Java基礎演習
(オブジェクト指向編)

オブジェクト指向の概念

オブジェクト指向の概念

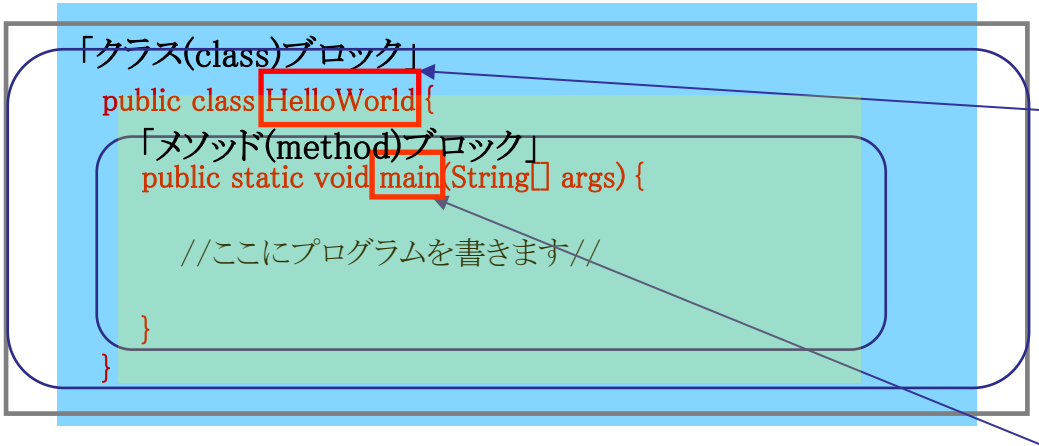
Javaは、
「一度（プログラムを）書けば、どこでも実行できる」
「Write Once、Run Anywhere（WORA）」
という思想をもっています。

また、Javaの強みとして、
・オブジェクト指向の世界を広げる言語
・新しいネットワーク・コンピューティング時代を担う言語
と言われます。

オブジェクト指向の概念

これまでに学習した「基本構文」とは、
プログラムを書くための”基本的な書き方、作法”でした。

ここまで以下のようにクラス(class)ブロックとメソッド(method)ブロックがあることを学び、
データ型、条件文、ループ文のような書き方、作法に触れてきました。



「クラス(class)ブロック」

ここでクラス名を宣言します。

必ずファイル名と同じ名前を使います。

○○○.javaの場合：

かならずpublic class ○○○とします。

「メソッド(method)ブロック」

オブジェクト指向の概念

Java基礎構文編では、どのように動けばよいかを考えながら、メソッドの先頭から順番にプログラミングをおこないました。

```
public class HelloWorld {
    public static void main(String[] args) {
        //ここにプログラムを書きます
    }
}
```

オブジェクト指向の概念

このような先頭から順番にプログラミングしたり、処理する仕組みは1960年代から存在しており、これは「手続き型プログラミング」と呼ばれる方法です。

「手続き型プログラミング」では、どのように動けばよいかを考えて、これを先頭から順番に命令してゆきます。

「手続き型プログラミング」に向いている言語は、

BASIC言語
C言語
COBOL
FORTRAN
Perl
PL/I
などがあります。

主にホストコンピュータ、汎用コンピュータ、汎用機と呼ばれる大型コンピュータで使用されています。

オブジェクト指向の概念

現在、「手続き型プログラミング」は、WindowsやMac、Linux等において、コマンドプロンプトやターミナルで動作するアプリケーションを作成する場合には、時折、この方法が採用されます。

例えばバッチプログラムやマクロプログラム等は、「手続き型プログラミング」で作成されます。

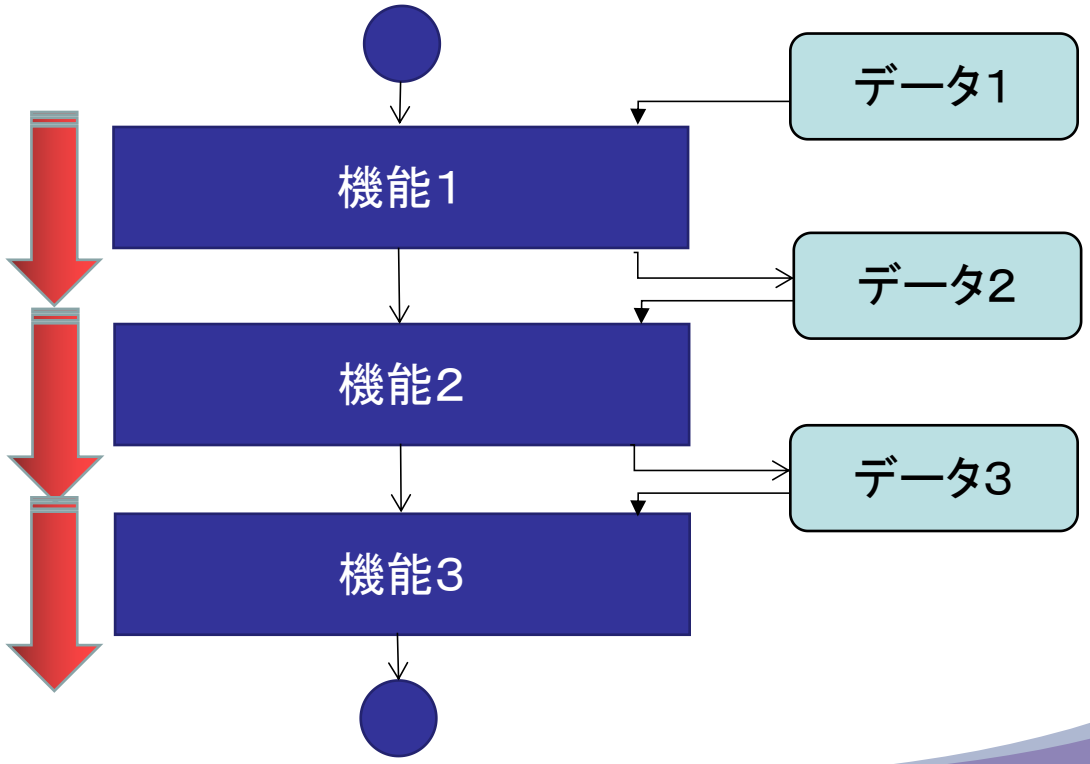
一方、Webサイトや業務アプリケーションを作成する場合、「手続き型プログラミング」は、あまり向いていないと言われます。

Java言語	コマンドプロンプト、 ターミナル等	Webサイト、 業務アプリケーション等
手続き型プログラミング	○ （向いている）	△ （向いていない）

オブジェクト指向の概念

これは、「手続き型プログラミング」が一連の定義された機能の塊を呼び出し、これを連鎖的に処理してゆく、という思想をもっている為です。

「手続き型プログラミング」は、機能に対して必要なデータが連続的に受渡しされることで、全体のアプリケーションを構成します。
受渡しに失敗すると、処理は止まります。

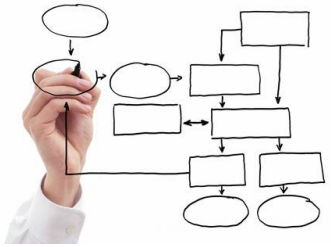


オブジェクト指向の概念

これに対して、「オブジェクト指向プログラミング」は「オブジェクト」という塊でプログラムを考えてゆきます。

「オブジェクト」とは、「ひと」や「もの」や「概念」などを表現する仕組みです。
「ひと」や「もの」や「概念」がどのようなデータや機能を持っているかを考え、まずこれをプログラムします。

