アナリスパターン第6章
在庫管理と会計

小島一朗
上手　裕

まえがき

ビジネスシステムのほとんどは、企業内の金の動き、すなわち、どのように金が入ってきてどのように出て行くかの記録を追跡するために作られている。会計および在庫管理の背後にある基本思想は、金と「もの」のさまざまな入れ物があること、そして金と「もの」がそれらの間でどう移動するか記録すべきだということである。
本章の「在庫管理と会計」パターンは、このような基本思想から生まれた。

目次

6．1　勘定

勘定は、品物やお金と行ったものの値を保持している。その値は、エントリによってのみの値の追加または削除（残高の更新）される。エントリは、勘定に対するすべての変更履歴を提供する。勘定を用いて、ある値の変更履歴を記録する場合、記載内容が失われないことを保証しなければならない。またエントリを借方／貸方で分けず数量の符号を使用する。もちろん、変種として借方／貸方のサブタイプを使用することも可能である。

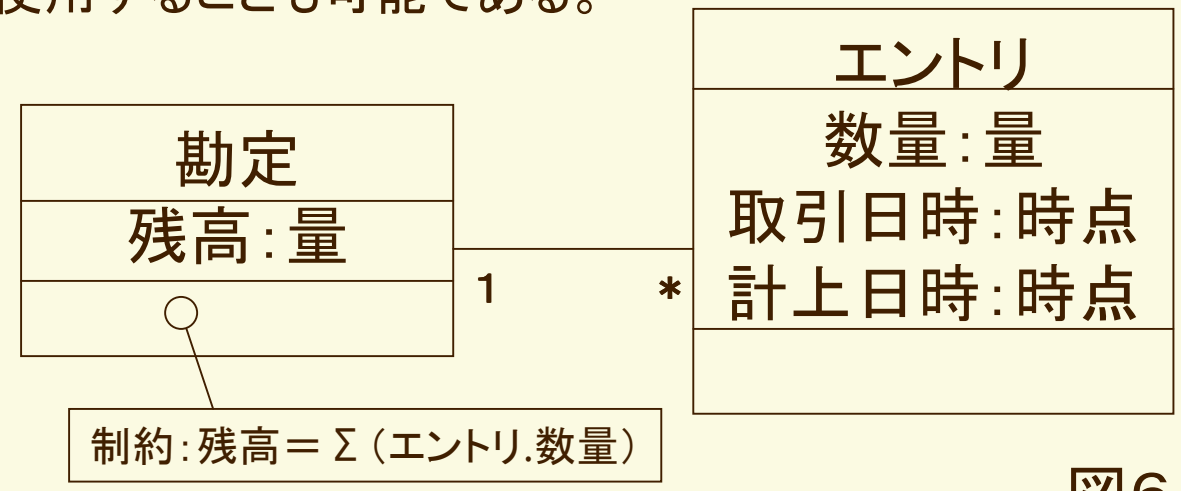


図6．1

6．1　勘定

モデリングの原則：ある値の変更履歴を記録するために、その値専用の勘定を使う。

6．1　勘定　（例題1）

筆者は自分の当座預金から100ドルを引き出す。これは、金額ー100ドルのエントリとして表され、これは当座預金に追記される。

当座預金：勘定	：エントリ
残高（前）－100ドル：量	－100ドル：量

※残高（前）とは、更新前残高という意味です。

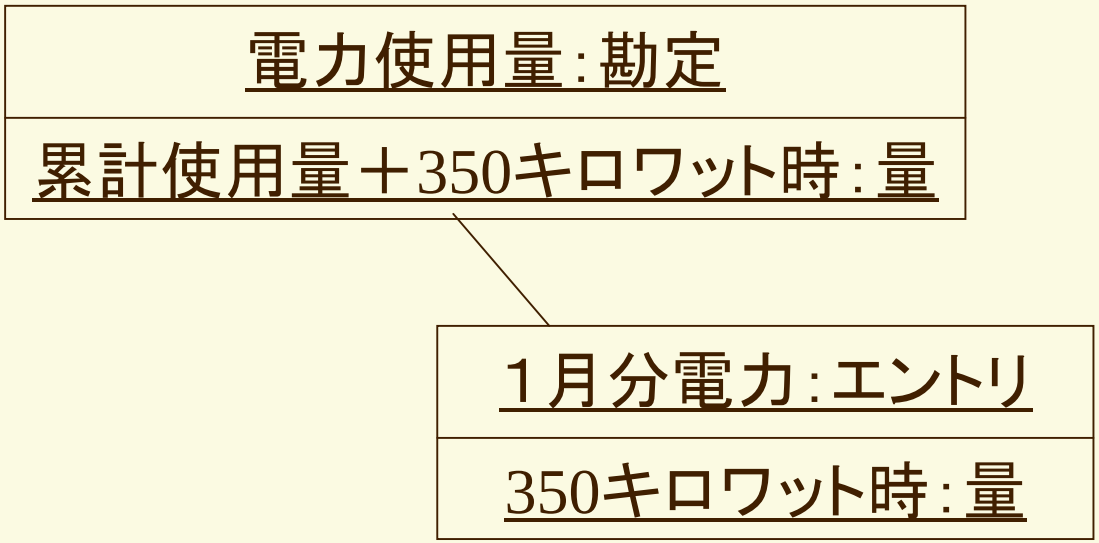
6．1　勘定　（例題2）

筆者はある店から標準便箋を4000枚買う。その店では、これを標準便箋勘定に、数量-4000枚のエントリとして表す。

標準便箋：勘定		：エントリ
残高（前）－4000枚：量		－4000枚：量

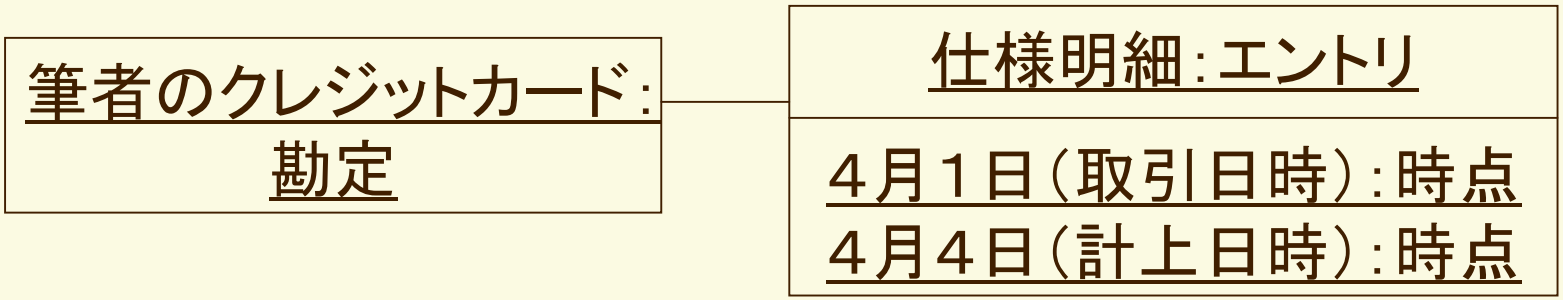
6．1　勘定　（例題3）

1月に350キロワット時の電力を使った。これは、家計簿の使用量勘定に、数量350キロワット時のエントリとして表される。



6．1　勘定　（例題4）

筆者は4月1日にジェーカフェで食事をした。クレジットカード会社は4月4日に支払いの通知を受け取った。このエントリは取引日が4月1日で、計上日が4月4日となる。



６．２　トランザクション

エントリの記載内容が多くなると、出入りを記録するだけでは不十分である。エントリがどこから来てどこに行くかも併せて記録するために「トランザクション」を設ける。
トランザクションは、ある勘定から払出と他への繰入を明示することでこれを促す。
２エントリ方式は、単に勘定間を移動するにすぎないという基本会計原則を反映している。

