

JUnit入門

効率的な単体テストの実行

参考文献：現場で使えるソフトウェアテスト -Java編-

発表者 吉富 楓雅

目次

- 単体テストの実行手順
- 機能テストの実践

単体テストの実行手順



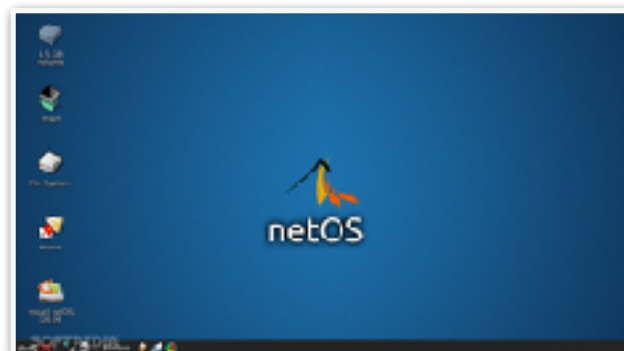
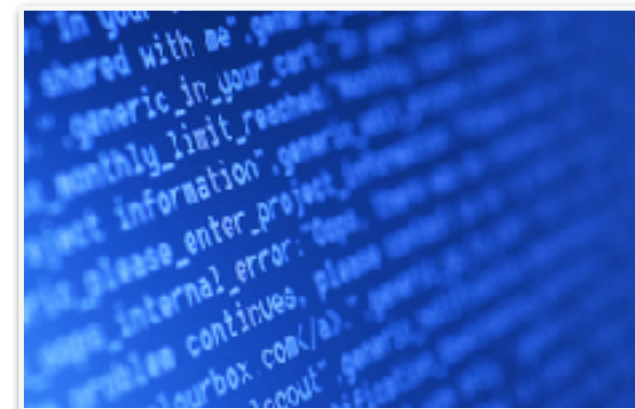
単体テスト環境の統一

- 正確なテスト結果を得るには、ソースコードの挙動に影響を与える外部環境について、何をどのように利用するのか事前に決めておく必要がある
 - * OS
 - * ミドルウェア
 - * 外部リソース(ファイルなど)
 - * クラス変数
 - * データベース(テーブル内のデータの値)

環境による処理結果の相違



同じプログラム

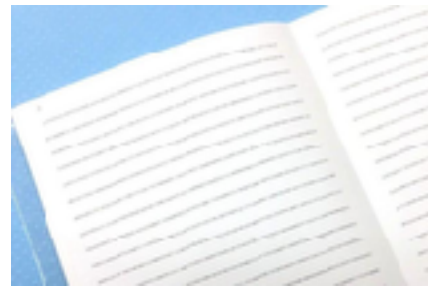


同じプログラムでも環境によっては
処理結果が異なる場合がある

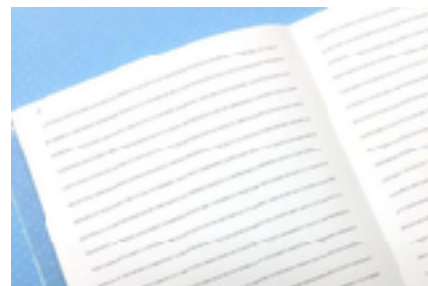
ソースコード・テストコードのバージョン管理

- 外部環境だけでなくテスト対象のソースコードおよびテストコードもバージョン管理が必要
 - * テストコードの実行中に故障が発見されるとその場で直して次の項目から始めるプログラマ
 - ▶ 単体テスト時のソースコード品質が把握できなくなるだけでなく、**修正したソースコードがすでに実施したテストケースをクリアできる保証がなくなる**
 - * テストコードに不具合があってテストが予想外の結果になった場合でも、**後からそのテストコードを変更すると原因がわからなくなる**
- **単体テストが始まったら、終了するまでソースコードに手を加えてはならない**