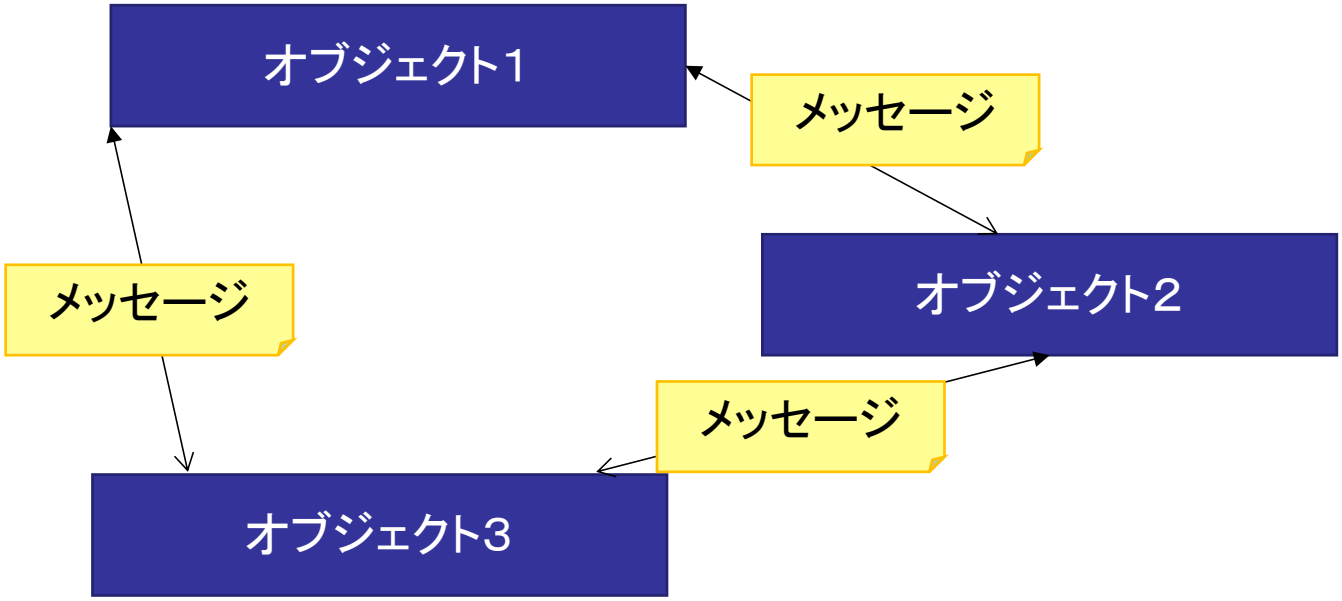
これらは抽象的なものでも構いませんし、具体的なものでも構いません。
また、プログラムを使って、あとから具体化したり、足りない仕組みを補完することも可能です。
（※ここは具体的なオブジェクト指向の考え方として後ほど解説してゆきます。）



オブジェクト指向の概念

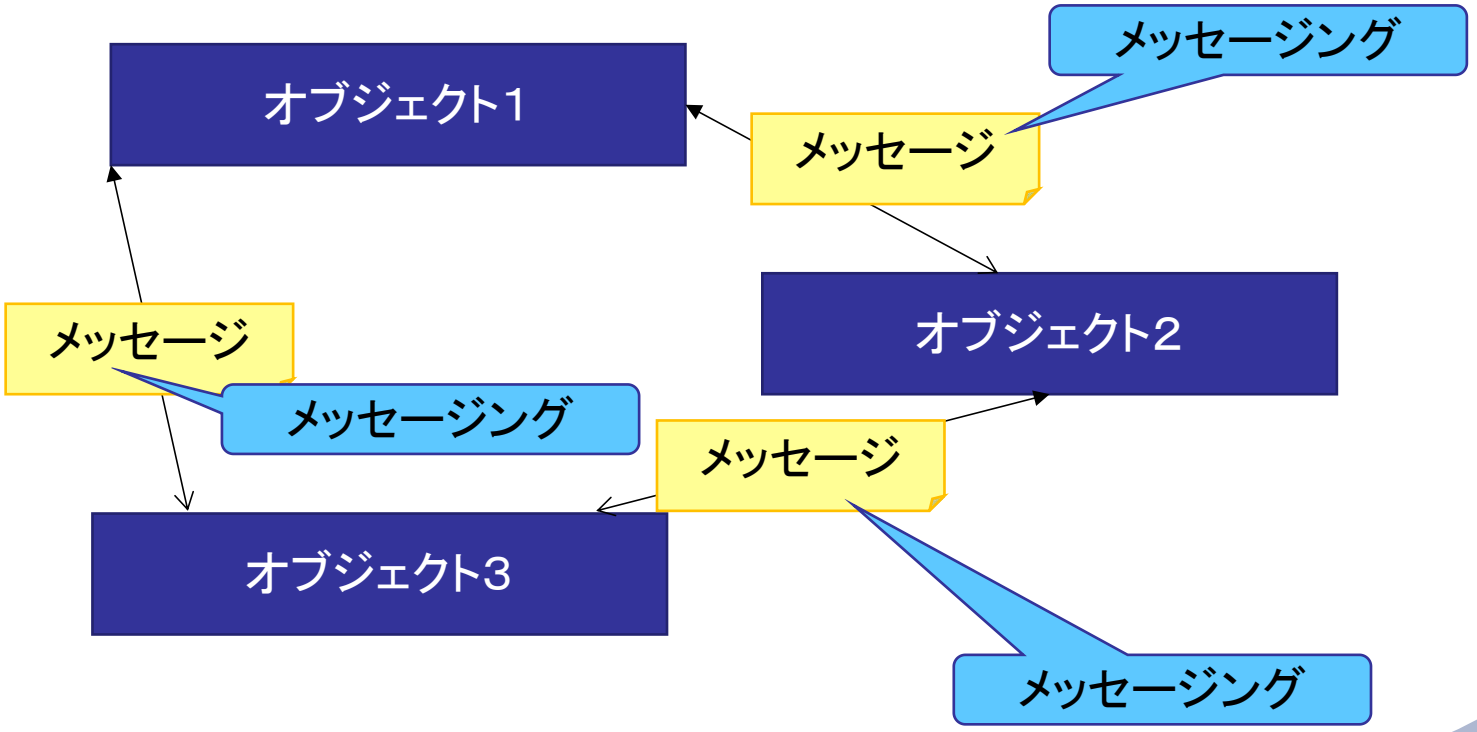
「オブジェクト指向プログラミング」は、「オブジェクト」同士がメッセージを受渡しながら、柔軟にやりとりしてゆく思想をもっています。

そもそも「オブジェクト指向プログラミング」は、1967年にアラン・ケイによって提唱された概念であり、”メッセージ指向プログラミング”と言い換えることもできる、とされましたが、混乱を招くことから、いまは「オブジェクト指向プログラミング」で統一されています。



