Vol.2013-IS-125 No.3
2013/9/12

# 消費税計算のための概念モデルの作成と試実装

亀井邦裕[†1]　児玉公信[†2]　細澤あゆみ[†2]　成田雅彦[†3]

ビジネスシステムにおいて代金計算に伴う消費税の計算は必須であるが，消費税率の段階的引き上げや特例の導入などが見込まれている中で，税制改定に追随し，適用時点に応じた税額計算を，合理的に行う必要がある．ここには，過去の取引データの遡及的な修正も含む．このための概念モデルのあるべき姿を，取引（商流）データおよび代金（金流）データのモデルを含めて提案する．

# Conceptual Modeling for Consumption Tax Calculation and the Test Implementation

KUNIHIRO KAMEI[†1]　KIMINOBU KODAMA[†2]　AYUMI HOSOZAWA[†2]　MASAHIKO NARITA[†3]

Calculation of consumption tax accompanied with the price calculation is inevitable in the business information system. However, it will be more difficult to catch up the tax revision and calculate the tax reasonably in accordance with the applicable time, because of expected gradual increase in the tax rate and uncertain application of special exceptions. In addition, the calculation must include retrospective fix of transaction data in the past. In this paper, we propose the conceptual model for consumption tax calculation, which can be an ideal one, because it includes models of the transaction representing commercial distribution and billing as the example of money distribution, which are the organizational basic activities.

## 1. はじめに

現在，ビジネス系情報システムをとりまく法制度や規制は多様化し，ますます変化の速度を速めている．成長戦略による規制の緩和などで企業環境は大きく変化する．ビジネスの基本的活動は変わっていないとしても，その取り扱い商品は，サービスとの組合せにより構造が複雑化し，その代金の計算も多様化している．例えば，かつて固定電話のみを扱っていた通信企業は，今やモバイル電話，IP 電話，インターネットプロバイダ，CATV 等を融合させたサービスを提供しなくてはならない状況にある．

このようなビジネスの変化に絶えずさらされる企業情報システムには，変化に柔軟に対応できる工夫が組み込まれている必要がある．オブジェクト指向技術は，ソフトウェア工学が歴史の中で模索してきた，そのような工夫の 1 つである．しかし，実務では，たとえそのような理想を掲げたとしても，短納期に対応せざるを得ないために，開発ドメインの簡潔な概念モデル化，モジュール化され再利用性・拡張性を保証した実装モデル，可読性の高いソースコードなどというオブジェクト指向の原則を逸脱せざるをえず，知識として蓄積されにくい状況にある．またその原則についても，教育や書籍で語られるものは，あくまでサンプル，教科書のレベルを出ていない．実務に直結した題材を扱い，かつ実装による確実な動作実証を行った例は極めて少ない．

消費税計算は，企業の基本的活動である代金計算には必須の機能である．現行の 5%から 8%，10%への税率変更が決定している．税率が改定されるときには，低減税率が導入される．これは低所得者向け商品等に適用されると見られるが，具体的にどの商品かは未定である．そのため決定には多くの論争が予想され，10%適用直前に決定される可能性が高いと考えられる．

情報システムにおいては，もちろん現在この法改正の実施をにらんだ改修仕様検討は進んでいる．しかし，10%施行時については，低減税率適用商品が不明なことや，その他の特例について未定部分が多く，突っ込んだ議論は行われていない．

本報告は，この消費税を題材に，変化に対応できる情報システムのあるべき姿の一例を，オブジェクト指向技術を基礎にし，実装例を含むことにより実現性を明確にした形で示すことを目的とする．

## 2. 研究の概要

### 2.1 課題の認識

我々は変化に対応し，かつ持続性のある企業情報システムの姿を長年追い求めていた．個々の企業活動を，概念モデルを使って情報システムに写像することは，長年の活動の成果として実現しつつあった（たとえば，文献 3)）．しかし，その努力を知識として共有するためには，大きな障害があった．それはモデル化を追求すれば，そこに企業秘密としての戦略までモデル化されてしまう．それはモデル化される企業が著作権の形で保持するために，公開して共有し，改良していくことは不可能な状態にある．

†1 富士通(株)，Fujitsu Ltd.
†2 (株)情報システム総研，Information Systems Institute, Ltd.
†3 産業技術大学院大学，Advanced Institute of Industrial Technology

消費税計算に企業戦略の余地はない．法律で決定された制度であり，会計制度での扱いも明確である．したがってそのモデルは共有できる．しかも，消費税計算はビジネス系情報システムの特徴をよく備えている．