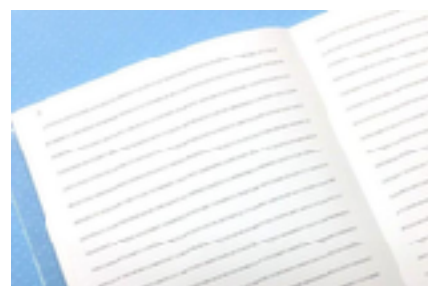
単体テスト実行中のソースコード修正はNG！



単体テスト中に故障発見



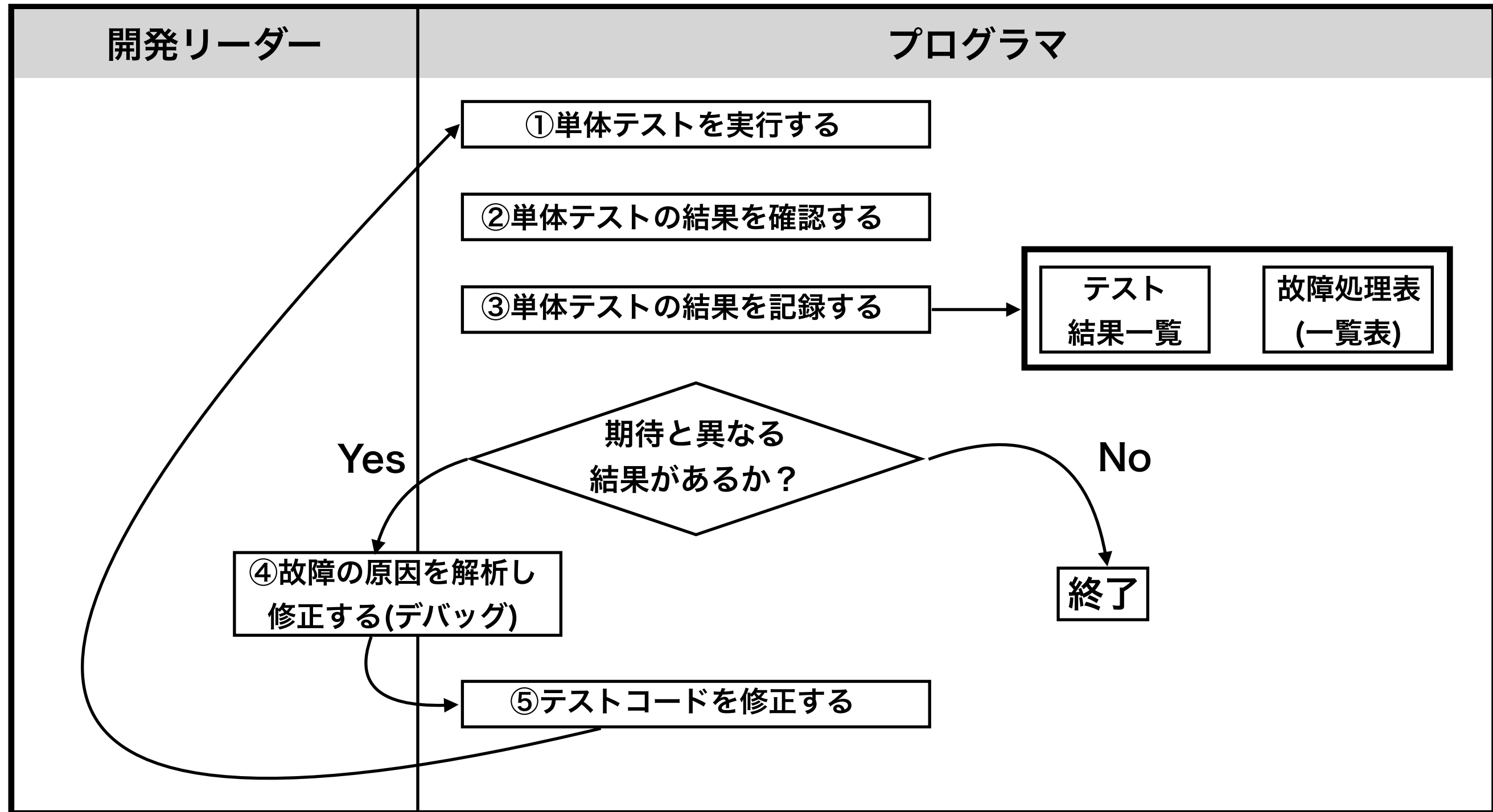
単体テスト中に修正



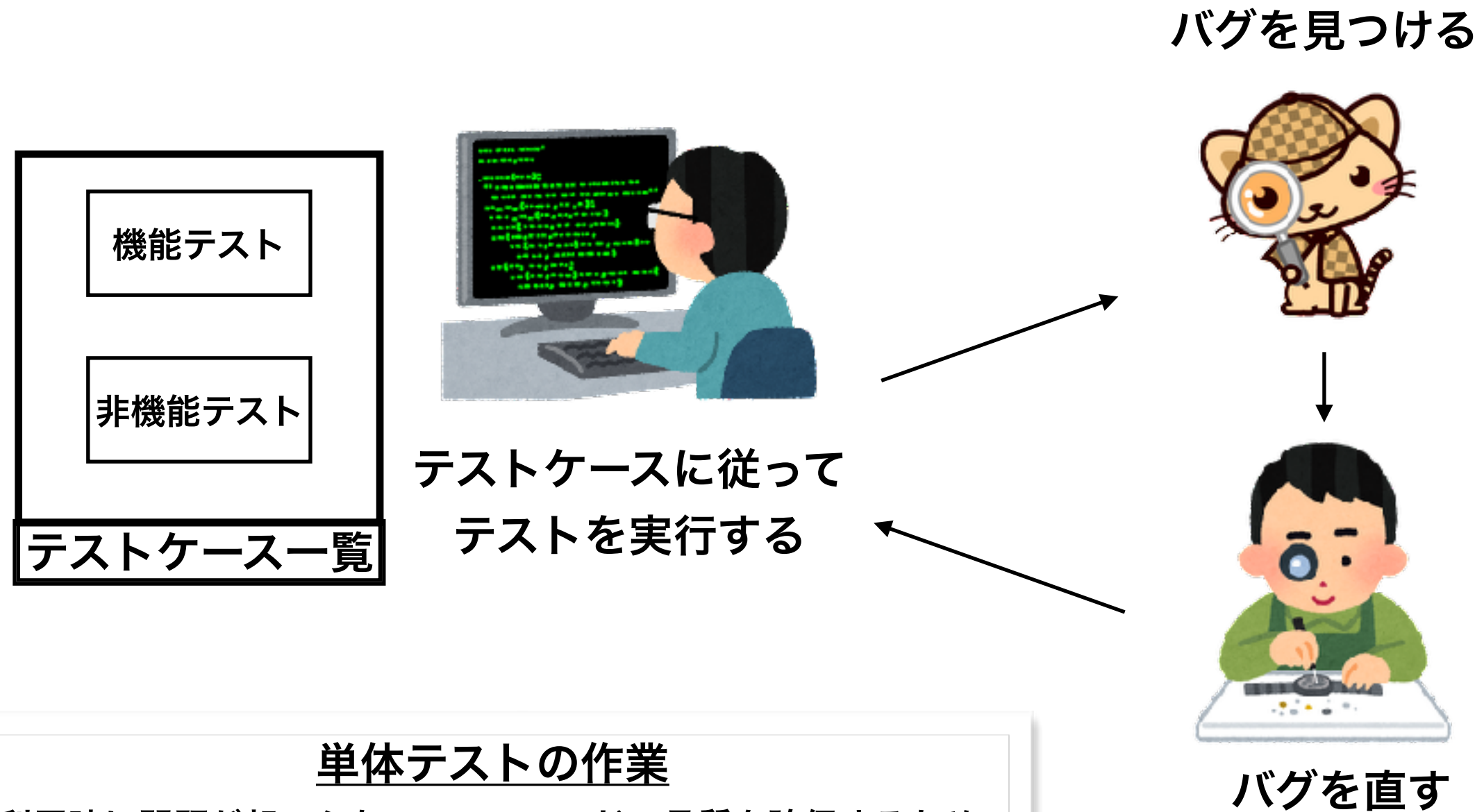
同じテストケースで
異なる結果となる



単体テスト実行の手順



単体テストの作業



単体テストの作業

利用時に問題が起こらないソースコードの品質を確保するため、
ソースコードのバグを見つけて修正する

非機能テスト

- 非機能要件

- * あるソースコードの機能を実現する際に、どのくらい遠く、少ないリソースで実現できるかという、処理ロジック以外の視点での要件

- ▶ リソース：記憶容量, 通信データ量など

- 精巧な地図が表示できるアプリケーションでも…

- * 表示時間に何時間もかかる・膨大な量のパケットが送られてくる

- ソースコードだけでなく非機能についてもユーザの満足を得るレベルにあるかどうかのテストを実施する必要がある

非機能要件のテストとは

- 非機能要求 = 性能
 - * プロファイリングの実施
 - ▶ ソースコードの実行に要する時間
 - ▶ 実行時のメモリ使用量などを分析すること
- ソースコードの実行・記録だけでなくCPU利用率やメモリ利用量などの情報も記録しなくてはならない
 - * 非機能要件のテストには、プロファイリングツールなどのツールが利用される