6．2　トランザクション

モデリングの原則：勘定を使うときは、保全原理に従え。
これによって、漏れの発見および防止が用意になる。

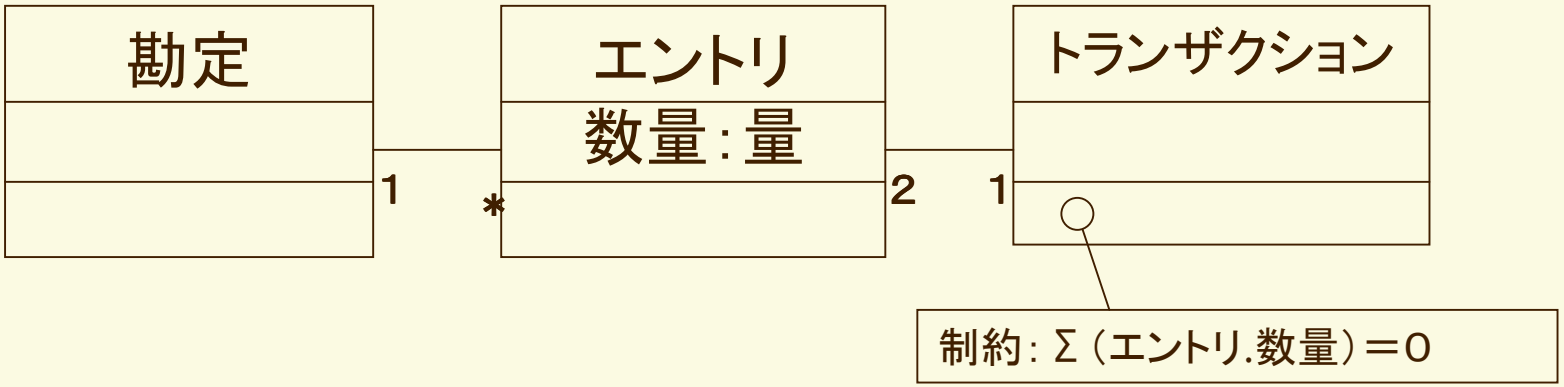


図6．2

6．2　トランザクション　（例題1）

筆者は航空券を買うためにボストン航空に500ドルを支払う。これはクレジットカード勘定からボストン航空への500ドルのトランザクションである。後日筆者は当座預金からクレジット勘定へ、クレジット勘定のバランスを0にするためのトランザクションを作成することになろう。

6．2　トランザクション　（例題1続き）

筆者は航空券を買うためにボストン航空に500ドルを支払う。これはクレジットカード勘定からボストン航空への500ドルのトランザクションである。後日筆者は当座預金からクレジット勘定へ、クレジット勘定のバランスを0にするためのトランザクションを作成することになろう。

6．2　トランザクション　（例題2）

ACM社は、5トンのアラビアンモカをニューヨークからボストンへ移送する。これは、ニューヨーク勘定からボストン勘定への5トンのトランザクションになる。

6．2．1　多岐トランザクション

トランザクションが全体としてバランスしている限り、複数の勘定にまたがってエントリを持つことができる。
多岐トランザクションは、2岐の場合よりもトランザクション作成における柔軟性を高める。

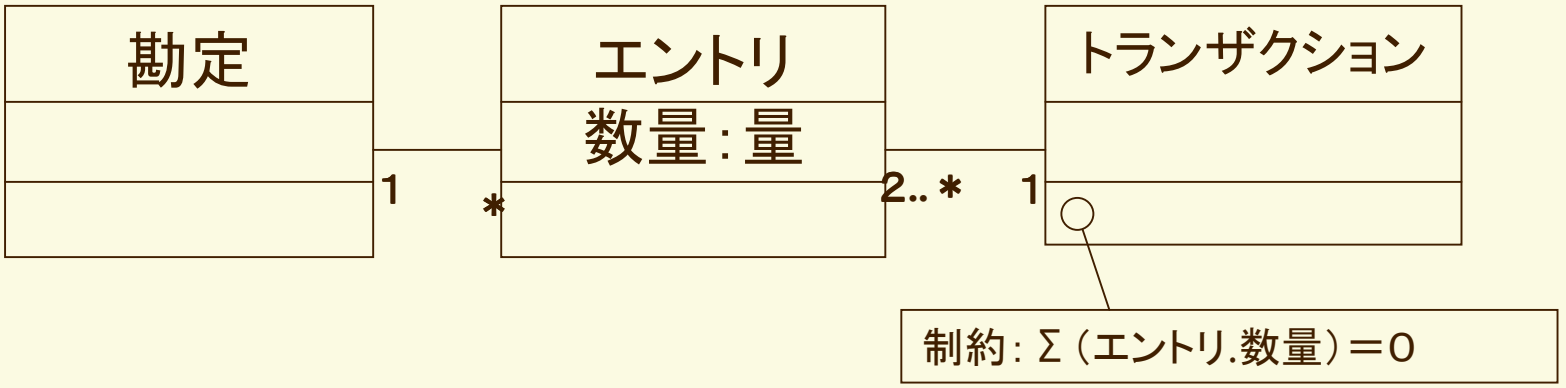


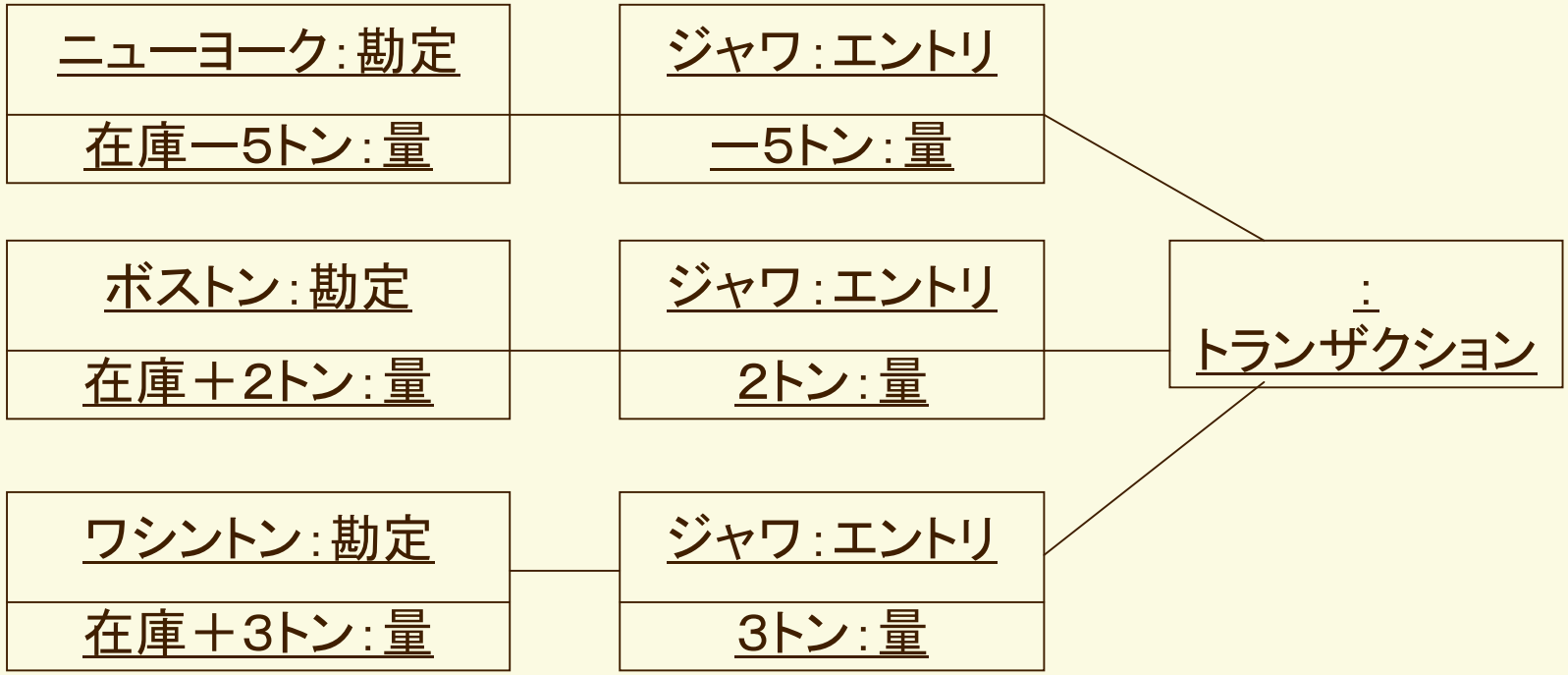
図6．3

6．2．1　多岐トランザクション　（例題）

ACM社は、ニューヨークからジャワ5トンのうち、2トンをボストンへ、3トンをワシントンへそれぞれ移送した。これは次のエントリを持つトランザクションである。[勘定：ニューヨーク、－5トン]、[勘定：ボストン、2トン]、[勘定：ワシントン、3トン]