オブジェクト指向の概念

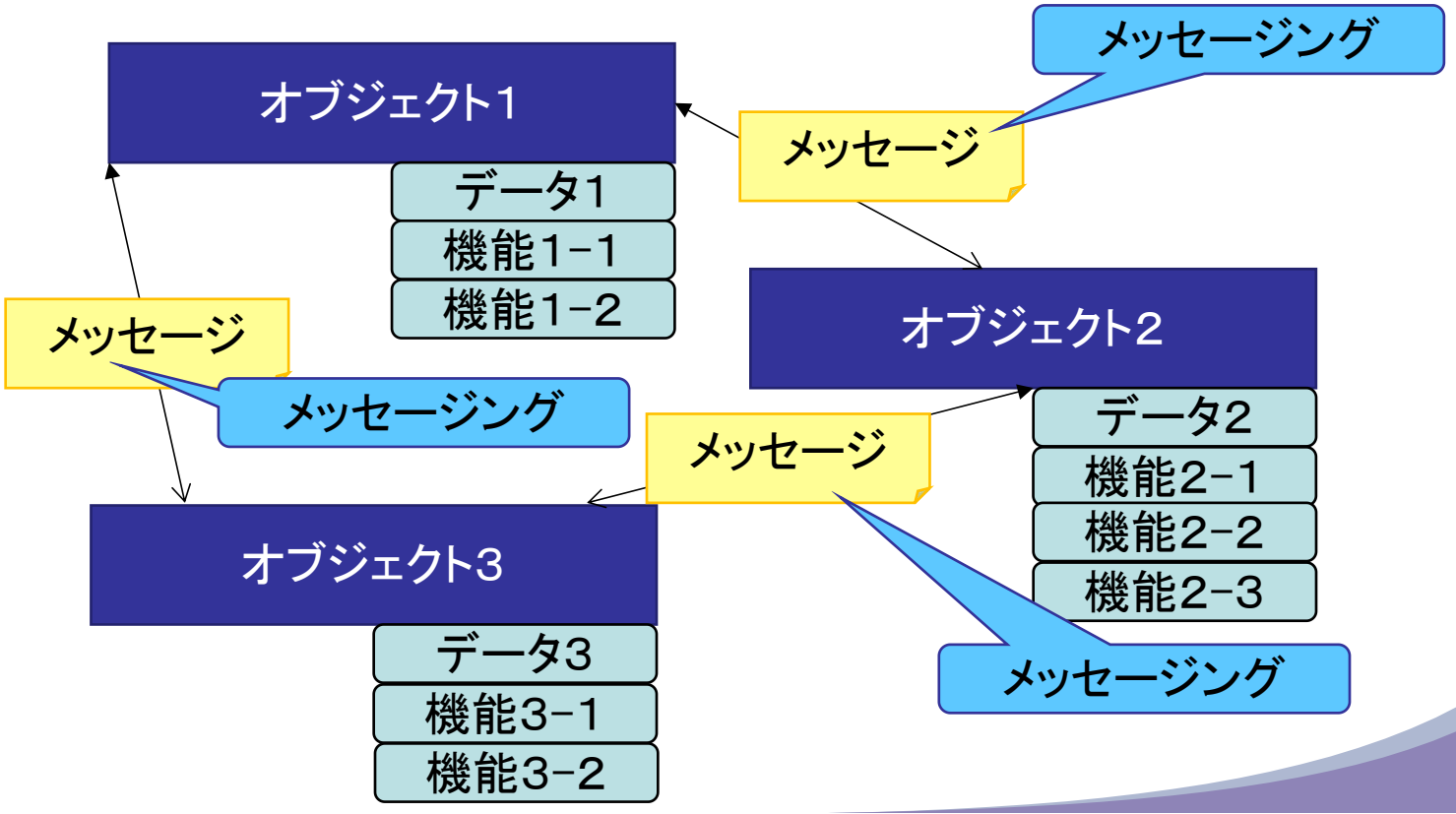
各オブジェクトがメッセージでやり取りする仕組みを、アラン・ケイは”メッセージング”と呼んでいます。



オブジェクト指向の概念

それぞれの「オブジェクト」には「データ」と「機能」を作成します。
（Javaでは、この「データ」と「機能」をそれぞれ、「フィールド」と「メソッド」とします。）

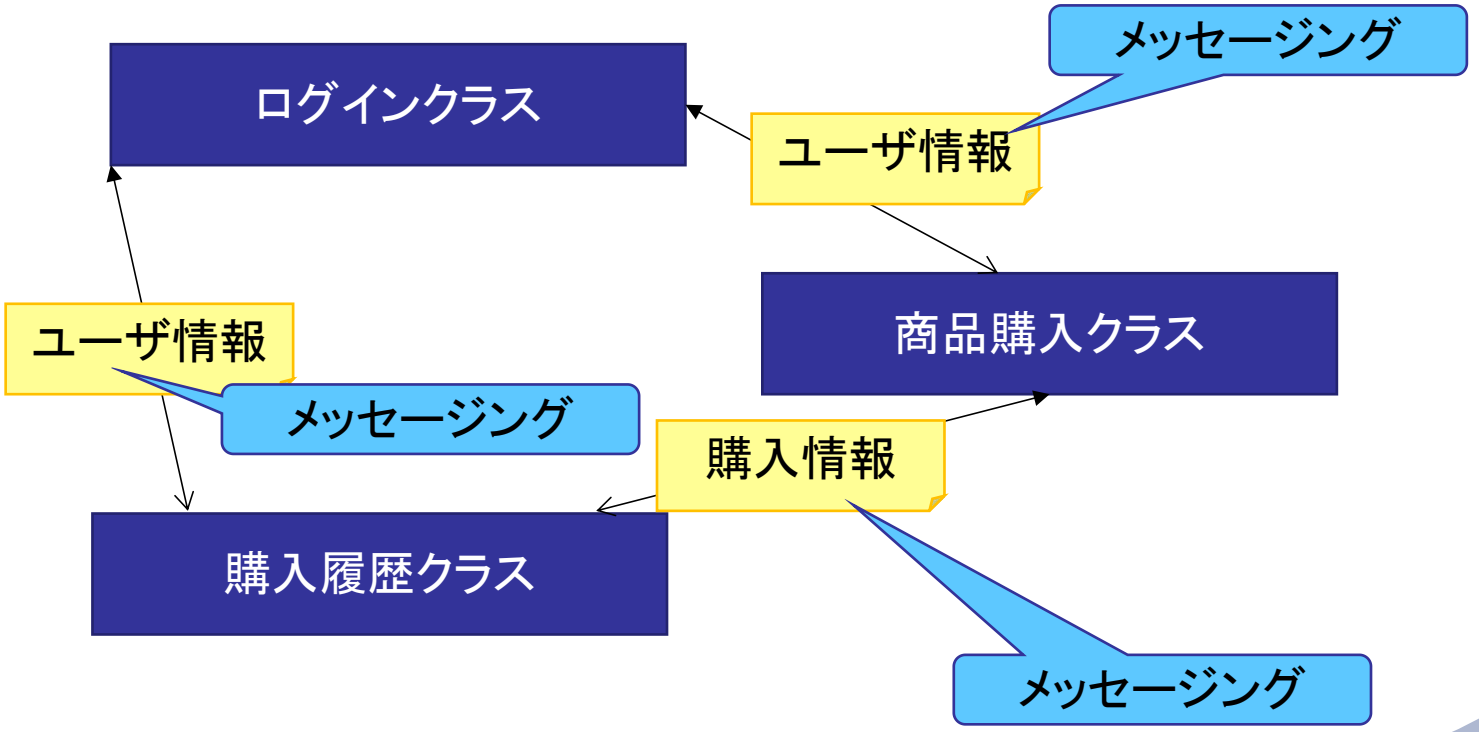
各「オブジェクト」は、メッセージングによって受け取ったメッセージをもとに、
必要な機能を実行したり、受渡し先を変更しながら処理を継続します。



オブジェクト指向の概念

以下のような画面で構成されたWebサイトがあったとしましょう。
Javaにおいて、各「オブジェクト」は「クラス」として作成します。

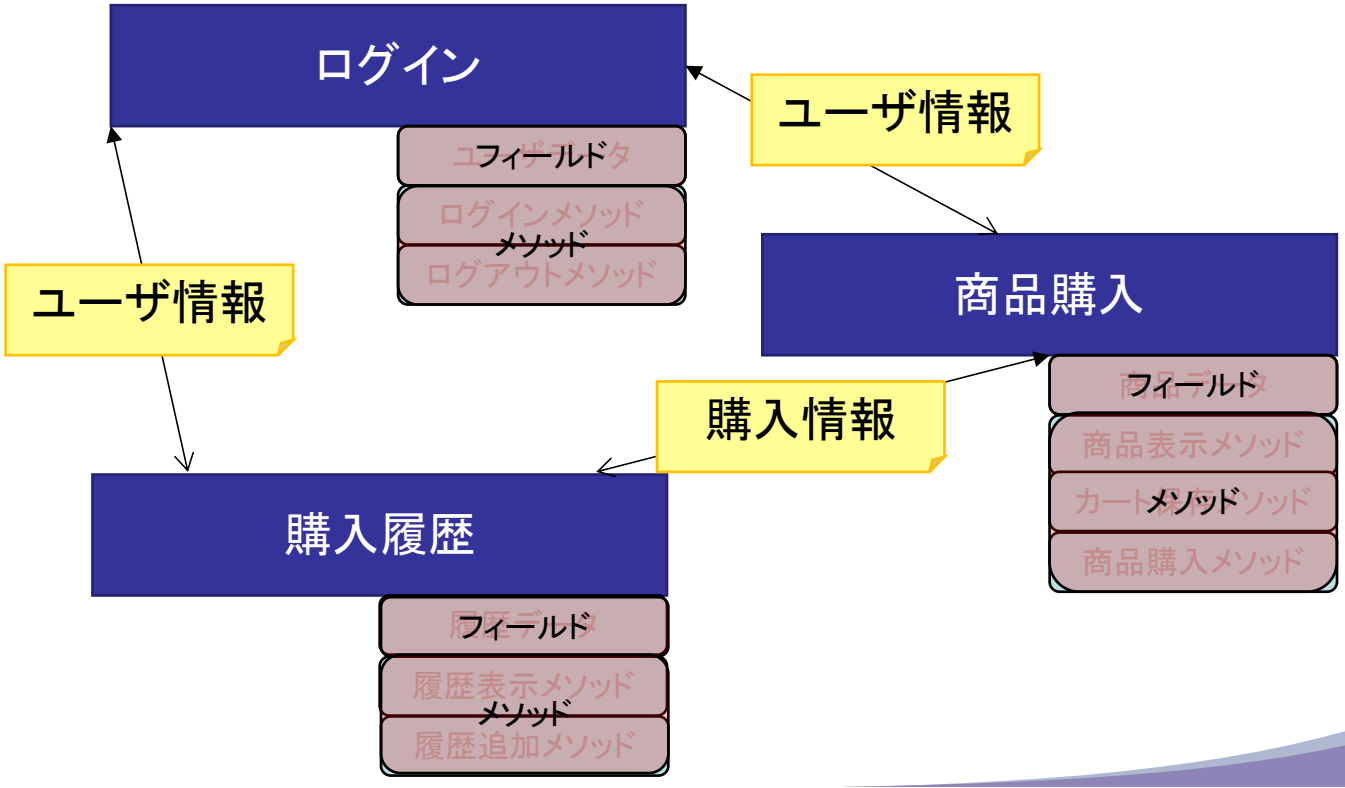
また、ログインクラス、商品購入クラス、購入履歴クラスは、「ユーザ情報」や「購入情報」を受渡しながら処理を継続します。