故障の原因を解析し修正する(デバッグ)

テストで故障が起こる原因リスト

原因	説明
設計書の記述ミス	そもそも設計が間違っていたために、それに合わせて書いたソースコードも間違っているケース
ソースコードのミス	ソースコードを作成する際に誤ったロジックを記述した場合や単純なプログラミングミス
テストケースのミス	テストケースでの期待値が間違っていて、実際のソースコードには問題がないケース
テスト環境のミス	テストで使用するツールやライブラリのバージョンが間違っているケース
テスト実施のミス	テストケースと異なる実行方法で実施して、実際のソースコードには問題がないケース
その他	テストで使用したツールやライブラリ自体にバグがあるケース

機能テストの実践



- ```

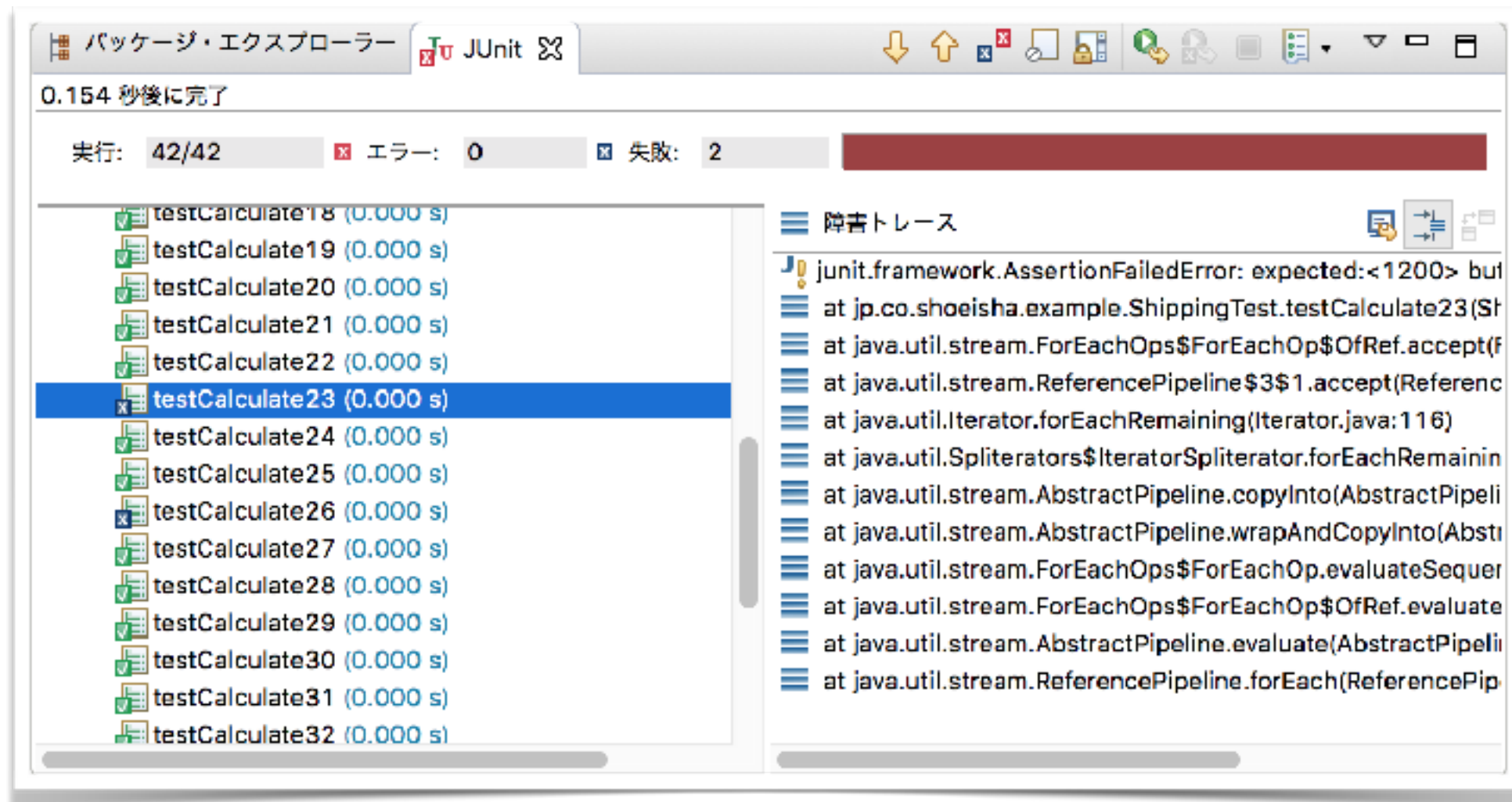
1 package jp.co.shoeisha.example;
2
3 /**
4 * 配送サービスに関するクラス
5 *
6 * @author 山田太郎
7 */
8 public class Shipping {
9
10 /**
11 * 配送料を計算する
12 */
13 private static final int SIZE_S = 1;
14
15 /**
16 * 配送料を計算する
17 */
18 private static final int SIZE_M = 2;
19
20 /**
21 * 配送料を計算する
22 */
23 private static final int SIZE_L = 3;
24
25 /**
26 * 配送料を計算する
27 *
28 * @param weight 重量
29 * @param x 距離
30 * @param y 距離
31 * @param z 距離
32 * @return 配送料
33 */
34 public int calculate(int weight, int x, int y, int z) {
35 // 重量が0未満または30を超える場合
36 if (weight <= 0 || weight > 30) {
37 throw new IllegalArgumentException();
38 }
39 // 距離が0未満または100を超える場合
40 if (x <= 0 || y <= 0 || z <= 0 || x + y + z > 100) {
41 throw new IllegalArgumentException();
42 }
43 // 配送料を計算する
44 int total = x + y + z;
45 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
46 throw new IllegalArgumentException();
47 }
48 // 配送料を計算する
49 // 重量が0未満または30を超える場合
50 if (weight <= 0 || weight > 30) {
51 throw new IllegalArgumentException();
52 }
53 // 距離が0未満または100を超える場合
54 if (x <= 0 || y <= 0 || z <= 0 || x + y + z > 100) {
55 throw new IllegalArgumentException();
56 }
57 // 配送料を計算する
58 int total = x + y + z;
59 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
60 throw new IllegalArgumentException();
61 }
62 // 配送料を計算する
63 int total = x + y + z;
64 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
65 throw new IllegalArgumentException();
66 }
67 // 配送料を計算する
68 int total = x + y + z;
69 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
70 throw new IllegalArgumentException();
71 }
72 // 配送料を計算する
73 int total = x + y + z;
74 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
75 throw new IllegalArgumentException();
76 }
77 // 配送料を計算する
78 int total = x + y + z;
79 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
80 throw new IllegalArgumentException();
81 }
82 // 配送料を計算する
83 int total = x + y + z;
84 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
85 throw new IllegalArgumentException();
86 }
87 // 配送料を計算する
88 int total = x + y + z;
89 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
90 throw new IllegalArgumentException();
91 }
92 // 配送料を計算する
93 int total = x + y + z;
94 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