6．2．1　多岐トランザクション （例題）

解答例



６．２．１　多岐トランザクション

トランザクションとエントリの関係：
制約によってトランザクションなしにエントリは生成できない。同様にエントリなしにトランザクションを生成する事もできない。

解決法：
部分的に定義されたエントリのリストを引数に取るようなトランザクションの生成操作を提供することである。エントリの生成操作は非公開にするが、トランザクションの生成操作だけにアクセスを許す。

6．3　要約勘定

要約勘定は、勘定のもつほとんどのレポート機能を勘定のグループにも当てはめたものである。
要約勘定は、要約と明細の2つの勘定で構成される。これは明細が葉となる階層構造になっている。要約勘定のエントリは、再帰的に、その構成要素の対象エントリから導出される。
構成要素間の関係が階層的であることは、明記されなければならない事に注意する。

6．3　要約勘定

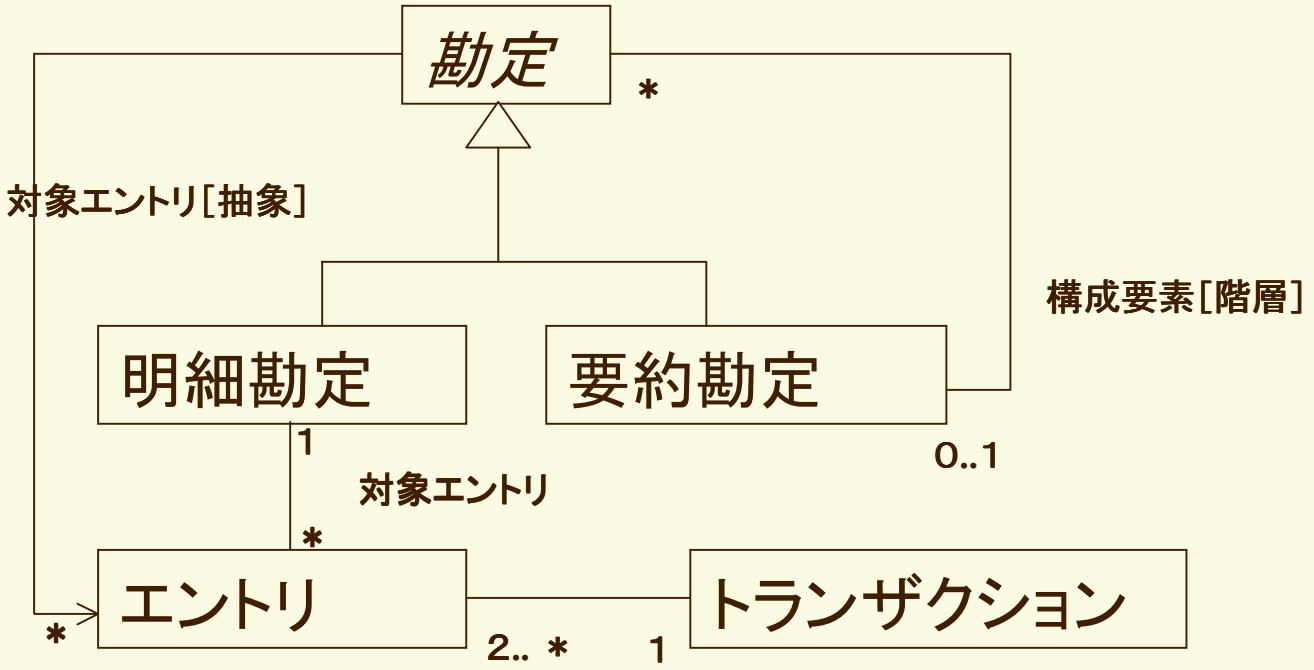
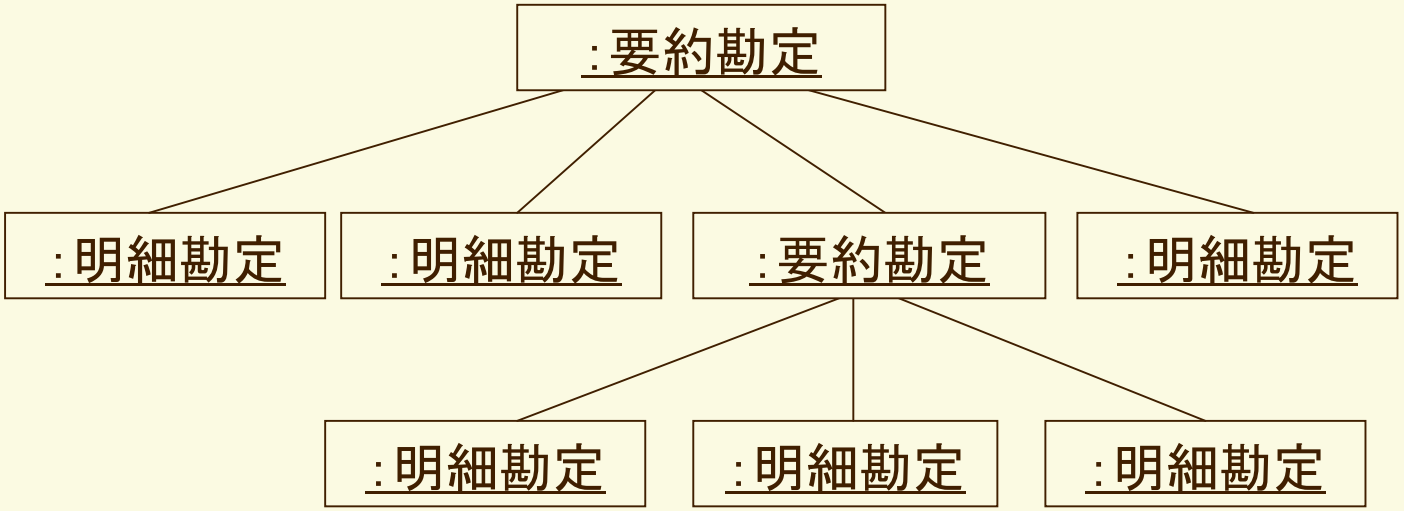


図6．5

要約勘定(例)

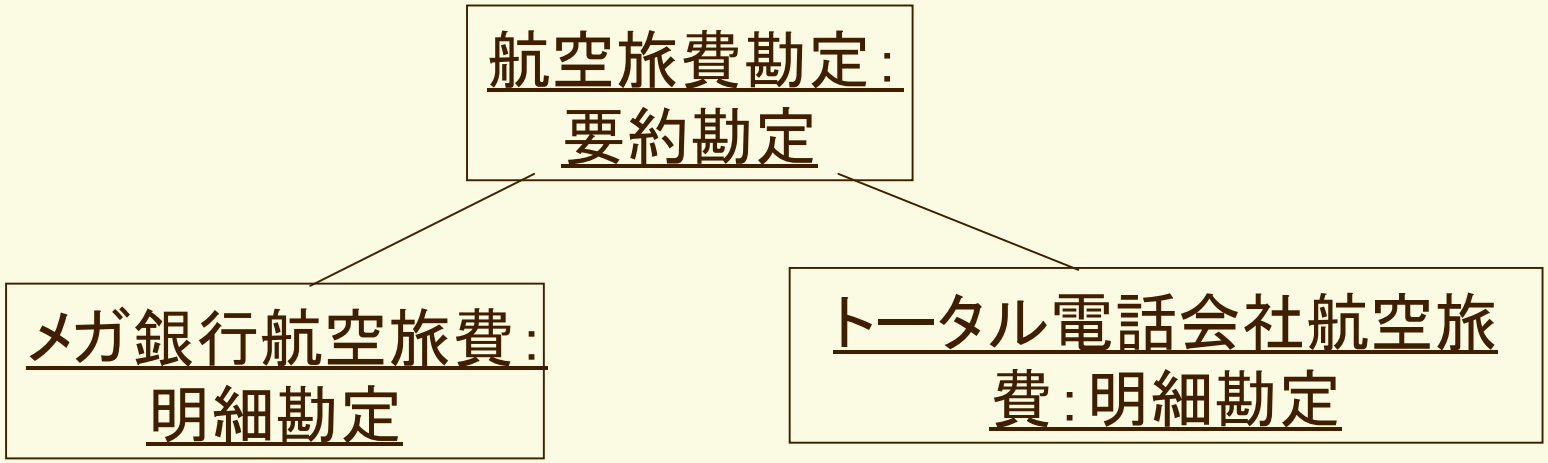


６．３　要約勘定　例題（１）

筆者は、メガ銀行航空旅費とトータル電話会社航空旅費の明細勘定に対する要約勘定として航空旅費勘定を設けている。

6．3　要約勘定　例題(1)

