマ先生 @Hatoyama  
私はボッチなので一人でつぶやき続けます。  
[2018/07/03 23:51:00]



フリー素材 @FreeMaterial  
@Hatoyama ドンマイです。  
[2018/07/03 23:54:00]



ハトヤマ先生 Retweeted   
フリー素材 @FreeMaterial  
@Hatoyama ドンマイです。  
[2018/07/03 23:54:00]



# オブジェクト指向

クラスを設計してみよう

各クラスが何を保有しているのか（フィールド）を考えていこう。



## Client

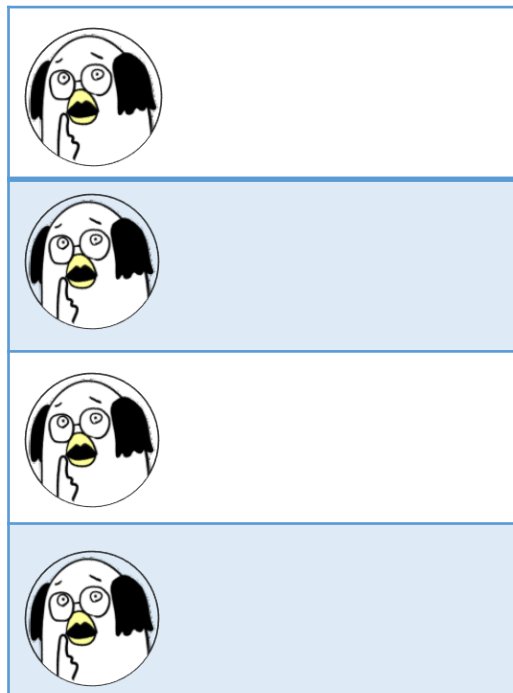
```
String username;  
String userId;
```

```
void tweet(String text);  
void reply(String text);  
void reply(Base target);
```

