

JUnit入門

技法の実践方法

参考文献：現場で使えるソフトウェアテスト -Java編-

(下書き)

発表者 吉富 楓雅

目次

- お題
- プロダクトコードの作成
- テストケース作成の進め方
- テストケースを作成するための技法

お題



“荷物の送料を計算するコード”

次の条件1～5に基づいてコードを作成せよ

それでは、ここからは具体的な題材を利用して、プロセスと技法についての理解を深めていきましょう。まずは、次の例題に目を通してください。

例題1

以下の条件1～5に基づいて、「荷物の送料を計算する」ソースコードを作りなさい。

条件1 作成するクラスは、表Aに従ったクラス定義とする。

▼表A クラス定義

パッケージ名	jp.co.shoeisha.example		
クラス	Shipping		
クラスの修飾子	public		
メソッド	int calculate (int weight, int x, int y, int z)		
	引数	weight	荷物の重さ (kg)
		x	荷物の縦幅 (cm)
		y	荷物の横幅 (cm)
		z	荷物の高さ (cm)
	戻り値	送料 (円)	
メソッドの修飾子	public		

条件2 calculate()メソッドは、次の3つの仕様に基づいて作成すること。privateメソッドは、自由に定義しても良い。

- ①引数weightには、荷物の重さを指定する。ただし重量は0kg超30kg以下の範囲で指定する。範囲外の数値が設定された場合は、IllegalArgumentExpectionをスローする
- ②引数x、y、zには荷物の各辺の長さを指定する。ただし各辺は0cm超であり、かつ3辺の合計が180cm以内の範囲で指定する。範囲外の数値が設定された場合は、IllegalArgumentExpectionをスローする
- ③指定されたデータから、表Bの送料一覧に基づいて送料を算出し戻り値として返却する。なお、表Bのサイズは、表Cのサイズ区分から決定される。たとえば、3辺合計が50cmで重量が15kgの場合は、サイズはMとなる

▼表B 送料一覧

サイズ	価格
S	600円
M	1200円
L	1800円

▼表C サイズ区分

サイズ	S辺合計	重量
S	60cm以下	10kg以下
M	120cm以下	20kg以下
L	180cm以下	30kg以下

条件3 Javadocを記述し、Sunのコーディング規約に沿っていること。ただしJavadocコメントの最初の一文の終わりには句点「。」を利用する。また、Sunのコーディング規約4章「インデント」の冒頭に
あるインデントに関してのみ、4個の半角スペースに相当するタブを利用する形に読み替える（Eclipseの
デフォルトフォーマットルールと同等）。

条件4 静的解析ツール（CheckstyleおよびFindBugs）のチェックもパスすること。なお、Checkstyle
で利用するチェックルールは、「Sun Checks (Eclipse)」から表Dで示した5つのチェックルールを変更
したものとする（Checkstyleのルール内容に関しては、巻末付録B「Checkstyle／FindBugs」ルール
一覧）を参照）。

▼表D 変更するチェックルール

グループ名	モジュール名	変更内容
Javadoc Comments	Package Html	チェック対象外とする
Javadoc Comments	Style Javadoc	checkFirstSentenceをfalseに変更
Coding Problem	Magic Number	チェック対象外とする
Class Design	Design for Extension	チェック対象外とする
Miscellaneous	Final Parameters	チェック対象外とする

条件5 次のレビュー観点を参考にソースコードをレビューすること。

- サイズ区分を決める3辺合計や重量の条件で指定された以上／下、以下／未満といった細かな表現が正しくプログラミングされているか？
- 異常値が入力されたときの例外処理がプログラミングされているか？
- クラス名やメソッド名が正しいか？
- ソースコードに妥当なコメントが記述されているか？
- Javadocコメントの最初の一文の終わりには句点「。」が利用されているか？
- CheckstyleとFindBugsでのエラーがすべて解消されているか？