現状の消費税計算における課題を示す．

・情報システムは一般税率のみを対象とし，将来の変動を予測された形では設計されていない，かつ計算仕様は個別システムごとにばらばらである．
・計算処理の行われる範囲は POS 端末の表示，請求書の発行など，幅広い範囲に及ぶ．しかし標準化されたモジュールがないため，改定時の工数は多大である．
・直前の実務決定があるため，特に低減税率施行時には短期間の対応が求められる．
・実務上は，税率変更後の商品返品，購買時の税率が分からない返品への対応など，例外処理が数多く存在する．
・税率の異なる複数商品のセット販売など，税率決定後も実務上の課題は残る．

### 2.2 研究の目的

消費税計算の例を通して，企業活動あるいは業務の記録と利用における情報システム設計のあるべき姿の一例を提示したい．それは，制度や規制などのルール記述が一か所にまとめられ，かつ，そのルールによって制御される企業活動が分離された概念モデルになるはずである．企業活動はあまり変化することはないと考えられるが，ルールは，消費税の例を引くまでもなく，頻繁に変動する．ルールと企業活動が分離されていれば，それは変化に強いアプリケーションと言えるだろう．

本論文では消費税計算ルールや税率決定などの変動要素と，商流，金流という企業情報システムの不変要素を分離したモデル，実装コードの一例を提示する．それは税制変更を見込んだ消費税計算ライブラリ，標準化された企業活動のモデルである勘定パタンおよび両者のインタフェースの提供により実現される．

### 2.3 研究設問

上記目的を達成できる概念モデル，実装モデル，ソースコードが存在すると言えるか．達成しているとは，次の要件を満たしているものとする．

・概念モデルは予測される消費税制と実務の変動に耐えられること
・実装モデルはドメインの概念を継承し，モジュール化の原則が保持され，コードとの対応は容易であること
・コードは可読性が高く，一目で処理内容が理解できること
・その実装は今後の業務適用に向けて，実現性がみこまれるものであること

### 2.4 研究の方法

ルールと企業活動の分離がどのように設計され実装されていくかについて設計および実装作業の流れを参与観察し，記録した．これは，変化に強いアプリケーションとはいかなるものかを考察する資料となる．

作業は概略，以下のように進んだ．各作業は順次に実施するのではなく，必要に応じて手戻りし反復するという緩い順序性をもつ．

#### 2.4.1 消費税制度の調査

消費税計算の概念モデル化に当たって，消費税制，会計（一般会計と税務会計）実務および取引実務を調査した．調査の方法とその結果については次節にて述べる．

#### 2.4.2 概念モデルの作成

このような調査を行った上でルールを整理し，概念モデル [1)2)]に取り込んだ．ルール整理とそのモデル化が今回最も注力したところである．社内・社外取引の扱い，国内・国外（免税を含む）取引の扱い，課税品目（一般税率，個別税率，低減税率）・非課税品目の区別，計上日付（仕入/販売とその返品）の扱い，端数まるめルール，総額表示か明細表示か，外税計算をする場合（おもに企業間取引）と内税計算をする場合（おもに消費者との取引）などを盛り込んだ初期モデルを作成した．

#### 2.4.3 モデルの洗練

その上で企業活動である商流，金流の一般モデルと消費税計算部分を分離し，消費税計算部分を別ライブラリとした．同時に前記の計算ルールを企業間の契約によるもの(端数まるめルール，総額表示か明細表示か)，取引種別によるもの（社外・社内，国内・国外取引)，品目によるもの（課税・非課税，一般税率・個別税率・低減税率）を整理してモデルの各部分に配置した．企業で定義する品目の分類と税率を決定する品目とは異なるため，税率を決定する品目を税品目とし消費税計算過程に入る時点で変換できる仕組みを盛り込んだ．さらに洗練を重ね，消費税計算に照準を当てた企業活動の一般化された概念モデルの第 1 版を完成させた．

#### 2.4.4 ユースケースの記述

次にユースケースをユースケース記述に表現した.「契約を記録する」および「契約を記録する」のユースケースである．

「契約を記録する」ユースケースでは税の明細単位計算か合計計算か，さらにこの両方に違算が出たときの対処と，税額端数のまるめ方法（切り捨て，四捨五入，切り上げ）を新しく販売契約を締結した顧客との契約条項としてアクタ（例えば，契約担当者）が入力して記録することにした．

「注文を記録する」ユースケースでは，契約している顧客からの注文をアクタが入力し，システムに登録されている商品代金から消費税を計算し，表示し，記録することにした．ここでは，取引のあった日の税率により，消費税が