## TimeLine

```
ArrayList<Base> logs;
```

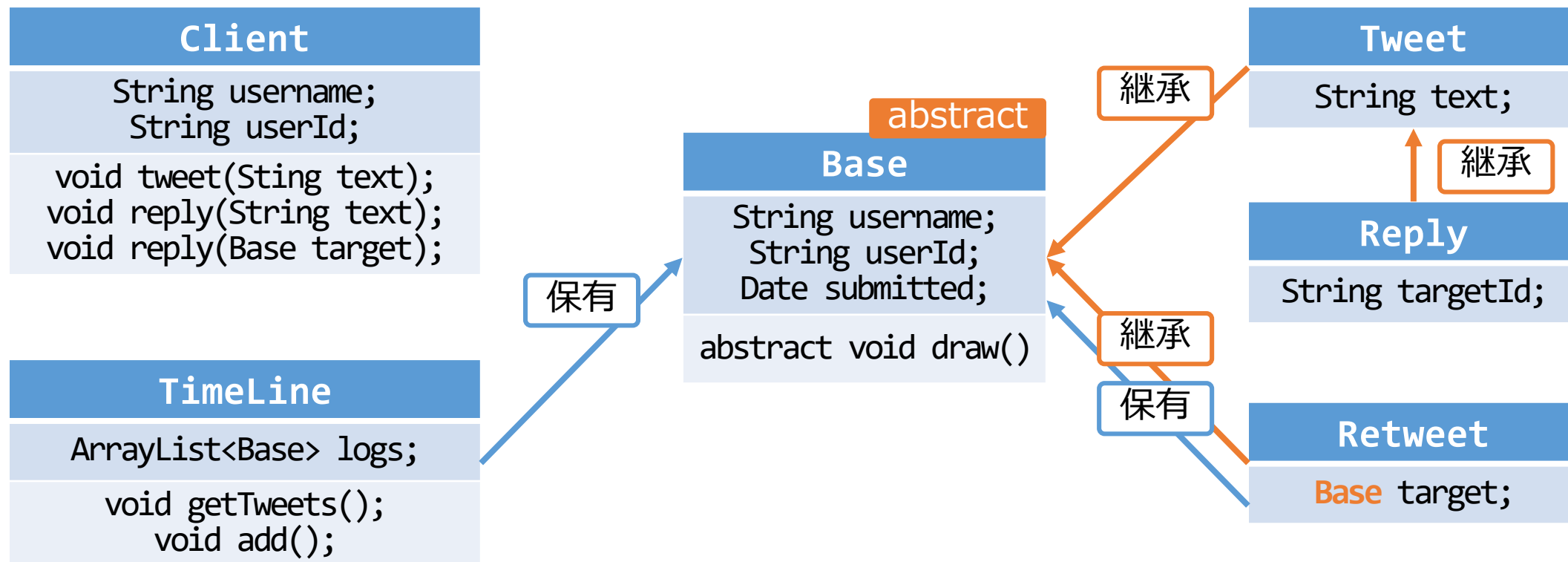
```
void getTweets();  
void add();
```



# オブジェクト指向

クラスを設計してみよう

まとめるとこんな感じ, , , ということで第四回の演習問題のクラス関係となった (細かな点は若干変更したが) .



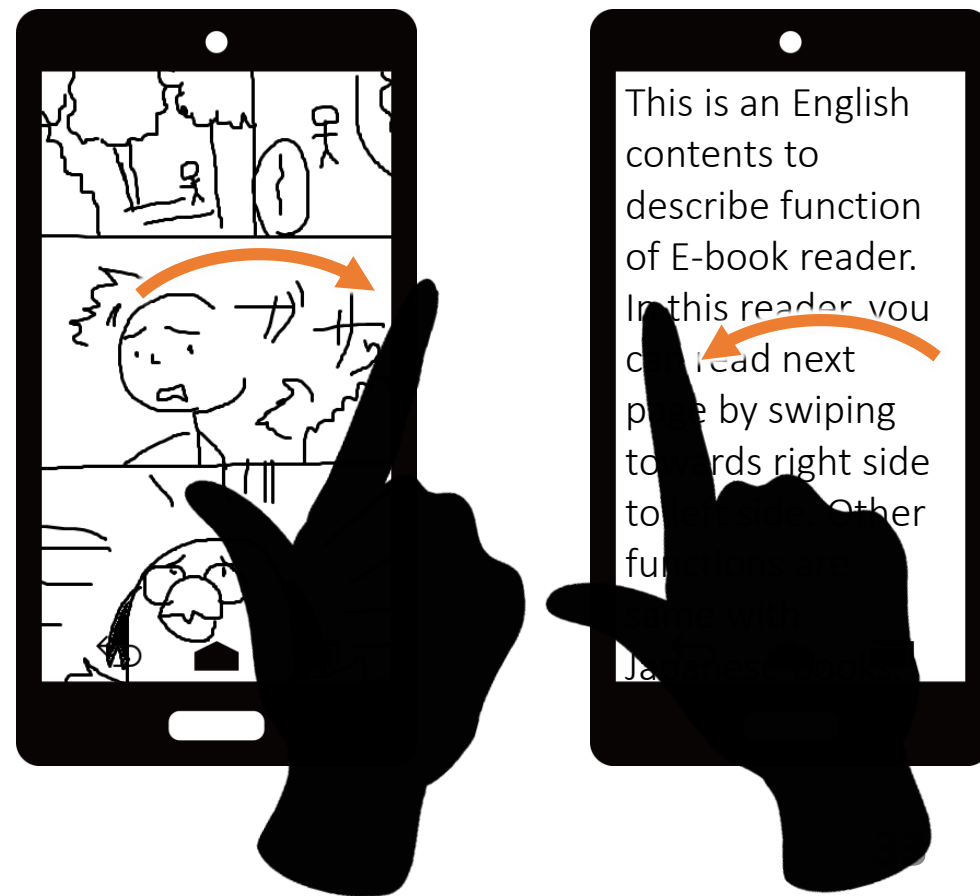


# オブジェクト指向

練習問題：クラスを設計してみよう

電子書籍リーダー関連のクラスを設計してみよう。

「電子書籍リーダーでは、各種書籍を閲覧画面で閲覧することができる。ザックリ3種類、漫画本閲覧画面は右から左に向かってページをめくって閲覧し、見開き表示することもできる。日本語本閲覧画面は右から左に向かってページをめくって閲覧する。英語の本閲覧画面は左から右に向かってページをめくって閲覧する。」