プロダクトコード の作成



プロジェクトの作成

- プロジェクト名やパッケージ名も忠実に従ってください
 - * 自分勝手にプロジェクトを作成しない
 - ▶ 統一性が無くなればトラブルが発生する原因となる

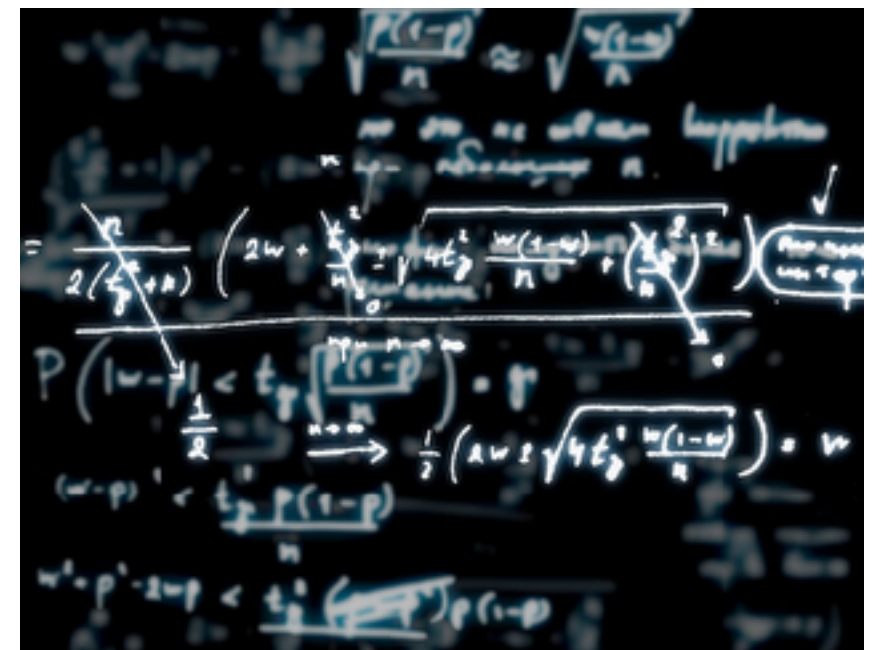
コードの生成

- Javaの書き方に従ってコードを記述してください
 - * インデントなどにも注意して, わかりやすいコードを記述することに努める
- privateとpublicも使い分けてください
 - * 計算式はprivateでも構わないと思いますが, 数値など他のプロジェクトで使われるとまずいものはprivateにしてください

コードについて

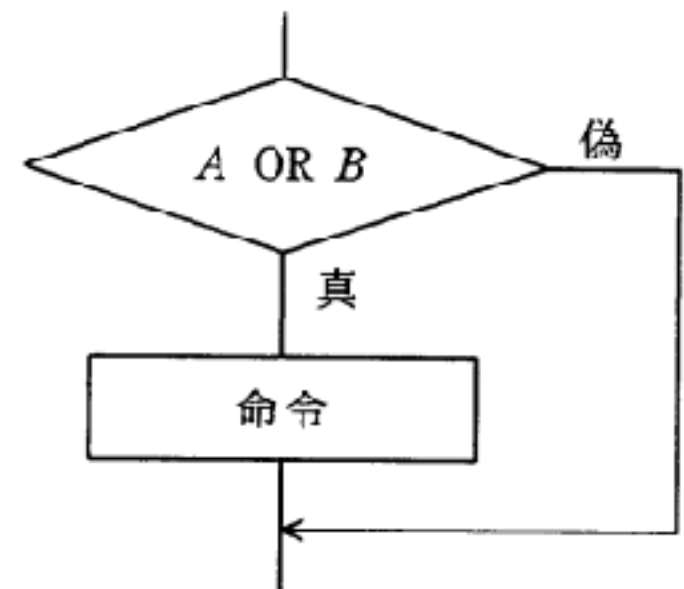
- 模範解答に当たるコードは以下のURLからダウンロードすることも可能です. 答え合わせに使ってください
 - * <http://www.seshop.com/book/download/>
 - ▶ ISBNを聞かれますので, 「978-4-7981-1463-7」と入力してください
- 以降のテストコードにつきましても, 圧縮ファイルに含まれていますので, ご覧になってください

