第4章
テストの構築

前回

- 不吉な臭いがするコードの紹介
  - 重複コード
  - 特性の横恋慕
  - スイッチ文　などなど

4章では

- リファクタリングをする上で欠かせないテストのお話

- リファクタリングに限らず，よいテストを設計することでプログラミングも加速するらしい

自己テストコードの意義

- プログラマが費やす時間の大半
  - コードを書く時間よりも，何が起こっているのか理解や設計に費やす時間の方が長い
  - **デバッグにかける時間が最も長いはず！**
- テストを自動化し，その結果もテストにチェックさせることでデバッグの時間を短縮する

自己テストコードの恩恵

- テストがおっくうでなくなる
  - コンパイル並にテストが簡単になるのでテストを実行する回数が増える
  - バグを埋め込んだ箇所の特定がスムーズに
    - テストが何度も繰り返されているとバグを埋め込んだときの記憶が鮮明なはず

自己テストはバグの検出に絶大な威力を発揮する
また，バグの発見にかかる時間を短縮できる

JUnitについて

- Javaのテストフレームワーク
  - 単体テストの自動化をおこなうためのフレームワーク
- Junitについては自分で調べてください

テストについて

- ほとんどのバグはテストで捕捉される
- バグがでたらまずテストを！
  - 原因を明らかにするために

- 実行されない完全なテストより実行される不完全なテストを！