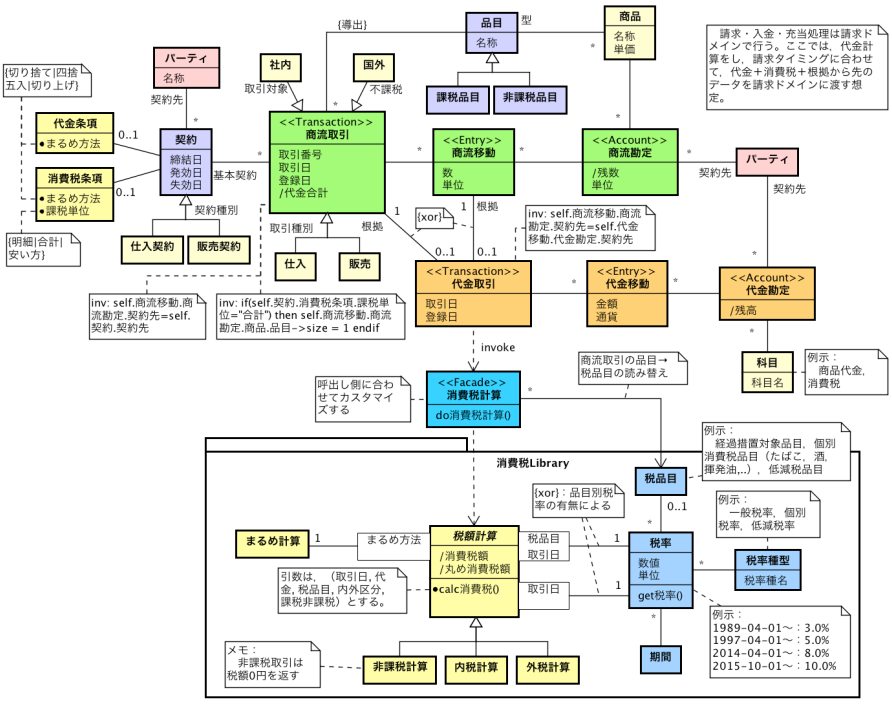



図 1　消費税計算の概念モデル

契約にある“まるめ方法”に従ってまるめられていることが確認できるようにすることで 2 つのユースケース間の整合をとるようにしている．

#### 2.4.5 テストシナリオの作成

アンチシナリオとして，一般税率の変更日を事前に設定しておき，その以前の日を取引した日とする入力，以降の日を取引した日とする入力，さらに変更日以前を取引した日とした注文記録を，変更日以降に取り消す（この場合はマイナスの注文として，変更日以前の税率で計算されなければならない）というシナリオを準備した．このようなシナリオに対しては，期間をもった税率の概念だけでなく，取引データも時間の概念が概念モデルに取り込まれている必要がある．取引データは，それを記録した日ではなく取引した日が基準になることも概念モデルで明確に記してある必要がある．

#### 2.4.6 実装モデルの作成

次に概念モデルを基に実装モデルを作成した．実装モデルとその解説は 3.2.1 に示す．ただし，低減税率や特に内税計算については試実装の対象外とした．

#### 2.4.7 テスト駆動によるコードの作成

実装では，シナリオに合わせ，テストコードを書くことから始めた．画面とのインタフェース，永続化に関しては今回のスコープ外として，引き続き実装モデルの改訂およびコードの追加および変更を行っていくことにした．

### 2.5 税制及び税計算実務の調査

調査の特徴は税制，会計実務（納税会計，一般会計），消費税計算実務の 3 つの観点から調査を行ったこと，実務調査では業種担当の技術者数名から複数の業種を調査したことである．これによって，完全ではないにしろ，消費税計算実務の一般形が示せている．

その調査結果の概略を以下に示す．

- 一般税率以外にたばこ税，酒税，揮発油税等の個別税率があり，さらに低減税率の導入が予定される．
- 国内取引のみにかかる税であり，国外取引は不課税となる．
- 非課税取引は品目により決まる．
- 税務会計と異なり実務では 1 円未満で端数調整を行う．
- 実務では社内取引も記録するが，消費税は計算されない．
- 一般に企業間取引は外税，消費者販売は外税または内税が併用されている．
- 取引全体の総額から計算する総額表示と，取引明細単位に計算する明細表示がある．この併用による税額の不整合については，事前の取り決めが必要である．

