JUnitで
ごあいさつ

今日やること

- 日付が関連する機能のテスト作成を通じて、モックを使ったテストを作成してみる
- TDDに関して広く浅く知識をつける

お品書き

- TDDの目的（さわりだけ）
- TDDに関するツールの使い方（ちょっとだけ）
- 日付を挨拶メソッドの実装（設計除く）
- まとめ

TDDの目的

TDDって

なんのためにあるの？

「動作するきれいなコード」がTDDの目標である。

- 有機的に設計しなければならない。コード内容の決定後に実行結果からフィードバックが得られる。このフィードバックを使ってコード内容を変更する。

Kent Beck
テスト駆動開発入門より

ツールの使い方

QuickJunit

- テストの作成および
  テストクラスとプロダクションコードの移動
  Ctrl+9

- テストの実行
  Ctrl+0

Eclipseでの
リファクタリング

- リネーム（変数、メソッド、クラス）
  Alt+Shift+r(ename)

- メソッドの抽出
  Alt+Shift+m(ethod)

- ローカル変数の抽出
  Alt+Shift+l(ocal)

- 変数、メッソドのインライン化
  Alt+Shift+i(nline)

- その他リファクタリングメニューの表示
  Alt+Shift+T

自動でやればバグらないって
ものじゃないんでご注意を

挨拶メソッドの実装

前回のテスト作成で
やったこと

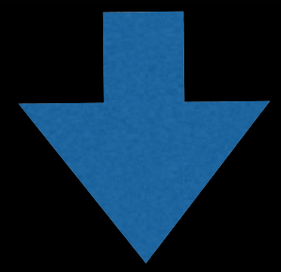
- 関数(FizzBuzz)のテスト作成
- JUnitApiの使用方法
- TDDの流れ
  レッド→グリーン→リファクタリング

世の中には自動でテストしにくいものがある

- 乱数
- 日付
- 外部依存

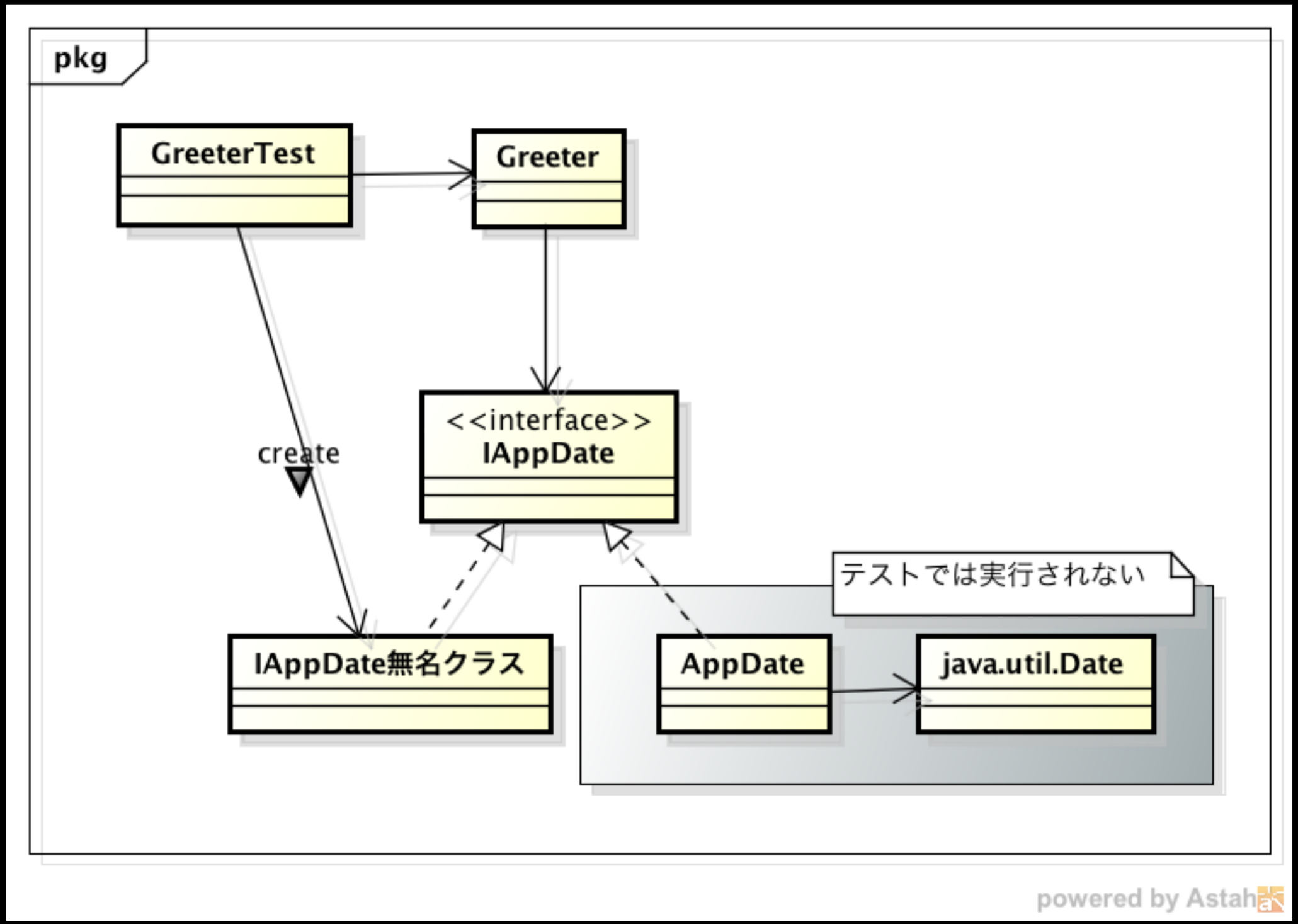
etc…

自分で制御出来ないもの



自分で制御出来るようにして（モック化）
自動テストする

スケルトンの構成


pkg
GreeterTest
Greeter
<<interface>>
IAppDate
create
IAppDate無名クラス
テストでは実行されない
AppDate
java.util.Date
powered by Astah

仕様

- あいさつするメソッド
  - 5:00以上12:00未満→”おはよう”
  - 12:00以上18:00未満→”こんにちは”
  - 18:00以上5:00未満→”こんばんは”

自動テストの注意点

- 再現性がある
  →「いつ」「何度実行しても」同じ結果になる
- 独立している
  →他のテストの影響を受けない

まとめ

今日やったこと

- Eclipse,QuickJunitのショートカット
  →TDDの高速化に役立ちます
- モックを使用したテストの作成
  →自動テスト出来る幅が大きく広がります
  →レガシーコードの自動テスト作成にも
  　応用出来ます。