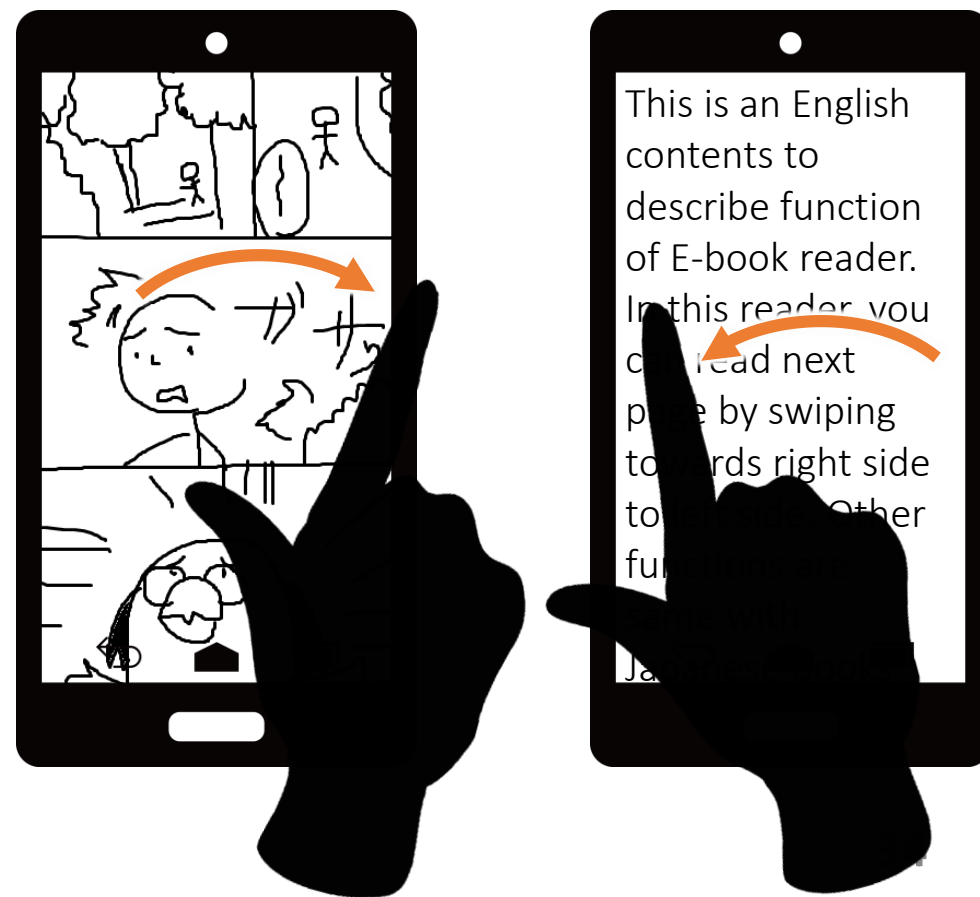


# オブジェクト指向

練習問題：クラスを設計してみよう

手順は次の通り.

1. 名詞と動詞を抽出し, クラスとメソッドを検討する.
2. 各クラスの中身 (フィールド) を検討する.
3. 抽象化できるものを検討する.
4. 関係を整理する.