## 3. 結果

作成されたモデルとコードを以下に示す．なお，これらは試実装であることに注意されたい．

### 3.1 概念モデル

概念モデルを図 1 に示し，以下に概説する．

#### 3.1.1 消費税ライブラリ

消費税ライブラリは，消費税計算をアプリケーションから独立させるため，計算機能をライブラリとして提供するものである．特徴を以下に示す．

- Façade となる消費税計算クラスは呼び出し側に合わせ

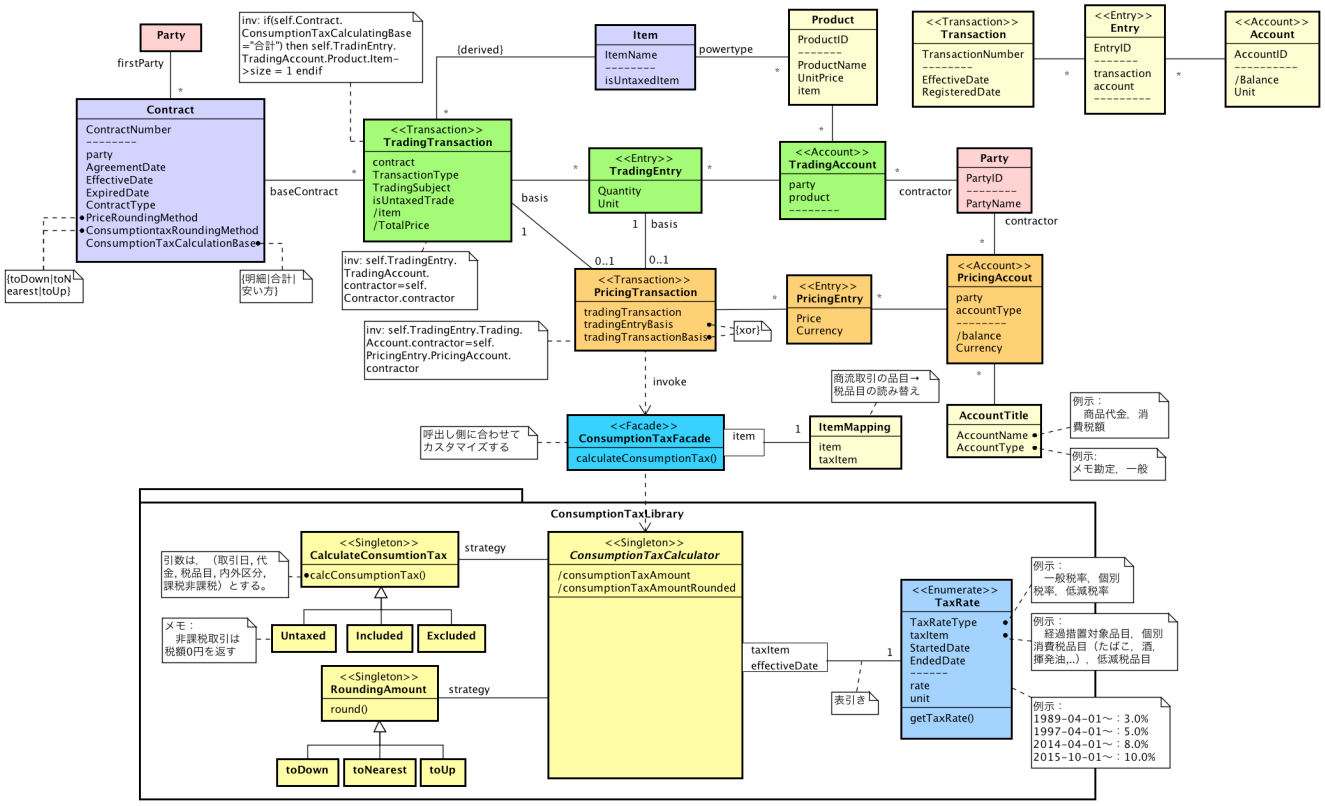



図 2　消費税計算の実装モデル

てカスタマイズ可能とする．

- 期間の概念を導入し，税率の期間変動に対応した．税率の決定要素が税品目，税率種型，期間であることを示している．
- 税額計算クラスを抽象クラスとし，非課税，内税，外税等の計算方式の多様性に対応できる．さらに個別税率の拡張などにも対応できる．
- 税額計算は取引日を限定子として税率クラスと関連している．これにより，期間とは取引日が含まれる期間であることを示している．
- 低減税率など消費税率を決定する品目と企業活動で使用される品目とは概念が異なるため，この対応をとるための処理を消費税計算 Façade で行うこととした．この変換規則は使用システムによって最適にカスタマイズされる．