95 throw new IllegalArgumentException();
96 }
97 // 配送料を計算する
98 int total = x + y + z;
99 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
100 throw new IllegalArgumentException();
101 }
102 // 配送料を計算する
103 int total = x + y + z;
104 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
105 throw new IllegalArgumentException();
106 }
107 // 配送料を計算する
108 int total = x + y + z;
109 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
110 throw new IllegalArgumentException();
111 }
112 // 配送料を計算する
113 int total = x + y + z;
114 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
115 throw new IllegalArgumentException();
116 }
117 // 配送料を計算する
118 int total = x + y + z;
119 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
120 throw new IllegalArgumentException();
121 }
122 // 配送料を計算する
123 int total = x + y + z;
124 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
125 throw new IllegalArgumentException();
126 }
127 // 配送料を計算する
128 int total = x + y + z;
129 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
130 throw new IllegalArgumentException();
131 }
132 // 配送料を計算する
133 int total = x + y + z;
134 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
135 throw new IllegalArgumentException();
136 }
137 // 配送料を計算する
138 int total = x + y + z;
139 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
140 throw new IllegalArgumentException();
141 }
142 // 配送料を計算する
143 int total = x + y + z;
144 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
145 throw new IllegalArgumentException();
146 }
147 // 配送料を計算する
148 int total = x + y + z;
149 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
150 throw new IllegalArgumentException();
151 }
152 // 配送料を計算する
153 int total = x + y + z;
154 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
155 throw new IllegalArgumentException();
156 }
157 // 配送料を計算する
158 int total = x + y + z;
159 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
160 throw new IllegalArgumentException();
161 }
162 // 配送料を計算する
163 int total = x + y + z;
164 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
165 throw new IllegalArgumentException();
166 }
167 // 配送料を計算する
168 int total = x + y + z;
169 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
170 throw new IllegalArgumentException();
171 }
172 // 配送料を計算する
173 int total = x + y + z;
174 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
175 throw new IllegalArgumentException();
176 }
177 // 配送料を計算する
178 int total = x + y + z;
179 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
180 throw new IllegalArgumentException();
181 }
182 // 配送料を計算する