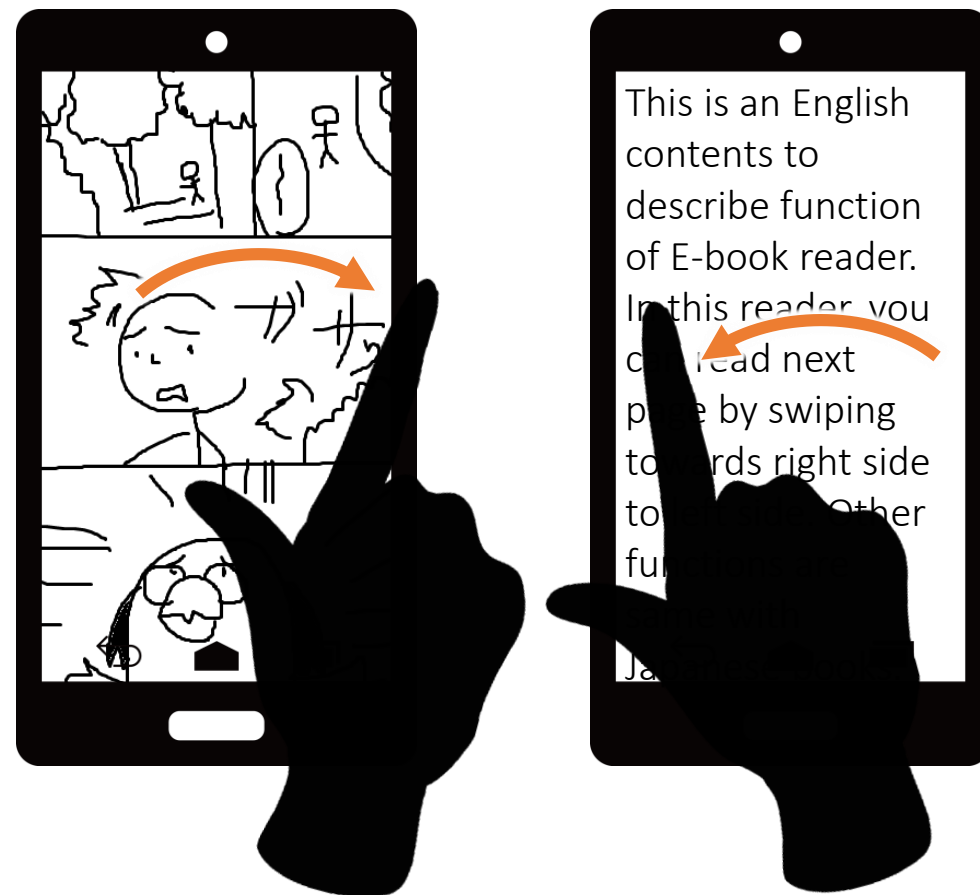


# オブジェクト指向

練習問題：クラスを設計してみよう

電子書籍リーダー関連のクラスを設計してみよう。

「電子書籍リーダーでは、各種書籍を閲覧画面で閲覧することができる。ザックリ3種類、漫画本閲覧画面は右から左に向かってページをめくって閲覧し、見開き表示することもできる。日本語本閲覧画面は右から左に向かってページをめくって閲覧する。英語の本閲覧画面は左から右に向かってページをめくって閲覧する。」