解答例

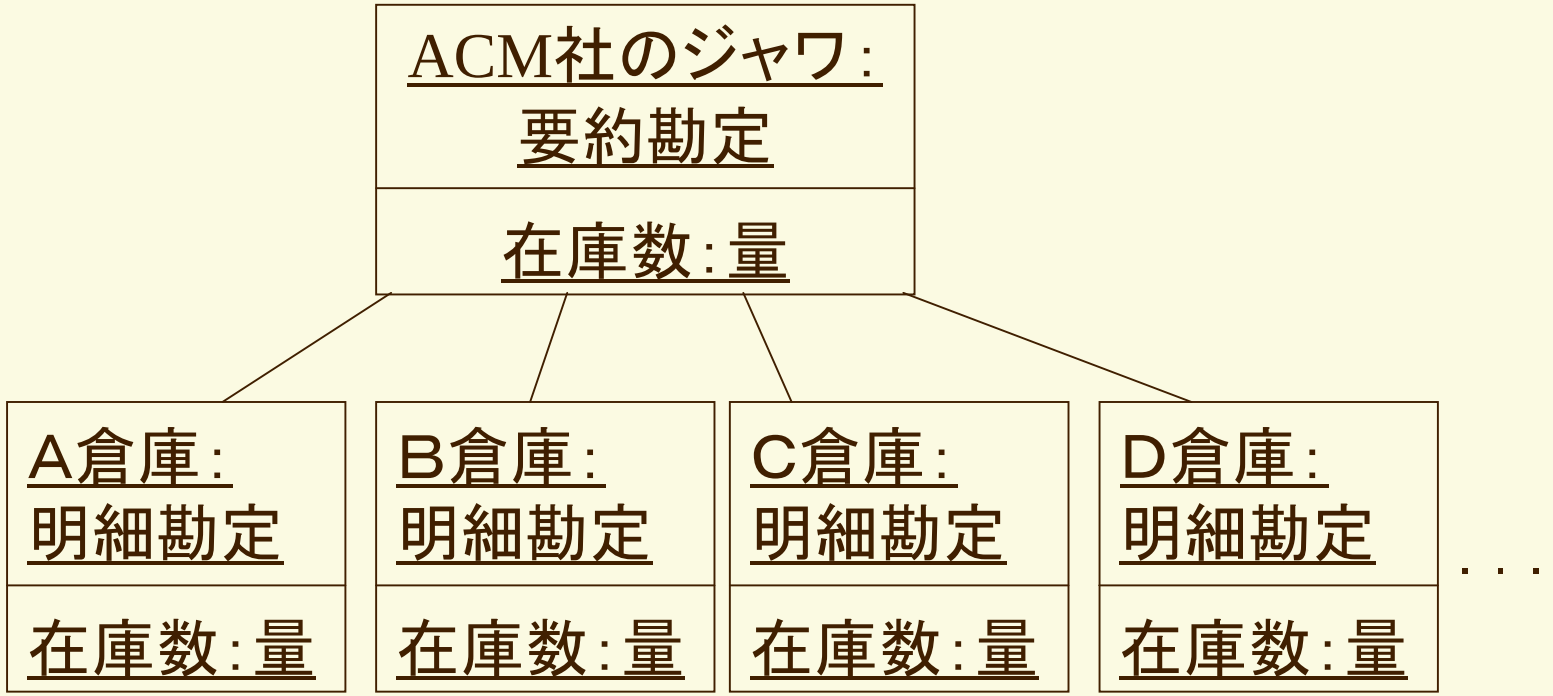


６．３　要約勘定　例題（２）

ACM社には、保管倉庫ごとの明細勘定をまとめたジャワの要約勘定がある。これで同社の抱えるジャワの合計数量がわかる。

6．3　要約勘定　例題(2)

解答例



6．3　要約勘定

要約と明細の勘定を分離しない例:
要約と明細の区別をなくした場合、いかなる勘定にもエントリを作成でき、すべての勘定を階層構造の中に置ける。これは勘定からエントリに2つの対応づけを与えることで可能になる。