しばらく理論を お話しします



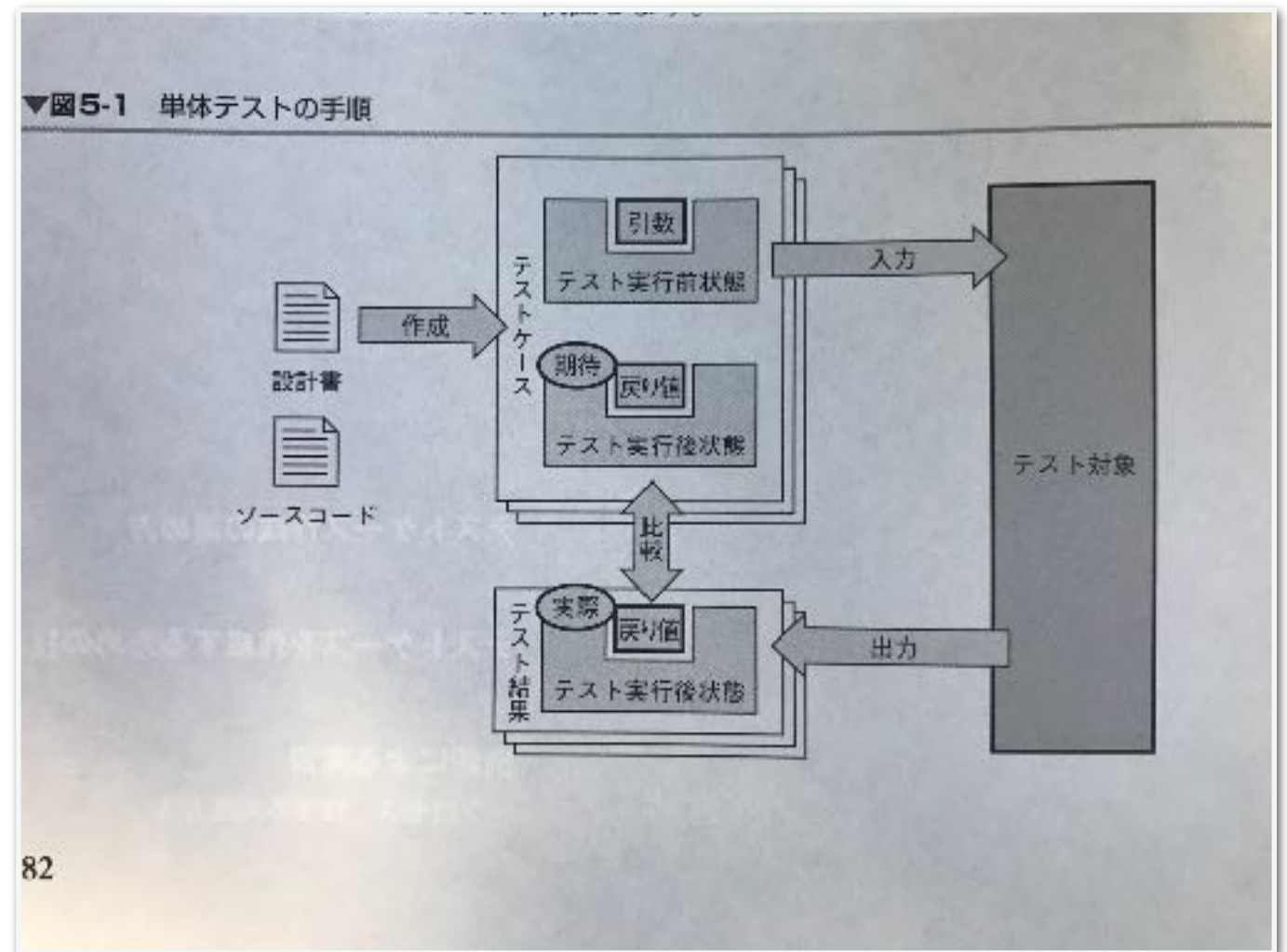
The image shows handwritten mathematical derivations on a dark background. At the top, it states $\sqrt{\frac{P(1-P)}{n}} \approx \sqrt{\frac{w(1-w)}{n}}$. Below this, there is a complex fraction involving $\frac{n}{2(\frac{1}{2} + h)}$ and a large expression in parentheses. The main part of the derivation shows the probability $P\left(\left|w - \frac{1}{2}\right| < t_p \sqrt{\frac{P(1-P)}{n}}\right) = P$ being approximated as $\frac{1}{2} \left(1 + w t_p \sqrt{4 t_p^2 \frac{w(1-w)}{n}}\right) = w$. The final line shows $w - p < \frac{t_p^2}{n} P(1-P)$.