オブジェクト指向の概念

また、Javaでは「データ」と「機能」を「フィールド」と「メソッド」として作成してゆきます。

メッセージングによって情報を受け取ったクラスは、必要なメソッドを実行したり、受渡し先を切替えながら処理を継続します。
「オブジェクト指向プログラミング」では、このようにWebサイト全体を構成してゆきます。



オブジェクト指向の概念

「オブジェクト指向プログラミング」は、このように柔軟な処理を実現します。

「オブジェクト指向プログラミング」はWebサイトや業務アプリケーションに向いていると言われている理由は、こうした柔軟なプログラムや処理が可能である為です。

Java言語	コマンドプロンプト、 ターミナル等	Webサイト、 業務アプリケーション等
手続き型 プログラミング	○ （向いている）	△ （やや厳しい）
オブジェクト指向 プログラミング	○ （向いている）	○ （向いている）

オブジェクト指向の概念

「オブジェクト指向プログラミング」に向いている言語は、
Java言語
C＋＋
C#
Ruby
PHP
Python
R言語
などがあります。

最近人気があるプログラム言語の多くは、「オブジェクト指向」を採用しています。

※なお、すべてのアプリケーションを「オブジェクト指向プログラミング」で実現すべきではありません。また、自身の得意／不得意によって方法を変えるべきではありません。
それぞれの特性や用途に応じて、プログラミングの言語や方法は決定してゆきましょう。

手続き型プログラミング
と
オブジェクト指向プログラミング

手続き型プログラミングとオブジェクト指向プログラミング

ここからは、「手続き型プログラミング」と「オブジェクト指向プログラミング」を比較しながら、それぞれの考え方や手順の違いについて整理してみます。

```
public class HelloWorld {
    public static void main(String[] args) {
        //ここにプログラムを書きます
    }
}
```

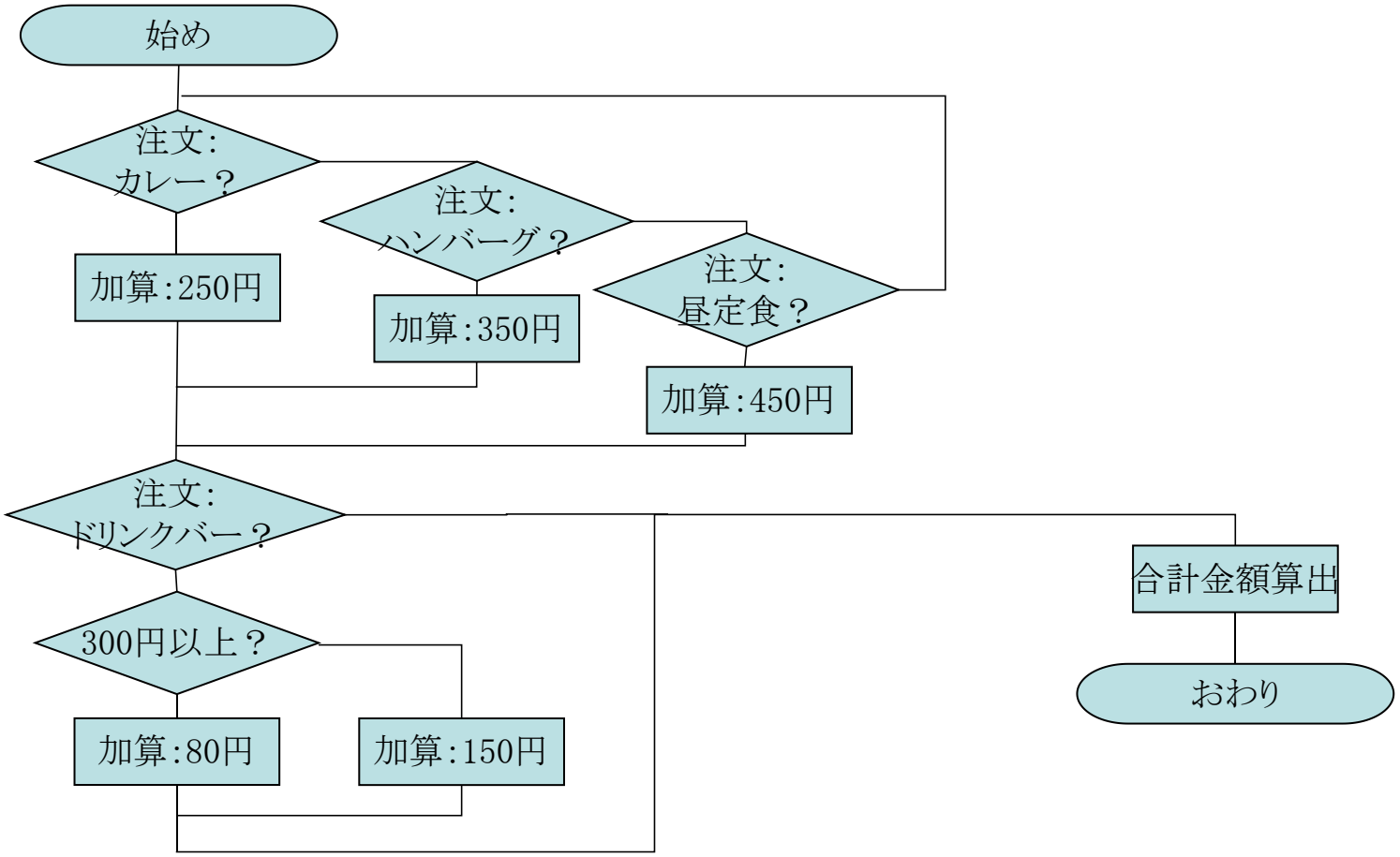
手続き型プログラミングとオブジェクト指向プログラミング

まず、「手続き型プログラミング」で考えてみます。
例題として、以下の問題をフローチャートにしてみましょう。

- ファミレスで食事をする。カレーが250円、ハンバーグが350円、昼定食が450円、ドリンクバーが150円とする。
- カレー、ハンバーグ、昼定食の中から一つを選び、またドリンクバーは選ぶかどうか分からない。
- ドリンクバーは300円以上のお食事をしたときには80円でよいとする。
- 何を注文したかを判断して食事の合計金額を算出するフローチャートを作ってみよう。