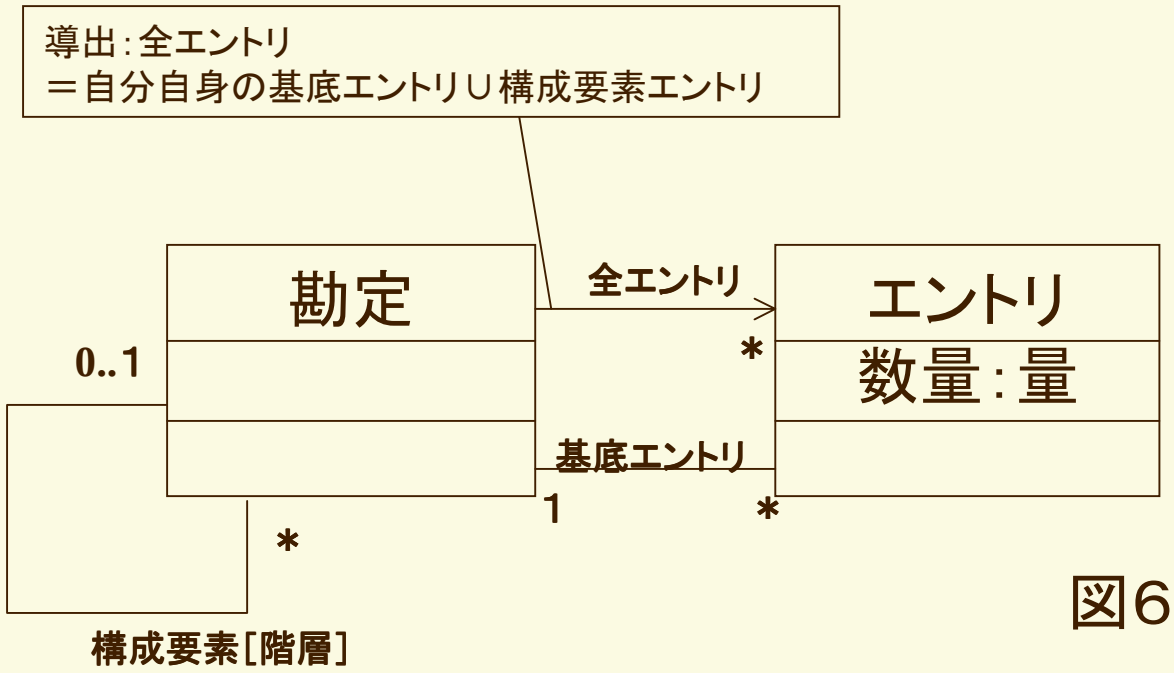


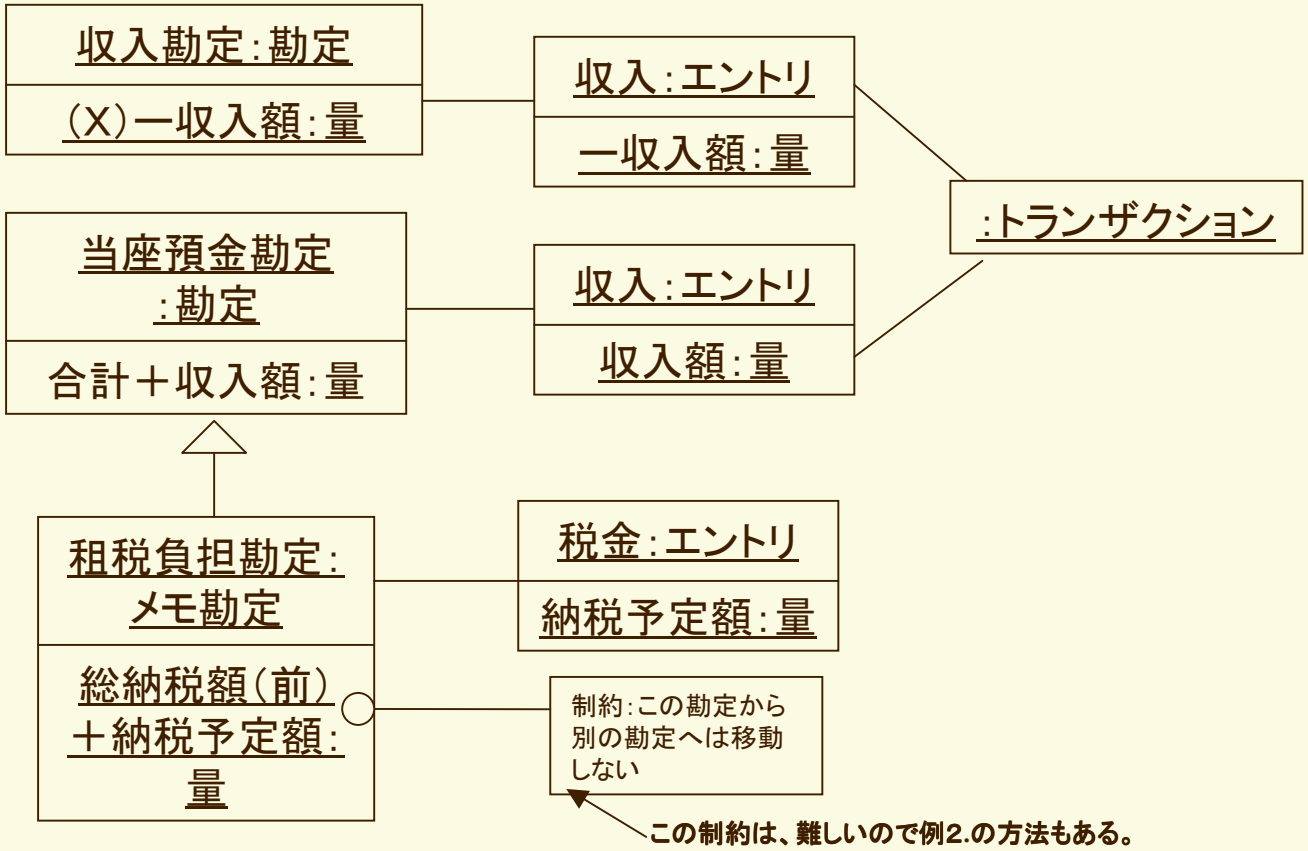
図6．6

6．4メモ勘定

メモ勘定:
バランスを持たないメモの役割をもつ勘定。

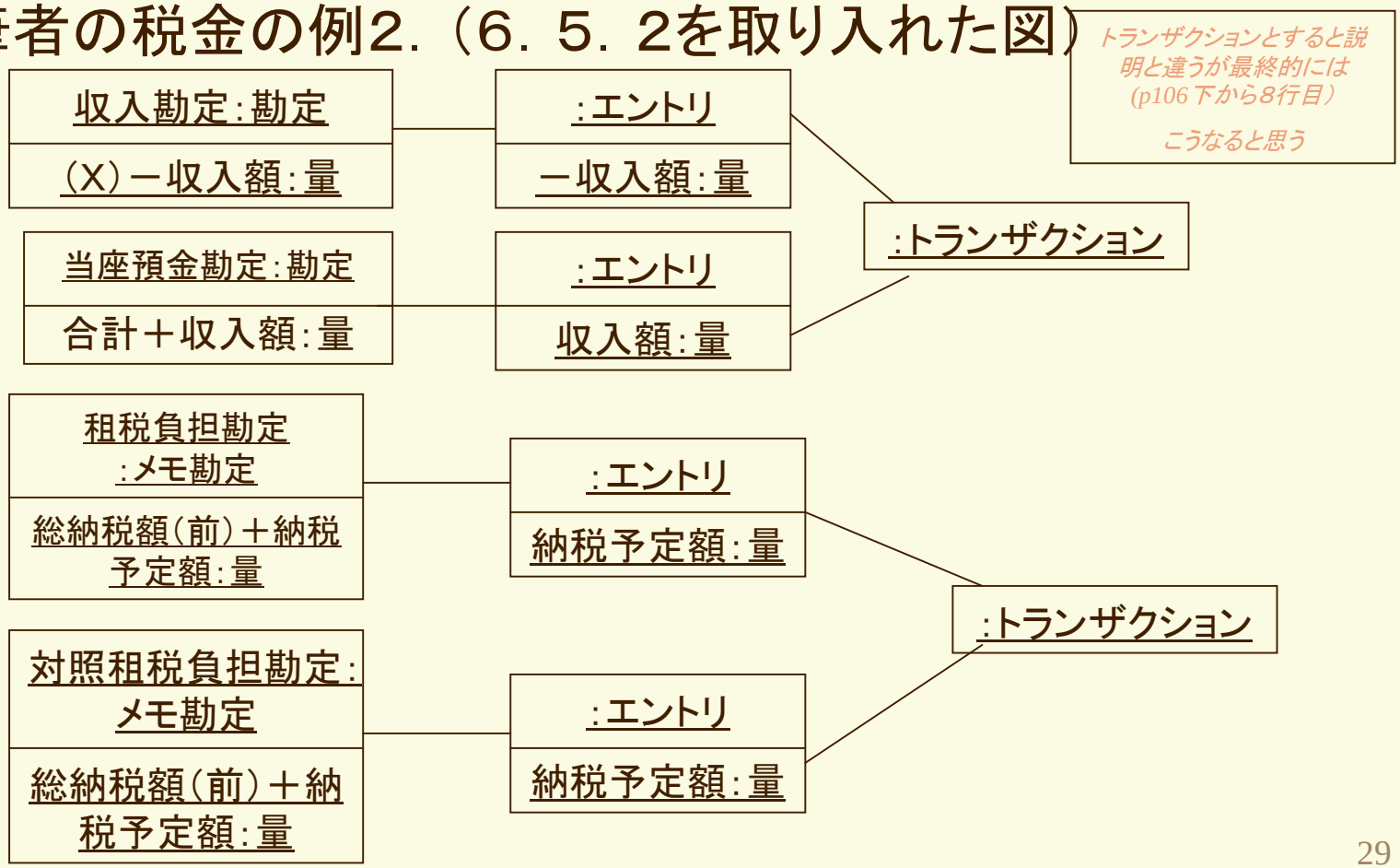
6．4　メモ勘定

筆者の税金の例1.