### 3.1.2　アプリケーション部分

その他の部分は企業活動の一般的なモデルから消費税計算に関係する部分を抜粋したものである．中心は，消費税計算を発生させる売買取引を標準化した商流部分，および商品代金とその消費税を記録する金流部分である．この基本構造には勘定パタン [3]が使われている．勘定パタンは，取引（Transaction），移動（Entry），勘定（Account）の 3 つのクラスからなる構造を持っている．後述するユースケース記述の中では，これを“TEA”と省略して用いている．

国外取引，社内取引などの取引の種別は消費税計算が不要であり，消費税を科目とする金流オブジェクトは発生しない．

また，契約クラスの消費税条項の値と，品目クラスとそのサブクラスが，消費税計算の税率とまるめ方法，明細単位計算か合計単位計算かを指定する．この情報は合計単位計算の場合商流取引オブジェクトから直接，または明細単位計算の場合は商流移動オブジェクトを通じて代金取引オブジェクトに渡され，それが消費税計算を起動することを示している．

## 3.2　実装モデル

実装モデルを図 2 に示す．

### 3.2.1 実装モデルの解説

概念モデルから実装モデルを作成する際の，変換操作を中心に実装モデリングを解説する．

- エンティティとして永続化されるものには基本的にすべて識別子を設定した．識別子は属性表示中の破線から上の属性である．エンティティとは離散的にこの世の中に存在するモノや事象を表し，それらは識別されるべきである．Evans も著書「ドメイン駆動設計」[6]でこの考えを述べている．
- 概念モデルでサブクラスとして表記したものを，実装判断に基づき boolean データ型の属性とした．
- 消費税計算のロジック部については，唯一のインスタンスに計算操作を持たせるよう Singlton パタン [4]による実

装を施工者に指示した．

・非課税・内税・外税計算(CalculateConsumptionTax)とまるめ計算(RoudndingAmount)にはそれぞれ Strategy パタン[4]の適用を施工者に明示した．

・税率計算クラス(TaxRate)は列挙型とし，税率を表引きできるようにした．例示に属性値を税制表記している．これはすべて組み合わせても，数十種類にしかならないからである．この実装を軽くし，品目変換との組合せを工夫することが消費税計算適用の鍵の 1 つとなる．

### 3.2.2 ユースケース記述

ユースケース「注文を記録する」の記述から一部を示す．

2. 注文を記録する

アクタ：営業
概要： 取引先からの注文内容を基に，代金計算を行った上で妥当な注文を記録する。
目的： 適切に出荷し，代金を回収したい。
事前条件：商品がある。商流勘定と代金勘定は，すでに存在している場合がある。
事後条件：商流取引，商流移動がある。商流勘定がなかった場合は，それが新たに作られている。これに関連して，代金取引，代金移動がある。代金勘定がなかった場合は，それが新たに作られている。
基本系列：
①アクタはこのユースケースを起動する。
②システムは，注文内容（取引種別［販売］，不課税取引区分，取引対象区分，商品，数，取引先，取引日）の提示を求める。
③アクタはそれらを提示する。上記注文内容はすべて必須項目。
④システムは，課税単位が“明細”の場合は，商流取引を作成し，商品ごとに商流移動を作成し，該当する商流勘定（パーティ×商品の関連）を検索（なかったら生成）し，そこへのリンクを付ける。次に，商流移動ごとに代金取引を作成し，代金計算（UC 2.1 代金を計算する）および消費税計算（UC 2.2 消費税を計算する）を行い，代金勘定（パーティ×勘定）を検索（なかったら生成）し，代金計算および消費税計算の結果に基づいて，代金移動（代金と消費税の 2 つ），代金勘定へのリンクを作成する（修正の場合は上書き）。この結果をアクタに提示して確認を促す。
⑤アクタは確認する。
⑥システムは商流取引（TEA）と代金取引（TEA）を永続化する。
代替系列：
①基本系列⑤でアクタが確認しない場合は，本ユースケースの実行過程で生成したインスタンスは消去（rollback）し，このユースケースを終了する。
備考：
①代金計算，消費税計算は円建て。
②［販売］注文は販売契約に基づいて消費税計算を行う。按分率は省略する。
③代金計算で用いる科目は，商品代金:メモ勘定と消費税：メモ勘定とする。
④本実装では，内税計算をスコープ外とする。
⑤課税単位が「安い方」の場合は，両方のケースを行って消費税額の合計を比較して，安い方の代金取引（TEA）を記録する。
⑥商流移動で，バラの商品が複数個ある場合，代金計算は商品別の合計額を代金として扱う。