テストケース作成 の進め方



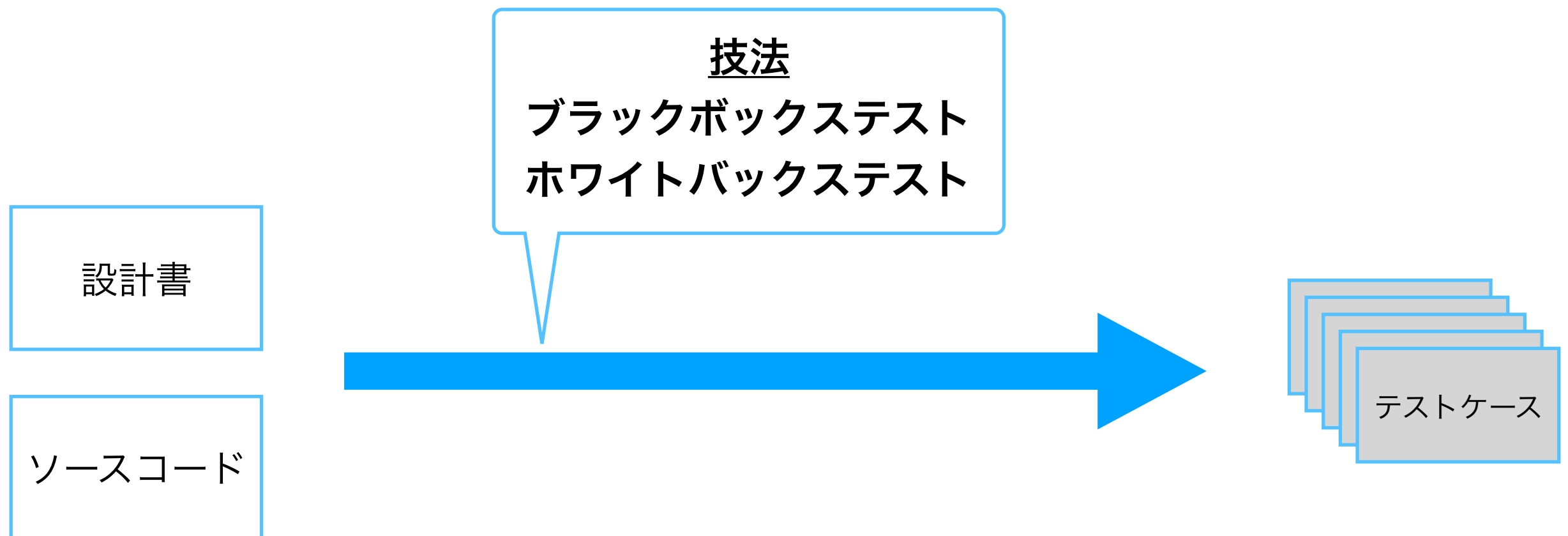
単体テスト

- まずテスト実行前状態を作り, その上で入力をテスト対象に与えて実行させる
- その結果, テスト対象はテスト実行後状態を作り, 出力を返す
- 最後に, これらの出力とテスト実行後状態があらかじめ設計書から計算した期待する出力と期待するテスト実行後状態と一致しているかを比較・検証する