6．4　メモ勘定

筆者の税金の例2.（6．5．2を取り入れた図）

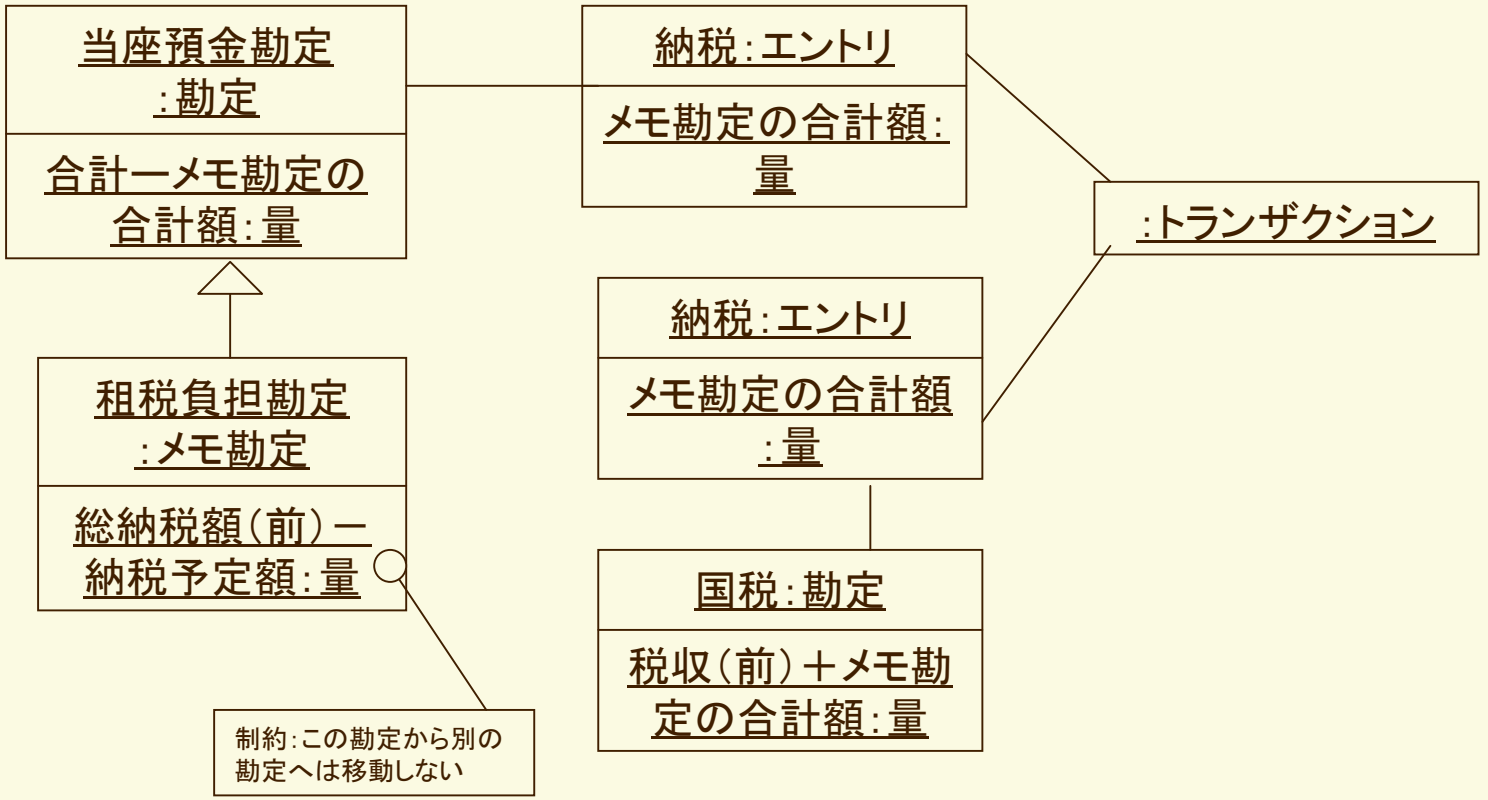


６．４　メモ勘定　（例題）

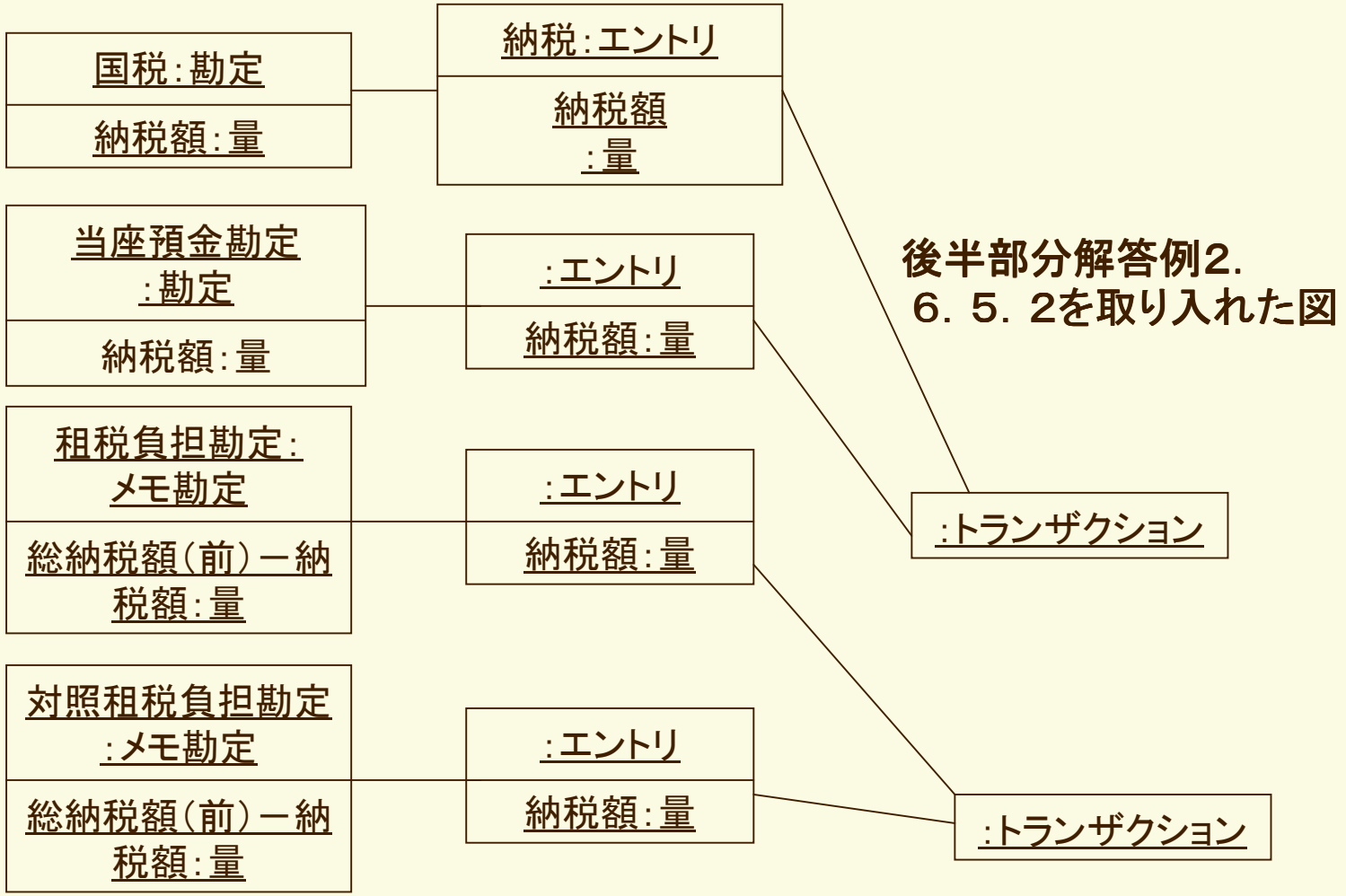
筆者は、依頼人から支払いを受けるたびに、それを収入勘定から当座預金へのトランザクションとして記録する。また、その一部を租税負担メモ勘定にも入れる。見込み税の支払時期になると、私は当座預金から国税勘定へのトランザクションを作成する。このトランザクションには、租税負担メモ勘定の合計額を支払額と同額だけ減らす、第３のエントリが加えられる。
（前半部分の回答は前ページ）

6．4　メモ勘定　（例題）

後半部分解答例1.



６．４　メモ勘定（例題）


国税：勘定
納税額：量
納税：エントリ
納税額
：量
当座預金勘定
：勘定
納税額：量
：エントリ
納税額：量
租税負担勘定：
メモ勘定
総納税額（前）－納
税額：量
：エントリ
納税額：量
対照租税負担勘定
：メモ勘定
総納税額（前）－納
税額：量
：エントリ
納税額：量
：トランザクション
：トランザクション
後半部分解答例2.
６．５．2を取り入れた図

6．5　転記ルール

転記ルール：
勘定には、金額や数量が勘定間をどのように移動するかを統制する固定的なルールを含めることができる。
図6．7では、転記ルールは、ある勘定をトリガとして指定することによって記述される。そのトリガとする勘定に対するどんな記載によっても、新たなエントリが作成される。