# オブジェクト指向

練習問題：クラスを設計してみよう



**閲覧画面** : ReadScreen

**閲覧する** : read()

よく考えると、「閲覧する」は人間の動作であり、リーダーが持つ機能とはちょっと違う。



**漫画本閲覧画面** : ComicReadScreen

**ページをめくる** : next(); back();

**見開きにする** : spread();

漫画本の内容データ  
を持ってそう



**日本語本閲覧画面** : JapaneseReadScreen

**ページをめくる** : next(); back();

日本語本の内容データ  
を持ってそう



**英語本閲覧画面** : EnglishReadScreen

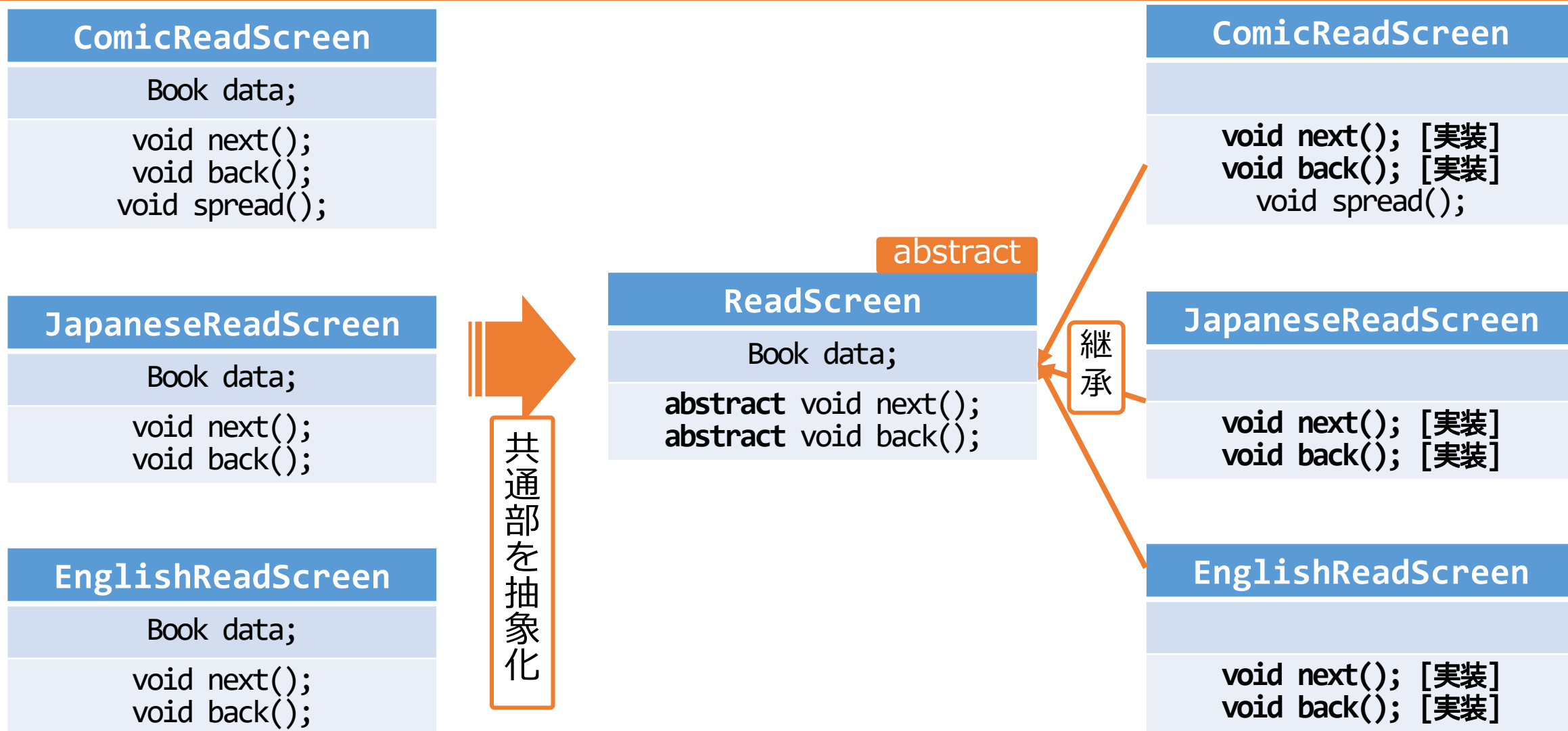
**ページをめくる** : next(); back();