手続き型プログラミングとオブジェクト指向プログラミング

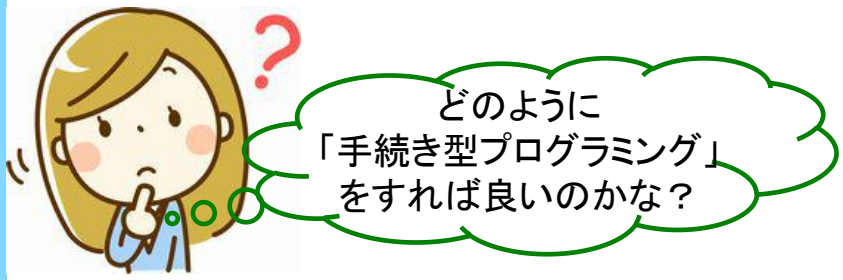
先程の例題について、フローチャートは以下の通りです。



手続き型プログラミングとオブジェクト指向プログラミング

では、これをもとに、どのように「手続き型プログラミング」をすれば良いか考えてゆきます。

```
public class HelloWorld {
    public static void main(String[] args) {
        はじめ
        注文はカレー？
        カレーであれば、250円を加算
        カレーでなければ、
        注文はハンバーグ？
        ハンバーグであれば、350円加算
        ハンバーグでなければ、
        注文は昼定食？
        昼定食であれば、450円加算
        昼定食でなければ、はじめに戻る
                    ・
                    ・
                    ・
    }
}
```



手続き型プログラミングとオブジェクト指向プログラミング

「手続き型プログラミング」では、手続きの流れを先頭から順番にプログラミングし、データやそれらの作用（機能）をまとめます。

```
public class HelloWorld {
    public static void main(String[] args) {
        while(注文==未決定){
          if(注文==カレー){
            金額=金額+250;
            注文=決定;
          } else if(注文==ハンバーグ){
            金額=金額+350;
            注文=決定;
          } else if(注文==昼定食){
            金額=金額+450;
            注文=決定;
          }
        }
                    .
                    .
                    .
    }
}
```

先頭から順番に
プログラミング

手続き型プログラミングとオブジェクト指向プログラミング

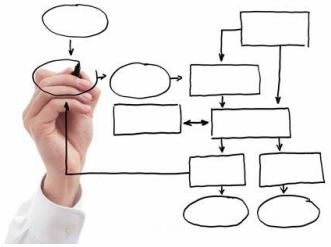
では、「オブジェクト指向プログラミング」の考え方で、どのように動けばよいかを整理してみましょう。

- ファミレスで食事をする。カレーが250円、ハンバーグが350円、昼定食が450円、ドリンクバーが150円とする。
- カレー、ハンバーグ、昼定食の中から一つを選び、またドリンクバーは選ぶかどうか分からない。
- ドリンクバーは300円以上のお食事をしたときには80円でよいとする。
- 何を注文したかを判断して食事の合計金額を算出するフローチャートを作ってみよう。

手続き型プログラミングとオブジェクト指向プログラミング

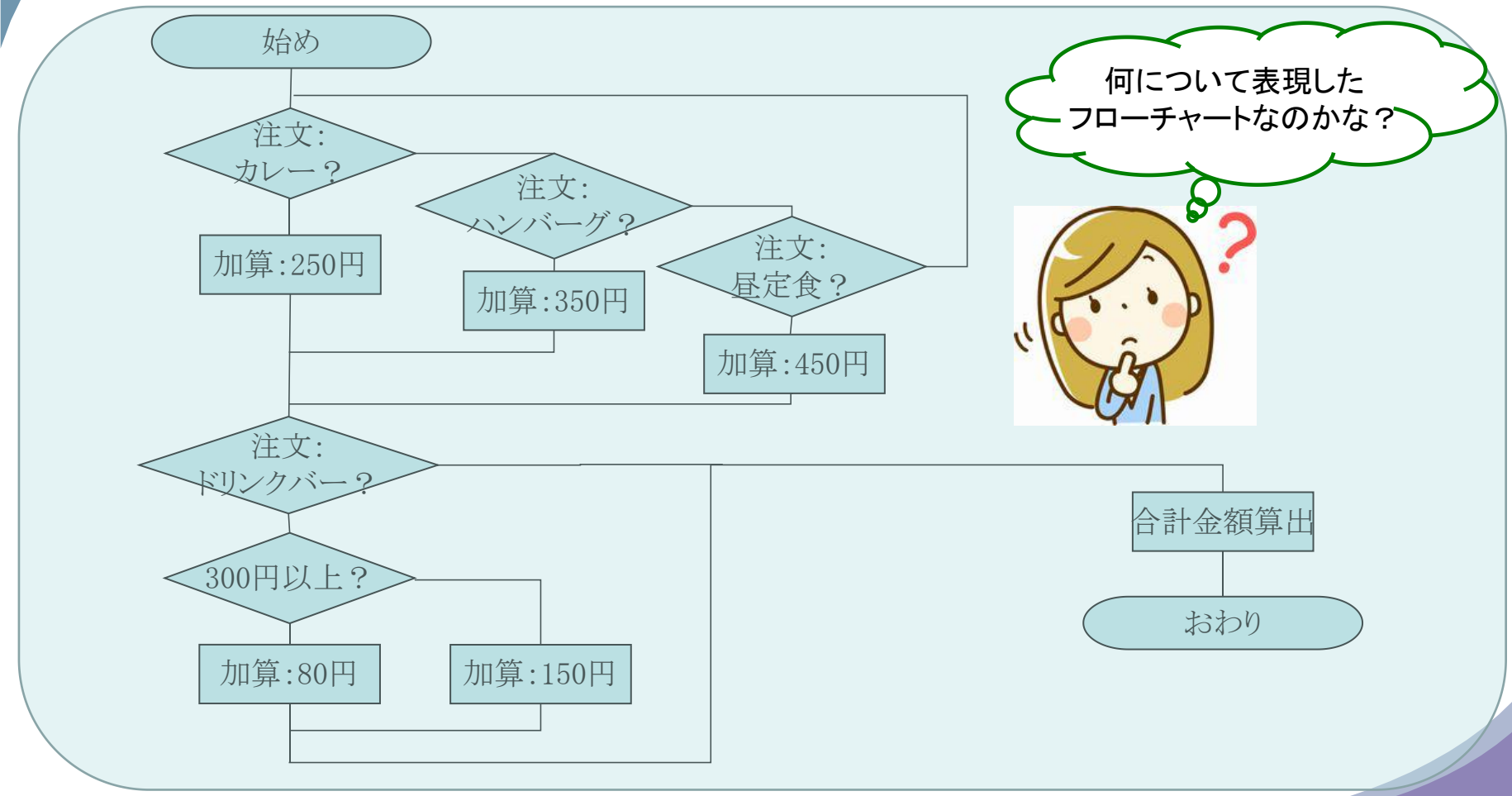
「オブジェクト指向プログラミング」は、「オブジェクト」という塊でプログラムを考えます。