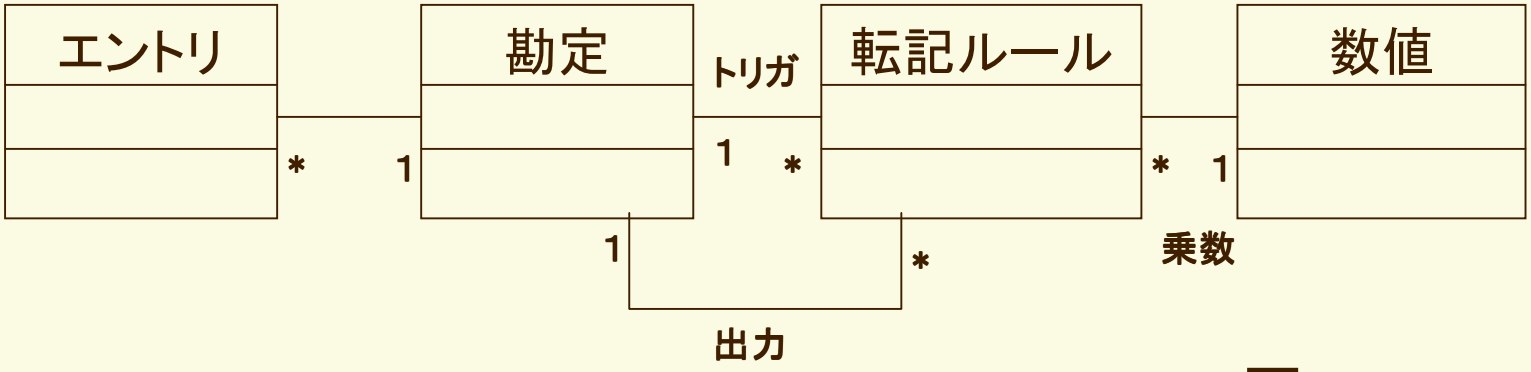
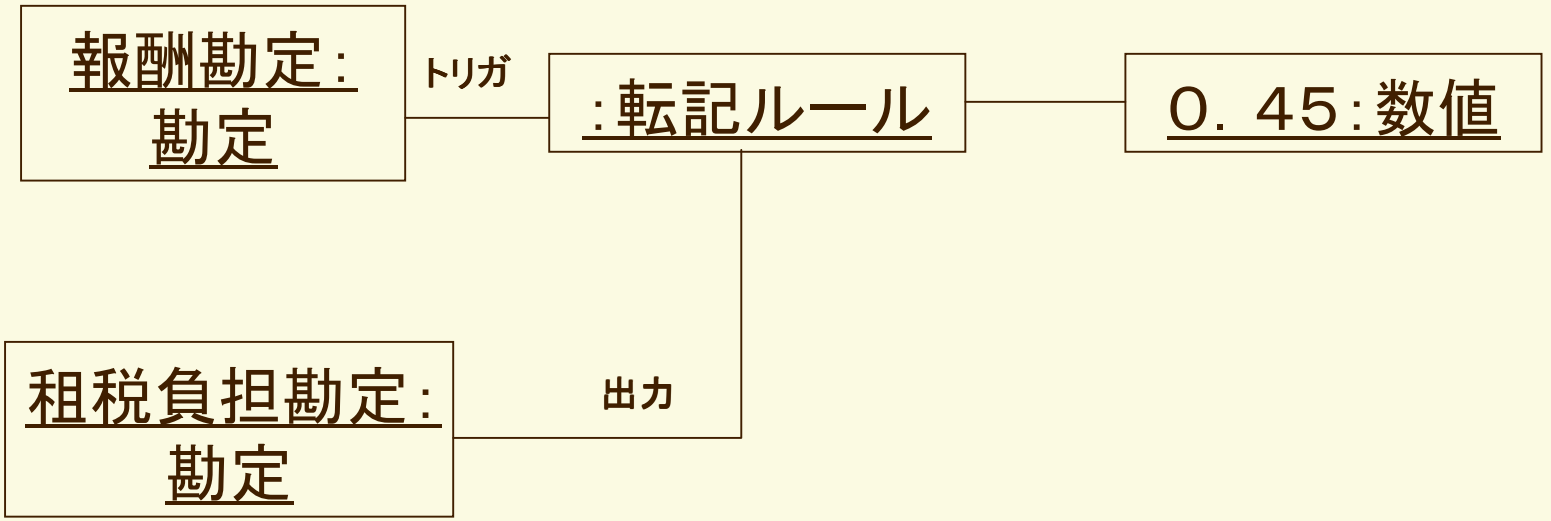


図6．7

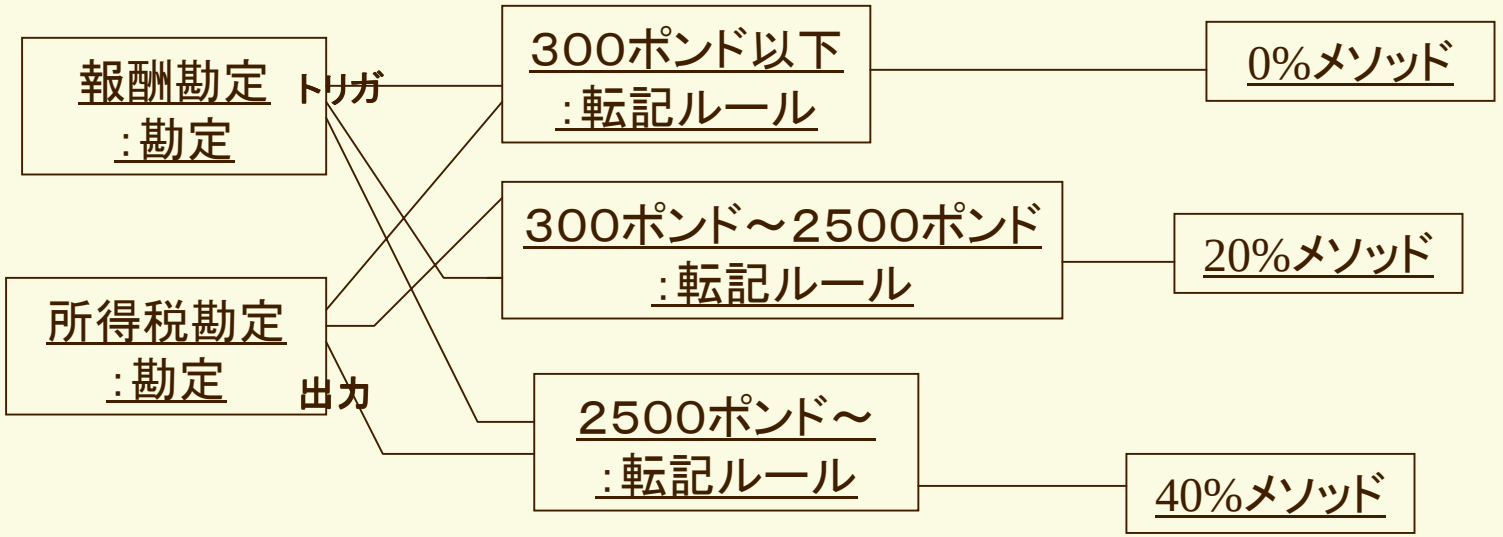
６．５　転記ルール　（例題）

筆者の租税負担は、トリガとしての報酬勘定をもつ転記ルール、出力先の租税負担勘定および乗数０．４５で扱える



6．5　転記ルール

累進課税の例：



６．５　転記ルール（続き）

転記ルールに、複雑な手続きを持たせるなど柔軟性を持たせるために、転記ルールのインスタンスそれぞれに計算をリンクする必要がある。

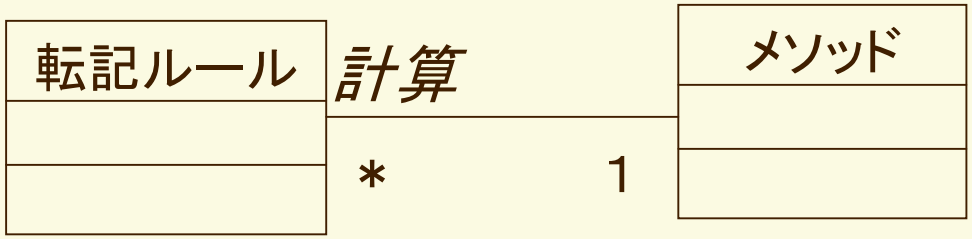


図６．８

６．５.1相殺可能性

転記ルールの重要な特性の１つは、「相殺可能」でなければならない。２つは、誤ったエントリを消去できないことである。

誤ったエントリの影響を取り除くためには、逆エントリを用いるしかない。

６．５．２トランザクションを使わない方法

すべてのエントリは、最初のエントリと転記ルールによって記述できるので、トランザクションを使わないことによるリスクは低減できる。
しかし．．．．．．．．
＜転記ルールだけ（トランザクションを使わない）場合＞
エントリごとの転記ルールが必要になり、エントリの合計が0になることを地道に確認する必要がある。

<トランザクションを使う場合>
１つの転記ルールから必ず２つのエントリを作成し、エントリの合計を地道に確認する必要はない。

６．６　個別インスタンスメソッド

転記ルールのインスタンスに固有の実行可能なメソッドを定義する一般的方法を説明する。

６．６．１　Singletonパターンを用いた実装

振る舞いを変えるための自然な方法は、サブクラス化に基づくポリモーフィックな操作を使うことである。これを行う最も簡単な方法がsingletonクラスをたくさん作ることである。