英語本の内容データ  
を持ってそう

それぞれ  
データは  
画像の配  
列なので  
同クラス  
Bookで  
よさそう

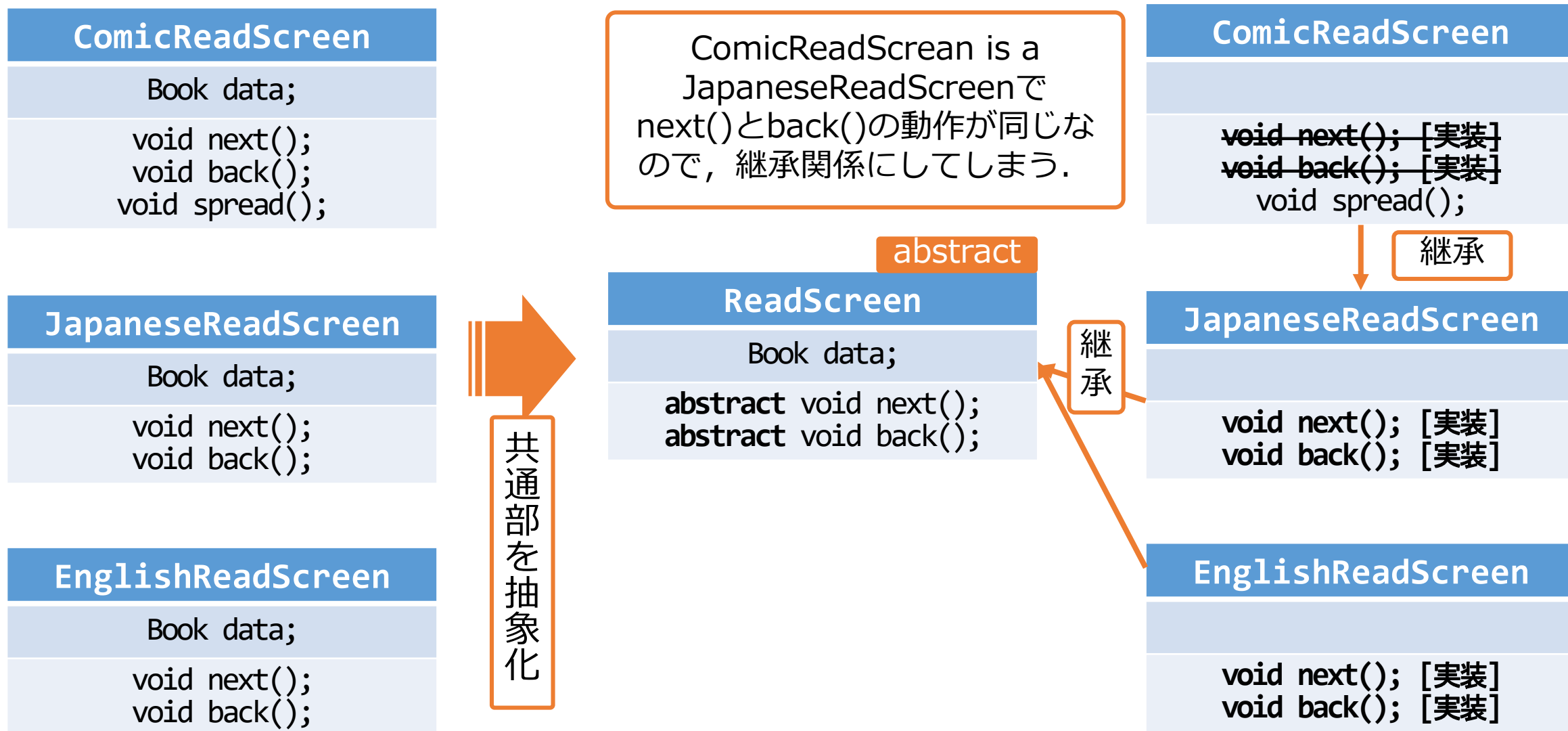
# オブジェクト指向

練習問題：クラスを設計してみよう



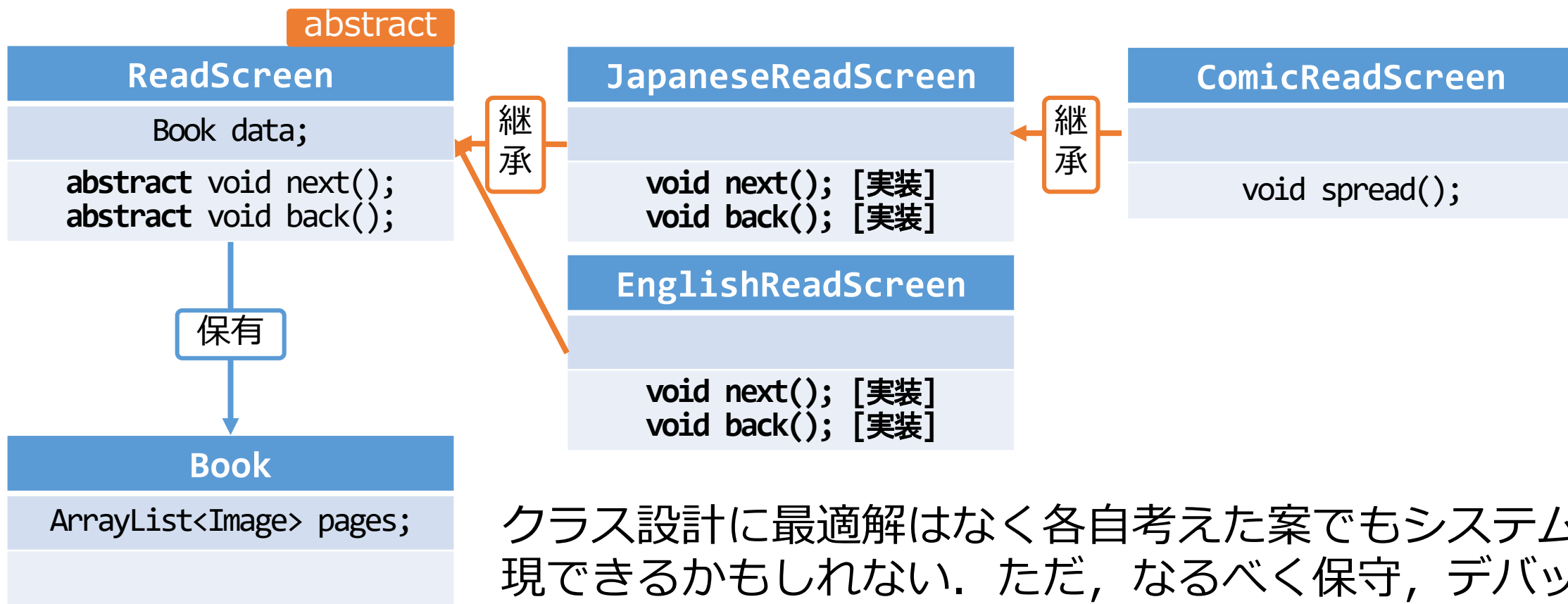
# オブジェクト指向

練習問題：クラスを設計してみよう



# オブジェクト指向

練習問題：クラスを設計してみよう



クラス設計に最適解はなく各自考えた案でもシステムは実現できるかもしれない。ただ、なるべく保守，デバッグ，拡張しやすいようにするために，オブジェクト指向を意識して設計することが望ましい。

# まとめ

- Javaの醍醐味はオブジェクト指向言語であるという点である.
- これまで学んできたJavaの文法がわかることとオブジェクト指向ができるというのは別の話である.
- 本日はオブジェクト指向の復習と, クラス設計の方法について学習した.
- クラス設計ができるようになるには, 色々なケースの経験が必要なので今後も学んでいってほしい. 本来はもう少し細かいクラスや機能の設計を行う.



# 次週予告

※次週以降も計算機室

前半

自由課題開発

後半

自由課題開発

# 本日の提出課題

## 講義パート

提出用パスワードは  
「Object」

### 課題 1

本日の授業を聞いて、  
**よくわかった**と思う内容を  
2点簡潔に述べよ。

### 課題 2

本日の授業を聞いて、  
**質問事項**または**気になった点**  
を1点以上簡潔に述べよ。

### 課題 3

感想（あれば）

### 宿題！

自由開発する内容を  
**決定してくる**こと

# 演習

- 昼休み, いつものWebページに演習問題をPDFで演習問題をアップロードする. 各自実施してプロII同様のWebページから提出すること.
- 質問は3人体制で受け付けるので遠慮なく申し出る. 質問の際は, どこまでわかっていて何がわからないのかを申し出ること.
- (ないとは思うが) コピペは発覚次第両成敗する.
  - ✓ {コピペ, カンニング} ∈不正行為
- つまらないミスも今回は問答無用で×とするので, 最終チェックを怠らないこと. (去年は目視で甘めに採点していたが, 自動採点を開発している意味がないので. . . )