「オブジェクト」とは、「ひと」や「もの」や「概念」などを表現する仕組みです。



手続き型プログラミングとオブジェクト指向プログラミング

オブジェクト指向は「ひと」「もの」「概念」などを表現する仕組み、とありました。
では、以下のフローチャートは何について表現しているかを考えてみます。



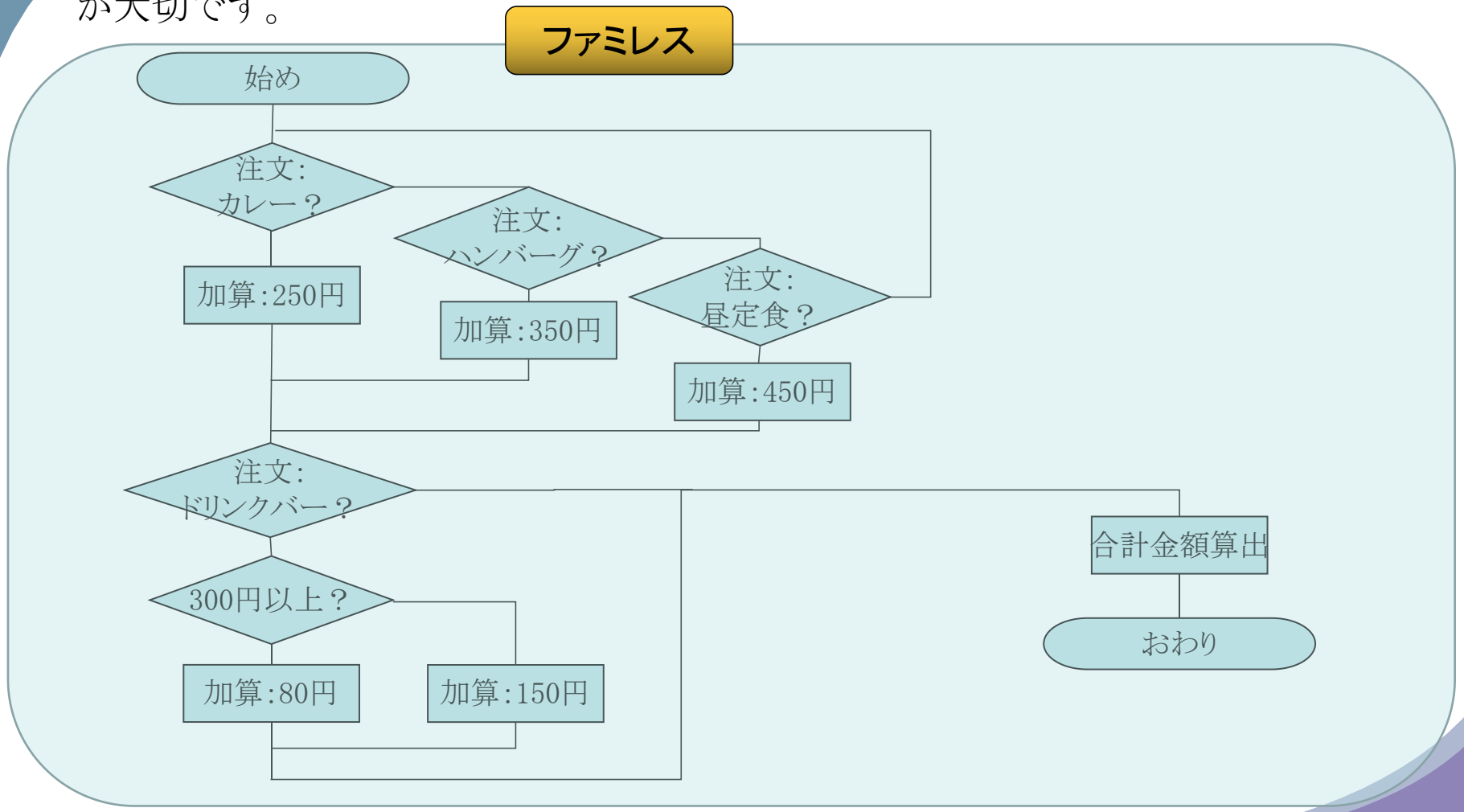
手続き型プログラミングとオブジェクト指向プログラミング

今回のフローチャートは「ファミレス」について表現している、と解釈できます。

- ファミレスで食事をする。カレーが250円、ハンバーグが350円、昼定食が450円、ドリンクバーが150円とする。
- カレー、ハンバーグ、昼定食の中から一つを選び、またドリンクバーは選ぶかどうか分からない。
- ドリンクバーは300円以上のお食事をしたときには80円でよいとする。
- 何を注文したかを判断して食事の合計金額を算出するフローチャートを作ってみよう。

手続き型プログラミングとオブジェクト指向プログラミング

「オブジェクト指向プログラミング」では、まず何について表現されているのかを捉えることが大切です。



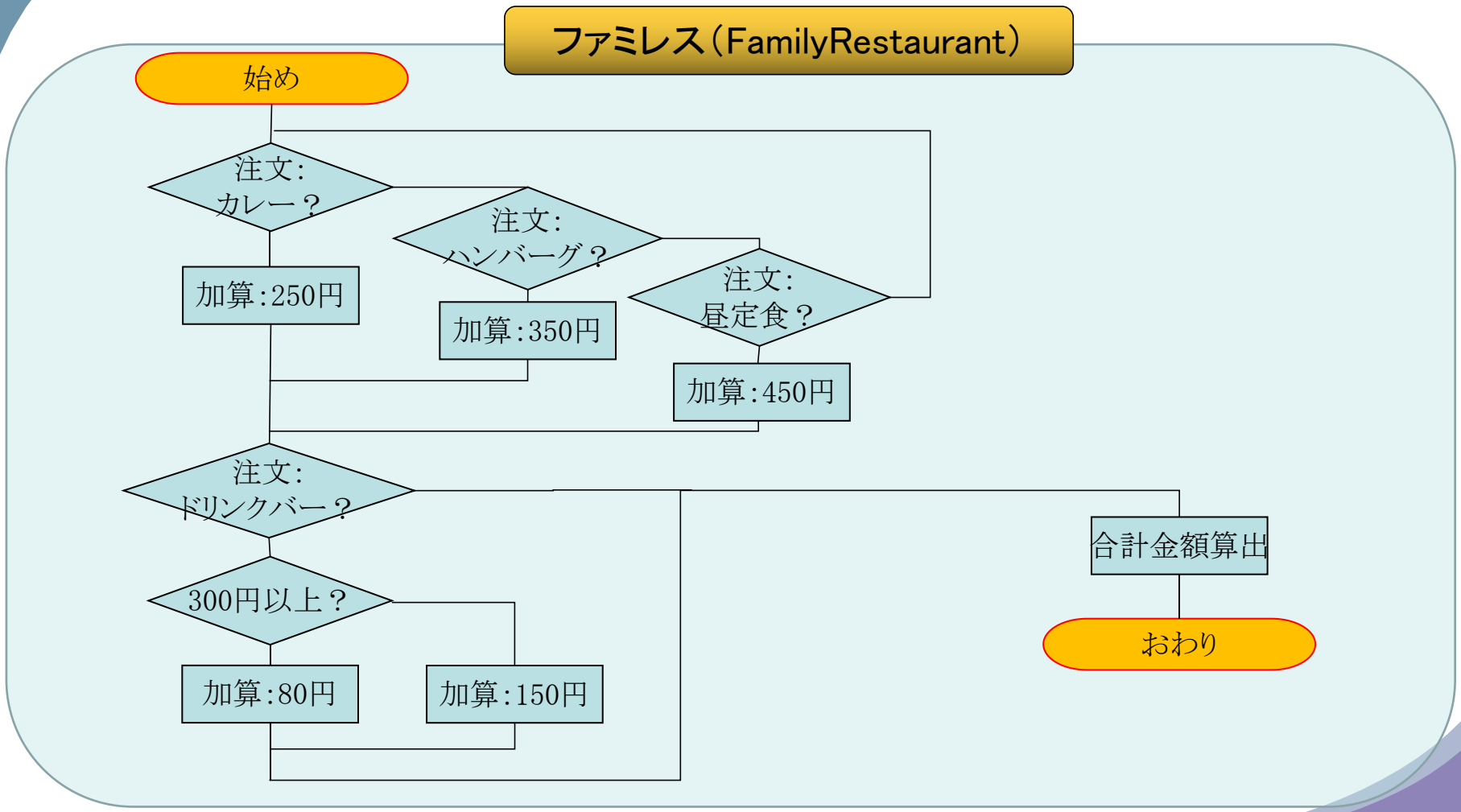
手続き型プログラミングとオブジェクト指向プログラミング

フローチャート全体を包括している「ファミレス（FamilyRestaurant）」は、ひとつのオブジェクトだと言い換えることができます。そこで、以下のように「FamilyRestaurant」クラスを作成してみます。

```
public class FamilyRestaurant {




}
```