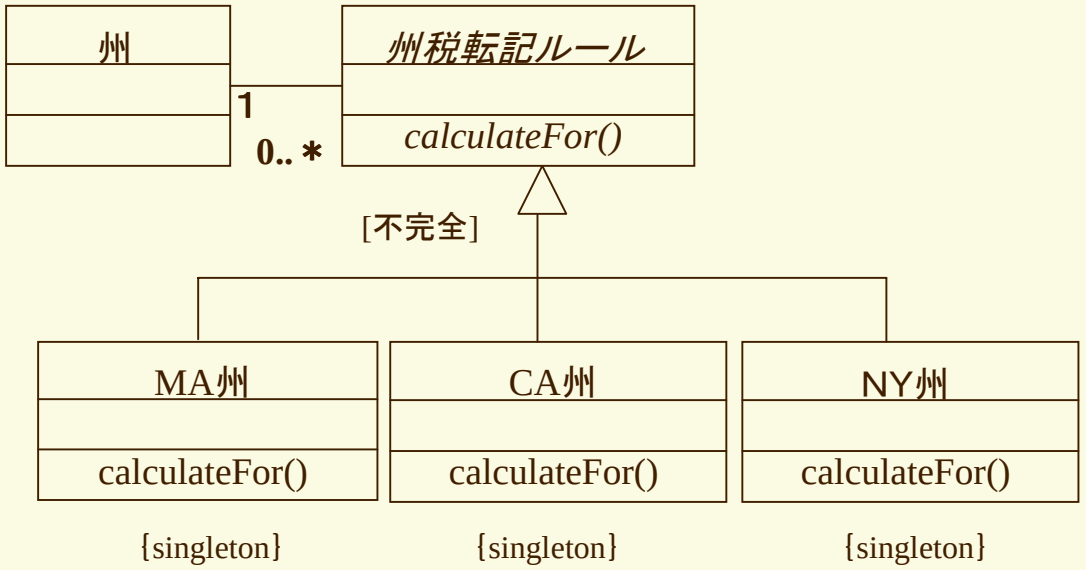


図６．９

問題点：このサブクラスは、州税転記ルールのインスタンス毎に振る舞いを変えることができない。（インスタンスのcalculatevalueを変更できない。）

6．6．2　Strategyパターンを用いた実装

Singletonパターンを用いた実装を分離したメソッドオブジェクトをサブタイプ化したパターン。

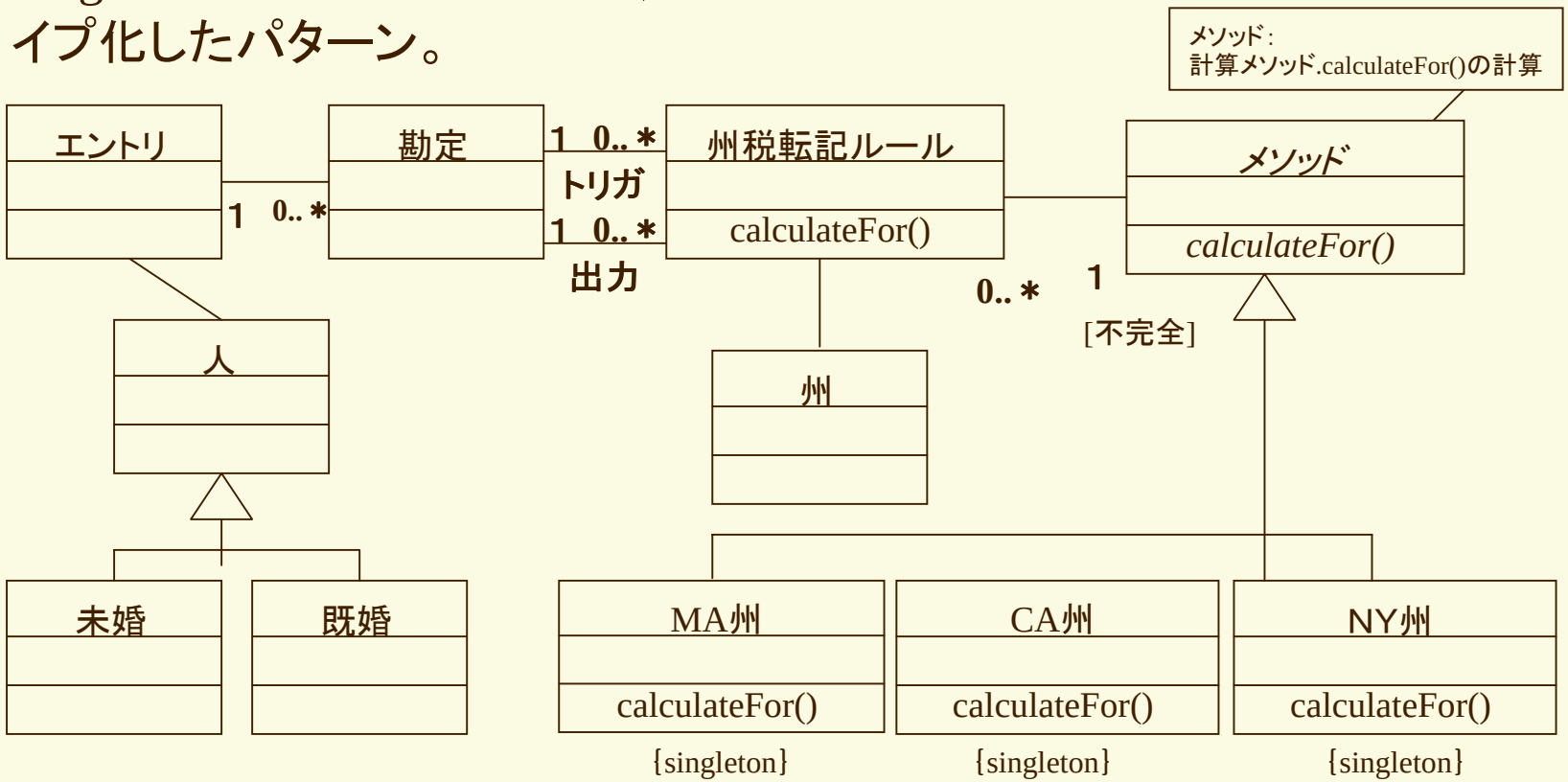


図6．10の変形版（問い合わせの発生）

6．6．3　内部Case文を使った実装

ポリモーフィックなメソッドを1つ処理するためだけにサブクラス化せず、転記ルールにCase文を使って非公開の操作を行う。

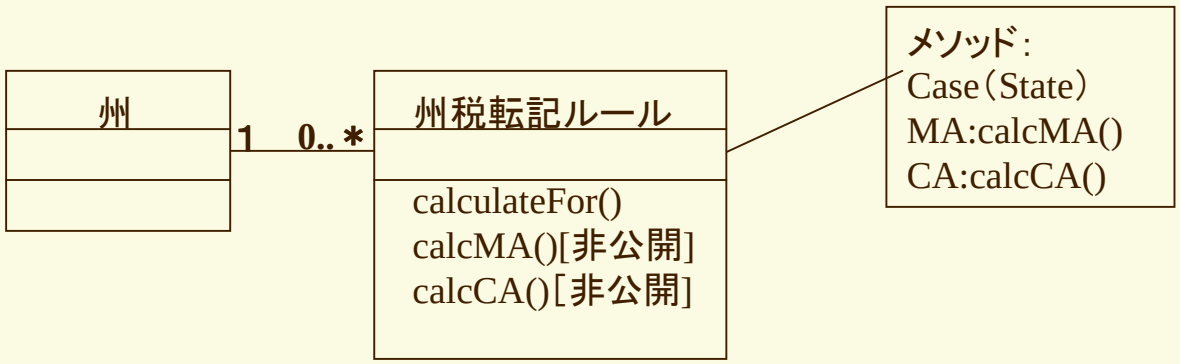


図6．12

６．６．４　パラメタライズドメソッドを用いた．．

パラメタライズメソッド方式は、転記ルール中の１つのメソッドで、転記ルールや関連クラスの属性に対する条件を使って、転記ルールにさまざまな振る舞いをさせる事である。

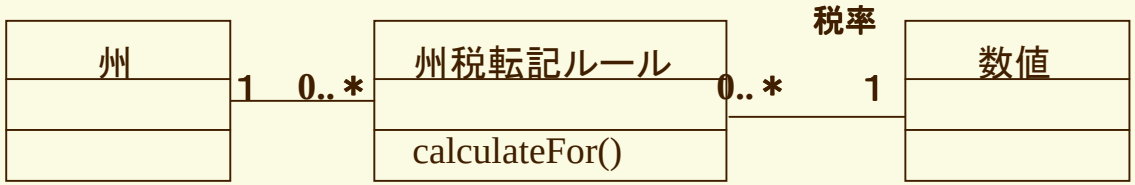


図６．１３

この方略は、計算のあらゆるバリエーションが少数のパラメタを変更することで記述し尽くせる場合に有効である

６．６．５　インタプリタを用いた実装

算術演算子、カッコ、簡単な関数からなる式を使うようなメソッドを実装する場合、interpreterを用いるのが適している。

６．６．６　実装方式の選定

パラメタライズドメソッド方式と他の方式との組み合わせが有効

・メソッドのバリエーションが少ない場合
　　－singleton方式、内部Case文方式
・メソッドが多くなる場合
　　－strategy方式
・メソッドが計算式などの簡単な言語で記述できる場合
　　－interpreter方式