183 int total = x + y + z;
184 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
185 throw new IllegalArgumentException();
186 }
187 // 配送料を計算する
188 int total = x + y + z;
189 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
190 throw new IllegalArgumentException();
191 }
192 // 配送料を計算する
193 int total = x + y + z;
194 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
195 throw new IllegalArgumentException();
196 }
197 // 配送料を計算する
198 int total = x + y + z;
199 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
200 throw new IllegalArgumentException();
201 }
202 // 配送料を計算する
203 int total = x + y + z;
204 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
205 throw new IllegalArgumentException();
206 }
207 // 配送料を計算する
208 int total = x + y + z;
209 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
210 throw new IllegalArgumentException();
211 }
212 // 配送料を計算する
213 int total = x + y + z;
214 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
215 throw new IllegalArgumentException();
216 }
217 // 配送料を計算する
218 int total = x + y + z;
219 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
220 throw new IllegalArgumentException();
221 }
222 // 配送料を計算する
223 int total = x + y + z;
224 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
225 throw new IllegalArgumentException();
226 }
227 // 配送料を計算する
228 int total = x + y + z;
229 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
230 throw new IllegalArgumentException();
231 }
232 // 配送料を計算する
233 int total = x + y + z;
234 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
235 throw new IllegalArgumentException();
236 }
237 // 配送料を計算する
238 int total = x + y + z;
239 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
240 throw new IllegalArgumentException();
241 }
242 // 配送料を計算する
243 int total = x + y + z;
244 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
245 throw new IllegalArgumentException();
246 }
247 // 配送料を計算する
248 int total = x + y + z;
249 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
250 throw new IllegalArgumentException();
251 }
252 // 配送料を計算する
253 int total = x + y + z;
254 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
255 throw new IllegalArgumentException();
256 }
257 // 配送料を計算する
258 int total = x + y + z;
259 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
260 throw new IllegalArgumentException();
261 }
262 // 配送料を計算する
263 int total = x + y + z;
264 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
265 throw new IllegalArgumentException();
266 }
267 // 配送料を計算する
268 int total = x + y + z;
269 if (x <= 0 || y <= 0 || z <= 0 || total > 100) {
270
```