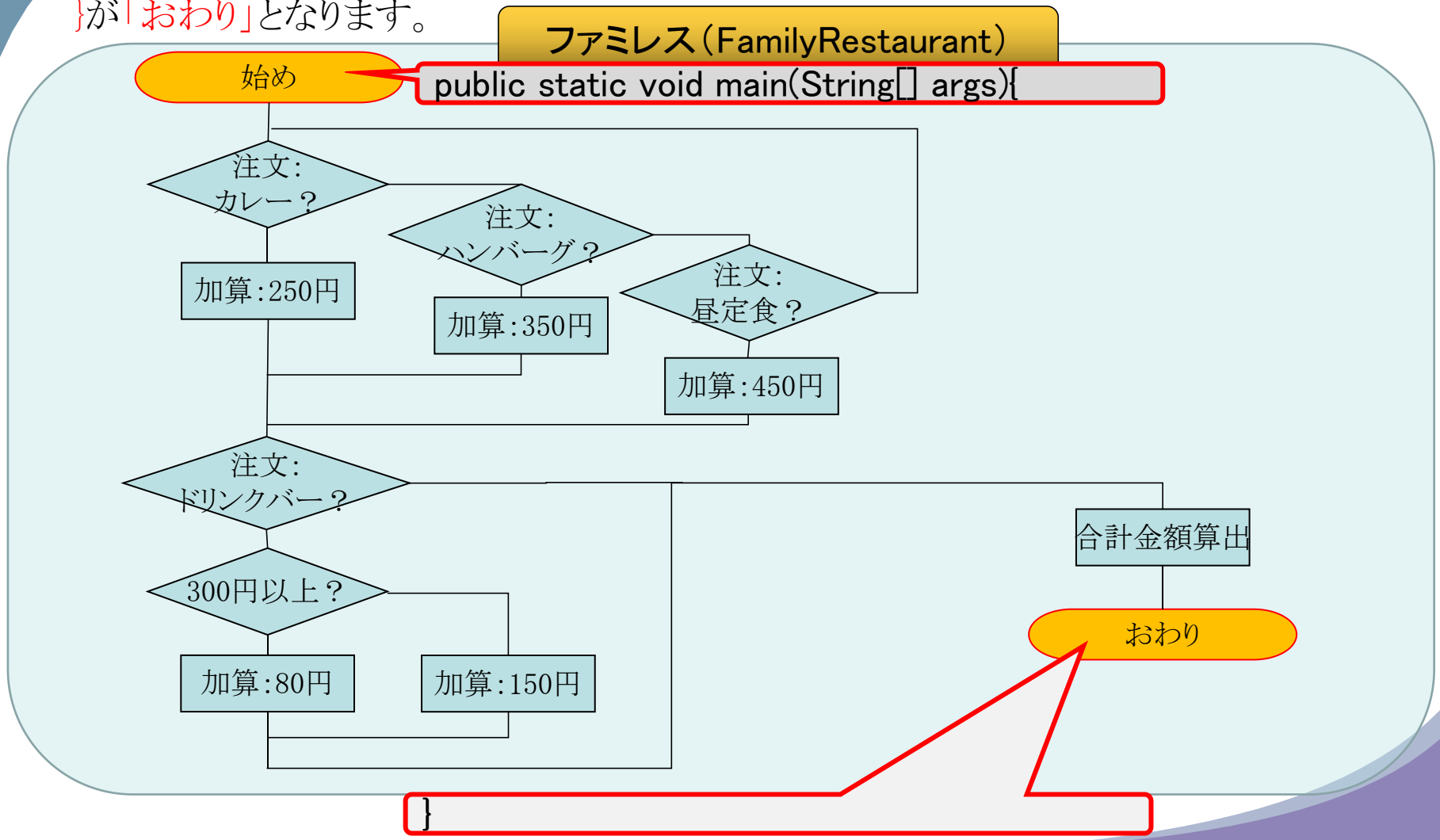
手続き型プログラミングとオブジェクト指向プログラミング

つぎに、「始め」と「おわり」をFamilyRestaurantクラスにプログラミングします。



手続き型プログラミングとオブジェクト指向プログラミング

Javaでは、以下のようにpublic static void main(String[] args){が「始め」、これに紐づく }が「おわり」となります。



手続き型プログラミングとオブジェクト指向プログラミング

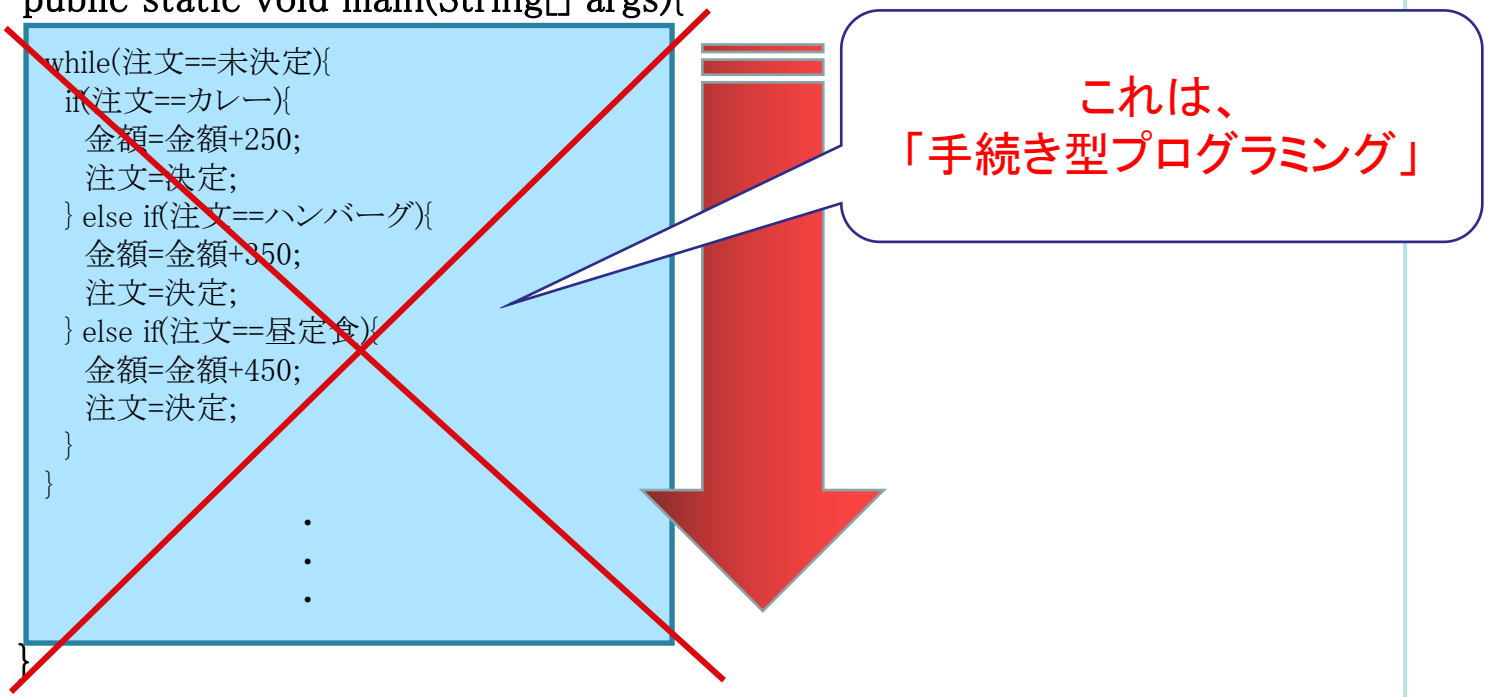
これをFamilyRestaurantクラスの中に記述します。
具体的には、FamilyRestaurantクラスの中にmainメソッドを以下のように追記します。

```
public class FamilyRestaurant {
    public static void main(String[] args){









    }
}
```

手続き型プログラミングとオブジェクト指向プログラミング

ここからが「オブジェクト指向プログラミング」の本題です。
以下の記述では「手続き型プログラミング」となってしまいます。
これを「オブジェクト指向プログラミング」で記述しなければなりません。