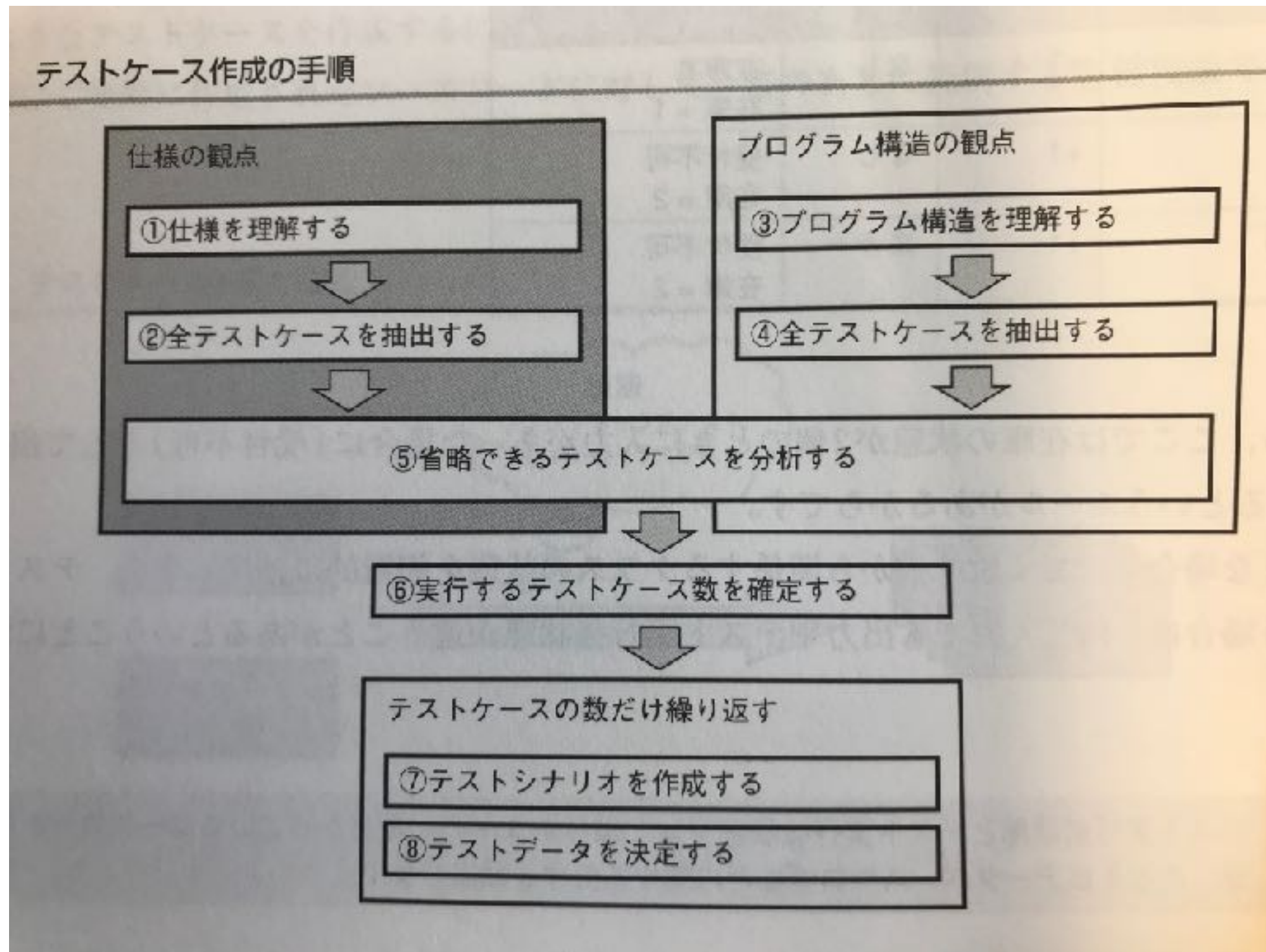


テストケース

- ・ テストケース作成と技法



テストケースの作成手順



テストケースを作成 するための技法



ブラックボックステスト

- ある入力に対してどのような出力があったかという点にのみ注目してテストケースを作成する

- * テストケース網羅手法

- ▶ 同値分割法
- ▶ 境界値分析法

同値分割法

- 仕様からデータを意味のあるグループに分類し、各グループから代表値を選ぶ
 - * 同じ意味を持つデータばかりに偏ったテストケースになることが避けられる
 - ▶ その結果効率よくテストが実施できる
- 例：int型を入力として成人であるかをチェックするようなメソッド
 - * 引数の値が0以上20未満を「未成年」20以上を「成年」と判断する
 - * 代表値を効率よく選ぶにはどうすれば良いか？