### 3.2.3 テストシナリオ

テストシナリオは，ユースケース記述の一部として記述され，ユースケースの理解とテスト駆動開発用のテストケースとして利用される．その一部を次に示す．

シナリオ：
①営業の山田直子は，7 月 30 日に麺屋海神から，鯛のアラ（課税品目）5.5 キログラム（273 円/キロ）とヒラマサのアラ(課税品目)8.3 キログラム（126 円/キロ）の注文を受け取り，販売取引，課税，社外取引として，その内容を記録した。代金計算の結果が，鯛のアラ 1501 円，ヒラマサのアラ 1045 円，消費税率は 5%，販売契約に基づき消費税は 127 円と表示され，これが正しかったので確認した。
②営業の山田直子は，8 月 1 日にも麺屋海神から，鯛のアラ 5.5 キログラムとヒラマサのアラ 8.3 キログラムの注文を受け取り，その内容を記録した。代金計算の結果が，鯛のアラ 1501 円，ヒラマサのアラ 1045 円，消費税率は 8 月 1 日から 8%になったので，消費税は 203 円と表示され，これが正しかったので確認した。

### 3.2.4 コード

次に実装モデルに基づいて作成されたコードを示す．コード全体の構造は，MVC のアーキテクチャ構造[5]となっている．言語は Java であり Java SE7 に準拠して作成した．開発環境には Eclipse を使用した．以下に示す例では，いわゆる setter/getter を省略している．

(1) **コード例** 1：ConsumptionTaxCalculator.java

消費税計算（CalculationConsumptionTax）およびまるめ計算（RoundingAmount）の条件による振る舞いの違いを，Strategy パタンで実装している．

```
public class ConsumptionTaxCalculator {

  private BigDecimal price;
  private Date effectiveDate;
  private TaxItem item;
  private boolean isTaxed;
  private CalculateConsumptionTax calculator;
  private RoundingAmount rounder;

  public ConsumptionTaxCalculator(
        BigDecimal price,
        Date effectiveDate,
        TaxItem item,
        Boolean isTaxed,
        CalculateConsumptionTax calculator,
        RoundingAmount rounder) {
    this.price = price;
    this.effectiveDate = effectiveDate;
    this.item = item;
    this.isTaxed = isTaxed;
    this.calculator = calculator;
    this.rounder = rounder;
  }

  public BigDecimal getConsumptionTaxAmount() {
    return calculator.calcConsumptionTax(effectiveDate,
        price, item, isTaxed);
  }

  public BigDecimal getConsumptionTaxAmountRounded() {
    return rounder.round(getConsumptionTaxAmount());
  }
}
```

(2) **コード例** 2：CalculateConsumptionTax.java

外税，内税，非課税ごとに消費税計算ロジックを用意する．Singleton の指定に対しては Java の enum 型を用いることで実現している．

```
public enum CalculateConsumptionTax {
  外税 {
    @Override
    public BigDecimal calcConsumptionTax(Date effectiveDate,
        BigDecimal price, TaxItem item, boolean isTaxed) {
      TaxRate rate = findRate(item, effectiveDate);
      BigDecimal number = BigDecimal.valueOf(rate.getRate());
      BigDecimal p = BigDecimal.valueOf(100);
      return price.multiply(number).divide(p);
    }
  },
  内税 {
      // スコープ外
    }
  },
  非課税 {
      // スコープ外
  };

  public TaxRate findRate(TaxItem item, Date effectiveDate) {
    return TaxRate.getTaxRate(item, effectiveDate);
  }

  public abstract BigDecimal calcConsumptionTax(
        Date effectiveDate,
        BigDecimal price,
        TaxItem item,
        boolean isTaxed);
}
```