# コードについて

- 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.
- \* <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
- 「P.177」という題名のファイルに今回のコードがあります

# JUnitの実行

- 前回と同じ手順でShippingTest.javaを実行する
- すると2箇所テストが失敗する



# 機能テストの結果の記録

- テストが失敗した2つのケースについて

- \* testCaculate23の詳細

- ▶ 値として1200が期待されているのに1800が出力された

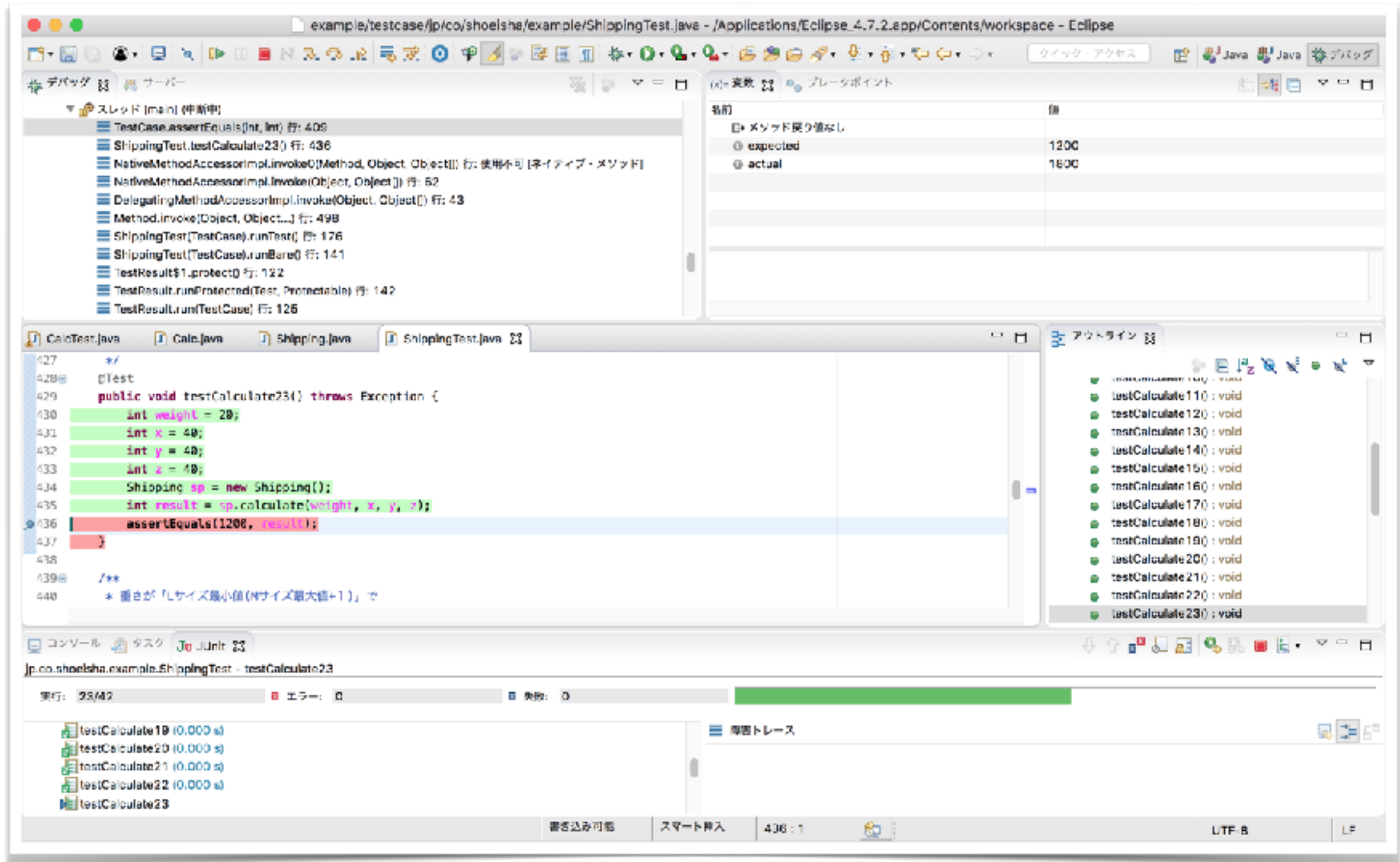
- 故障一覧表に今回の内容を記載しておく

# 故障の原因の解析と修正(デバッグ)

- デバッグの実行
  - \* パッケージエクスプローラでテストクラス  
「ShippingTest.java」を右クリック
    - ▶ 「デバッグ」→「JUnitテスト」を選択
    - ▶ デバッグを実行する

# 故障の原因の解析と修正(デバッグ)

## デバッグ実行後の画面



The screenshot shows the Eclipse IDE interface during a debug session. The main editor displays the `ShippingTest.java` file, with the `testCalculate23()` method highlighted. The console at the bottom shows the test results, indicating that the test passed. The variables view on the right shows the state of the program, including the `expected` and `actual` values for the `testCalculate23()` method.

| 名前        | 値    |
|-----------|------|
| メソッド戻り値なし |      |
| expected  | 1200 |
| actual    | 1800 |

# バグの修正

- 今回共に失敗しているのはMサイズ表示について
  - \* Shipping.javaのMサイズ表示について確認してみると、99行目が**if** (**total** < 120)となっている
    - ▶ 3辺合計120cm以下がMサイズなので**if** (**total** <= 120)にすべき

# テストの再実行

- 修正後JUnitテストを再度実行し、バグが修正されたことを確認する
  - \* この際、修正によってデグレード(新たなバグ)が発生していないかを確認するため、テストメソッドではなくテストクラスに対してテストを実行する必要がある
    - ▶ パッケージ・エクスプローラーでShippingTestクラスを右クリック→「実行」→「JUnitテスト」