代表値の選び方

- ・ 「代表値を6つ選べ」という状況である場合

- ・ 良い例

- * 年齢として無効： -10, -5

- * 未成年： 5, 10

- * 成年： 25, 30

- ・ 悪い例

- * 年齢として無効： なし

無効になるデータが全くないのはまずい

- * 未成年： 1, 2, 3, 4, 5

- * 成年： 25

境界値分析法

- 同値分割を行なった後に, 同値クラス間の境界値をテストデータとして選ぶ手法

* 境界値はソースコードの細かいミスを発見しやすい

正しい条件文

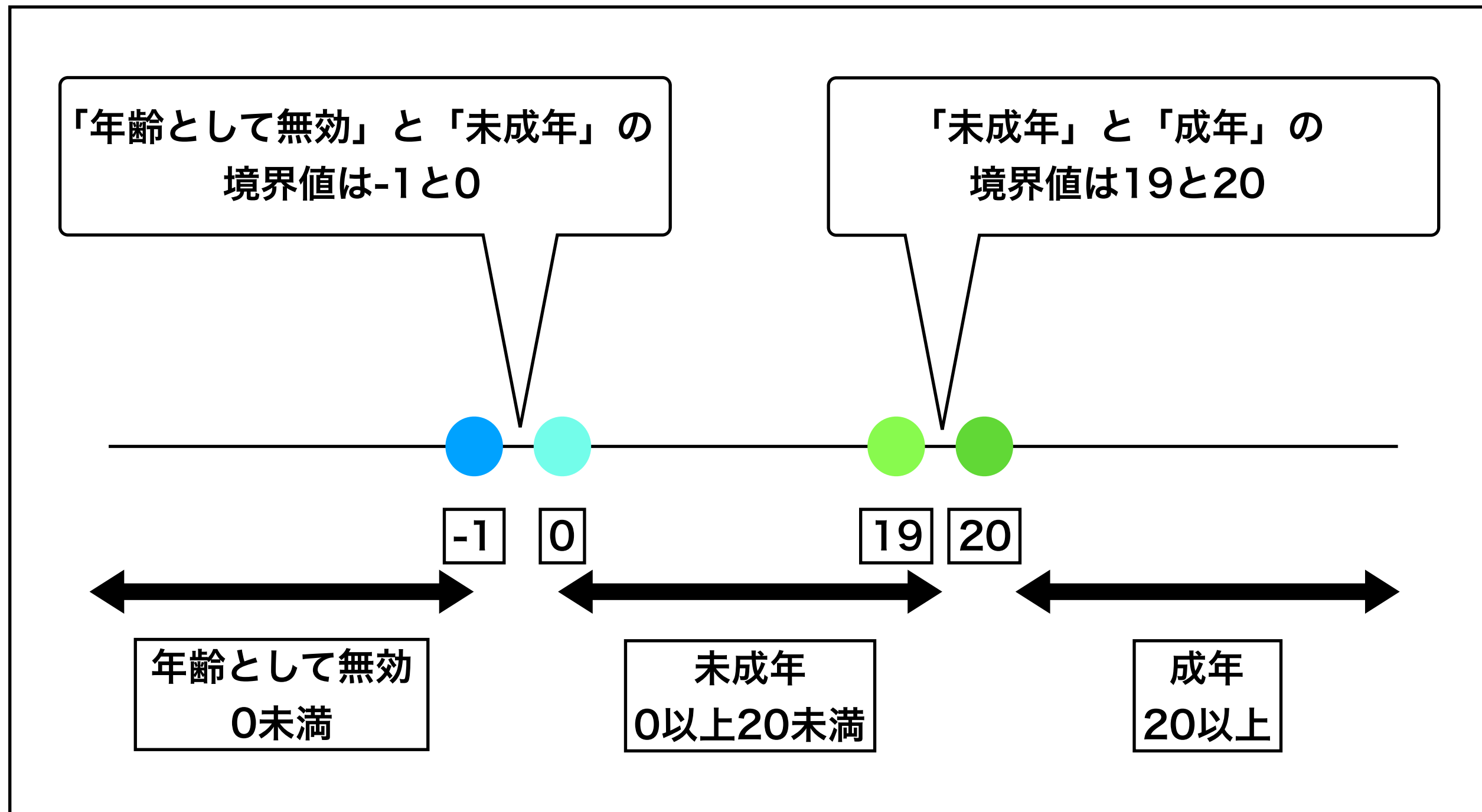
```
if (x >= 20) {  
    ///  
}
```

誤った条件文

```
if (x > 20) {  
    ///  
}
```

xに25や50など, 20から離れた値を入れてもこのバグは検出できない

境界値の選び方



ホワイトボックステスト

- ブラックボックスだけでは確認できない「仕様として決めていないのに勝手にプログラミングしてしまっている処理ロジック」が確認できる
 - * 制御構造に注目して、その処理ロジックを確認する
- 単体テストでは、特にソースコードに対するカバレッジとして、「コードカバレッジ」を利用する
- もっともよく使われるのは「制御パステスト」

コードカバレッジ

命令網羅C0	テストにより実行された命令文の割合。ソースコード内のすべての命令を1回以上実行すると命令網羅100%となる。
分岐網羅C1	テストにより実行された分岐の割合。ソースコード内のすべての分岐に対して、真と偽となるパターンを1回以上実行すると分岐網羅100%となる。
条件網羅C2	テストにより実行された条件の割合。ソースコード内のすべての条件に対して、真と偽となるパターンを1回以上実行すると条件網羅100%となる。