(3) コード例 3：TaxRate.java

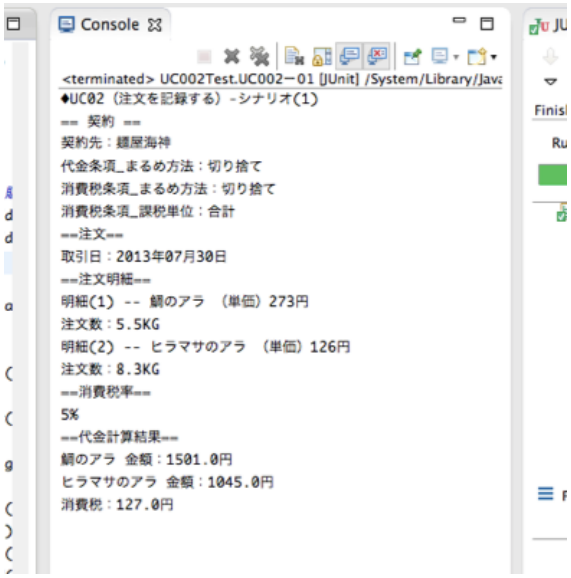
税率のクラス．実装モデルでは将来の拡張を考慮し，列挙型として表引きを行う指定になっているが，現実には税品目ごとの税率が定められていない．ここでは実装判断に基づき，単純な期間別の税率を定義した．

```
public class TaxRate {
  private TaxRateType type;
  private TaxItem taxItem;
  private Date startedDate;
  private Date endedDate;
  private int rate;
  private Unit unit;

  public static List<TaxRate>
        $instances = new ArrayList<TaxRate>();

  public TaxRate(TaxRateType type,
        TaxItem item,
        Date startedDate,
        Date endedDate,
        int rate,
        Unit unit) {
    this.type = type;
    this.taxItem = item;
    this.startedDate = startedDate;
    this.endedDate = endedDate;
    this.rate = rate;
    this.unit = unit;
  }

  public static TaxRate getTaxRate(TaxItem item, Date date) {
    for (TaxRate t : $instances) {
      if (t.getTaxItem() == item &&
          t.getStartedDate().compareTo(date) <= 0 &&
            t.getEndedDate().compareTo(date) >= 0 ) {
        return t;
      }
    }
    return null;
  }
}
```

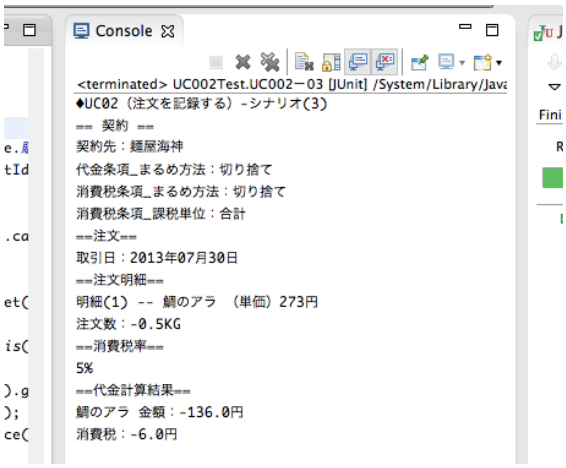
### 3.2.5 動作の確認

コードが設計どおりに稼働していることをテストツール JUnit4 の実行結果で確認する．下にテストツールのコンソール出力の当該部分をキャプチャしたものを提示する．テストケースは，上記ユースケースシナリオで一部を例示したものである．

(1) 画面サンプル 1：7 月 30 日取引の注文

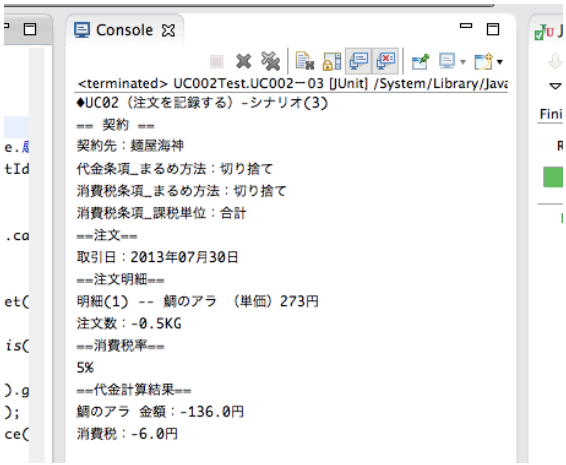



(2) 画面サンプル 2：8 月 1 日取引の注文




消費税が 8 月 1 日に 5%から 8%に改定されたことを仮定して前後の注文入力を実行している．消費税率が変わり税額が計算されている．複数明細の合計で計算していることにも着目されたい．

(3) 画面サンプル 3：7 月 30 日注文の遡及訂正




7 月 30 日に取引した注文を 8 月の消費税率変更後に取り消している．この時の税率は 5%で再計算されている．複数明細の一部訂正であるため，この場合遡及計算が必要となる．

## 3.3 成果物の評価

消費税計算を題材に取り上げることで，情報システムの現状における課題と解決策の例示ができた．

### 3.3.1 設計原則への準拠性

ルールと活動の分離，消費税計算部分の分離が行われ，勘定パタンが導入された概念モデルの標準形が得られた．ここから実装モデルおよびコードが導出されている．

### 3.3.2 概念モデルの良さ

概念モデルは，企業活動を商流・金流の基本部分と消費税計算を明確に分離している．消費税計算部分はライブラリの形で提供され，書庫から本を取り出すように，条件によって必要な機能を持つオブジェクトを取り出して実行する構造部品の概念を提唱している．税率を決める部分は，