複合条件網羅C2	テストにより実行された条件の組み合わせの割合。ソースコード内の各分岐におけるすべての条件に対して、真と偽のすべての組み合わせパターンを1回以上実行すると複合条件網羅100%となる

ではお題のテストケース
を考えましょう！



お題のテストケース

- Shippingクラスの単体テストで実施するテストケースの作成
 - * ただしコードカバレッジとして、命令網羅および分岐網羅は100%を満たしたい

お題について

- 引数weightには、荷物の重さを指定する。ただし重量は0kg超30kg以下の範囲で指定する。範囲外の数値が設定された場合は、illegalArgumentExceptionをスローする
- 引数x, y, zには荷物の各辺の長さを指定する。ただし各辺は0cm超であり、かつ3辺の合計が180cm以内の範囲で指定する。範囲外の数値が設定された場合は、illegalArgumentExceptionをスローする
- 指定されたデータから、次ページの表の送料一覧に基づいて送料を算出し戻り値として返却する。なお、表の「サイズ」は「サイズ区分」から決定される。例えば、3辺合計が50cmで重量が15kgの場合は、サイズはMとなる。

送料とサイズ区分

送料一覧

サイズ	価格
S	600円
M	1200円
L	1800円

サイズ区分

サイズ	3辺合計	重量
S	60cm以下	10kg以下
M	120cm以下	20kg以下
L	180cm以下	30kg以下

テンプレ

- テンプレ
 - * テンプレ
 - ▶ テンプレ
- テンプレ
 - * テンプレ
- テンプレ
 - * テンプレ

テンプレ

- テンプレ
 - * テンプレ
 - ▶ テンプレ
- テンプレ
 - * テンプレ
- テンプレ
 - * テンプレ

テンプレ

- テンプレ
 - * テンプレ
 - ▶ テンプレ
- テンプレ
 - * テンプレ
- テンプレ
 - * テンプレ