業界，企業に合わせたカストマイズを可能とする形となっており，このモデルの柔軟性と再利用性を高めている．

#### 3.3.3 実装結果

実装は，概念モデルの洗練の期間よりはるかに短期間で行うことを与儀なくされた中で，最善が尽くされた．ユースケース記述を描く中でスコープを限定したが，スコープ外についてもかなりの議論がされており，今後の拡張においても，コードの構造に大きな影響はもたらさないと考えられる．実装モデルは，識別子の導入，Façade, Singleton, Strategy というデザインパタンの適用などの実装モデルの特徴が記された．TaxMapping クラスと TaxRate クラスで消費税カストマイズ部分の構造が明確になった．コーディングはこれを継承している．テストの結果もユースケースの実行が目に見える形で示されている．これは実装の証明を説明するための重要なプラクティスとして評価したい．

#### 3.3.4 課題

期間的な問題で画面と永続化の実装にまで至らなかったことに起因する課題があると考える．実装モデルからコーディングへの反映が十分でなかった．実装モデルの意図はコードに完全に反映されるべきであった．

## 4. 結論

本研究の目的を達成できる概念モデル，実装モデル，ソースコードが得られた．具体的に，2.3 で挙げた要件を満足しているかどうかの評価は次のとおりである．

予測される消費税制と実務の変動に耐えられる概念モデルを得ることができた．調査した範囲での消費税計算仕様はすべて概念モデルに盛り込まれた．消費税計算ルールは契約と品目クラスに定義され，税率決定ロジックと税額計算ロジックは消費税ライブラリに集めてある．

実装モデルに関しては，ユースケース記述でスコープが明確にされ，ドメインの概念を継承しながら，実装に向けて，概念モデルから改定されている．データの一意性，計算の整合性が取れる構造が作られている．デザインパタンを持ち込むことで，モジュール化の原則を守るように努められた．税率計算も分かりやすい表形式で纏められている．

コードについては実装モデル段階で小さなモジュールにする工夫，可読性を高める工夫が盛り込まれている．

実装に関しては今後の業務適用に向け，実現性を予測され得るものになっている．実務運用を想起できるユースケース記述が書かれている．実装結果の画面イメージがその実現を裏づけている．例えばスーパーマーケットの POS のレシートを見れば，このテスト結果画面の情報にいくつかの情報を加えただけのものであることが分かるであろう（通常のレシートは内税計算であるが試実装の例は企業間取引の慣例に習い，外税計算である）．

## 5. おわりに

この概念モデルおよびその実装は，繰り返すが試行レベルである．現実社会への適用性の判定は，今後の繰り返しの実装を待たなければならない．例えば POS の運用を見てみよう．レシートなしで返品に来たお客様については，税率を POS から外部入力する必要があるかもしれない．レシートデータが POS 自体への記録と同時に，店舗バックヤードにあるサーバに送信される運用は今や当たり前である．店舗が大型化すれば，多くの POS データをリアルタイムで処理しなければならない．しかし，POS の前に立つお客様は一瞬たりとも待ってはくれない．もともと POS の能力や資源には限りがある上，オブジェクト指向技術は資源を多く消費する構造をしている．試実装を本格運用に向かわせるためには，多くの課題があること，筆者たちはそれを理解したうえでさらにアプリケーションの標準化，部品化を提唱している．その課題を乗り越えれば少しでも多くの業務アプリケーションソフトウェアがこのモデルを継承して作成できることを期待している．この消費税計算の概念モデル化と試実装が，それに少しでも貢献できれば幸いである．

## 謝辞

杉野康弘さんに感謝します．筆者達が最初にこのモデルの実装化を試行したとき，わずかな時間で Ruby 言語による実装をしてくれました．その画面は正確に消費税を計算するように動作しました．これは多くの場面で実際に動作するモデルがあることの証明として役立ちました．

## 参考文献

1) 児玉公信：UML モデリング入門，日経 BP（2008）．
2) 児玉公信：UML モデリングの本質，日経 BP（2011）．
3) Fowler, M.，堀内監訳：アナリシスパターン，ピアソンエデュケーション，1998
4) Gamma, E. et al，吉田和樹ほか訳：オブジェクト指向における再利用のためのデザインパターン（改訂版），ソフトバンククリエイティブ（1999）．
5) Buschmann, F. et al，金沢典子ほか訳：ソフトウェアアーキテクチャ―ソフトウェア開発のためのパターン体系，近代科学社（2000）．
6) Evans, E 今関剛監訳：ドメイン駆動設計，翔泳社（2011）．