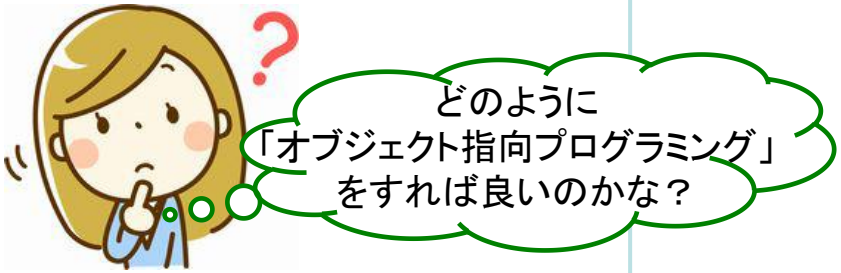
```
public class FamilyRestaurant {
    public static void main(String[] args){
        while(注文==未決定){
          if(注文==カレー){
            金額=金額+250;
            注文=決定;
          } else if(注文==ハンバーグ){
            金額=金額+350;
            注文=決定;
          } else if(注文==昼定食){
            金額=金額+450;
            注文=決定;
          }
        }
                    .
                    .
                    .
    }
}
```



手続き型プログラミングとオブジェクト指向プログラミング

「オブジェクト指向プログラミング」では、考え方や記述方法が変わります。
ただ、「オブジェクト指向プログラミング」では、いきなりmainメソッドの中身を書かなくても構いません。

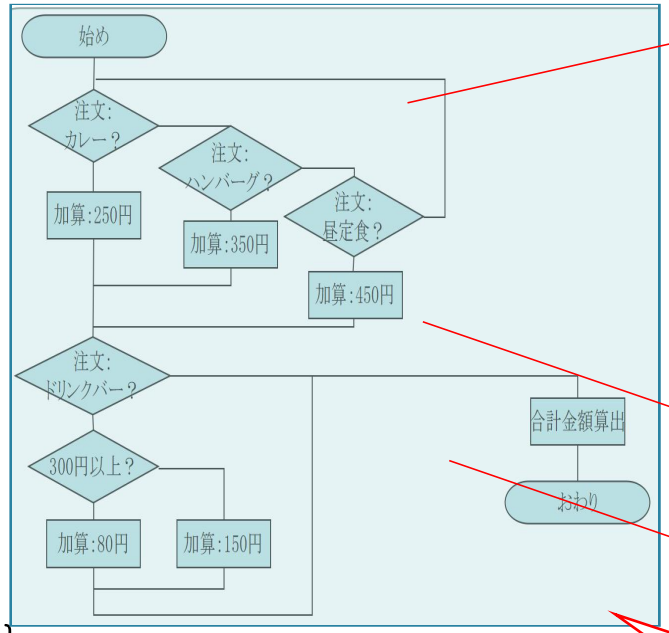
```
public class FamilyRestaurant {
    public static void main(String[] args){
        ?
    }
}
```



手続き型プログラミングとオブジェクト指向プログラミング

「オブジェクト指向プログラミング」では、まずオブジェクト（つまり、「ひと」「もの」「概念」）を抜き出して、これをプログラミングしてゆきます。

```
public class FamilyRestaurant {
    public static void main(String[] args){
    }
}
```



オブジェクトを抜き出して
プログラミング

ひと

プログラミング

もの

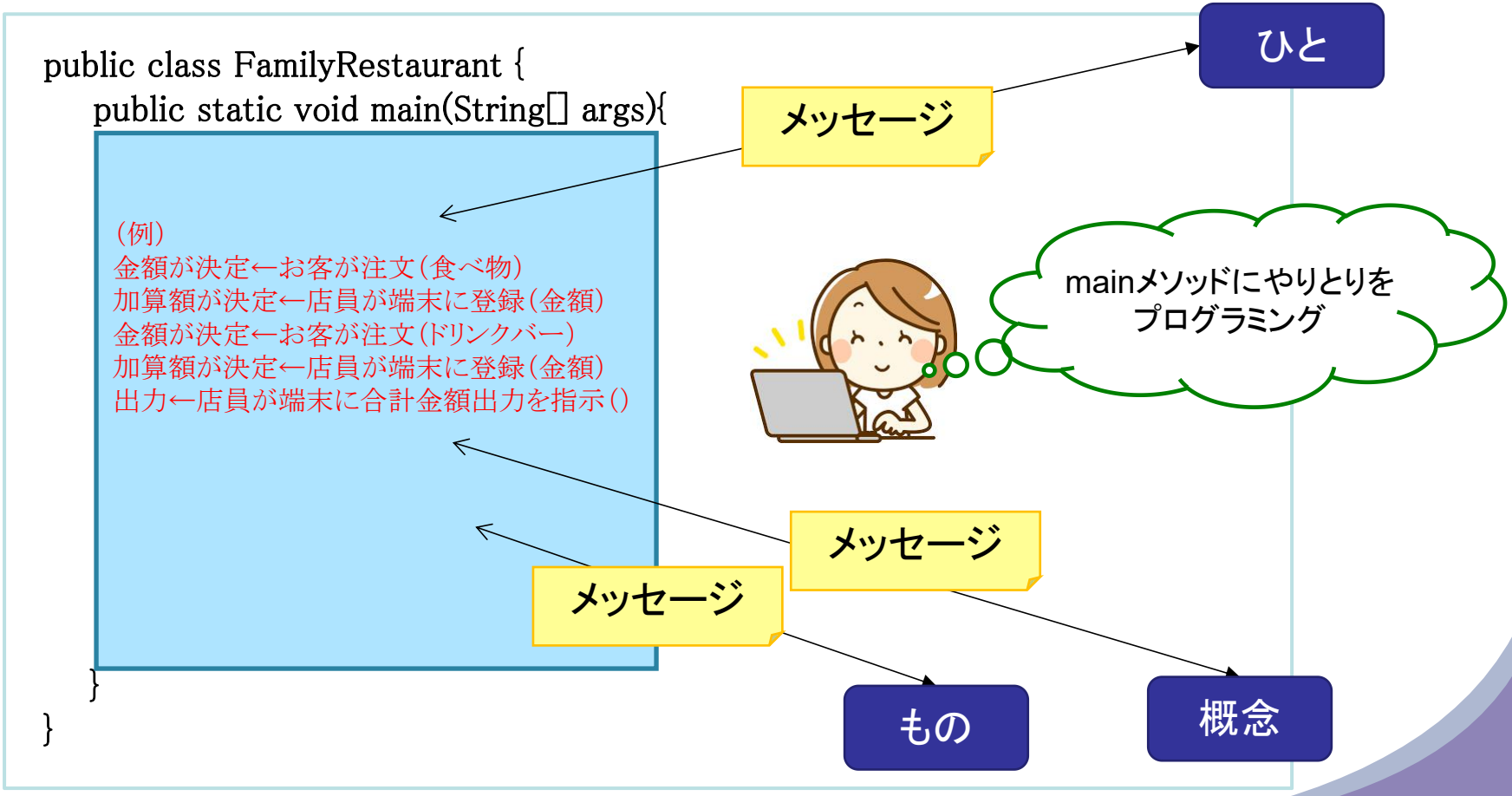
プログラミング

概念

プログラミング

手続き型プログラミングとオブジェクト指向プログラミング

つぎにmainメソッドには、先ほど作成したオブジェクト同士のやり取りや、メッセージのやり取りをプログラムしてゆきます。



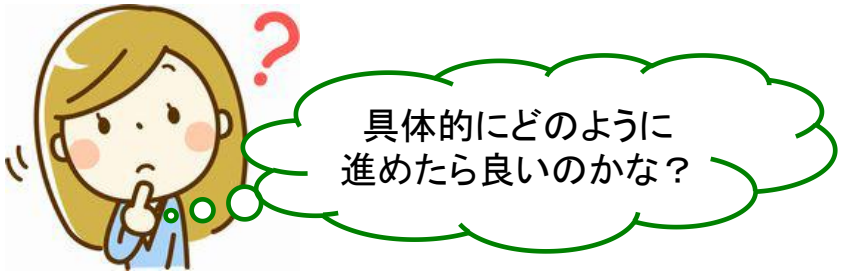
手続き型プログラミングとオブジェクト指向プログラミング

このように「オブジェクト指向プログラミング」では、

・オブジェクト（つまり、「ひと」「もの」「概念」）を抜き出してプログラミング
・オブジェクト同士のやりとりや、メッセージのやり取りをプログラミング

の2点をおこなってゆきます。

つぎに、この2点の進め方についておさらいしながら解説してゆきます。



手続き型プログラミングとオブジェクト指向プログラミング

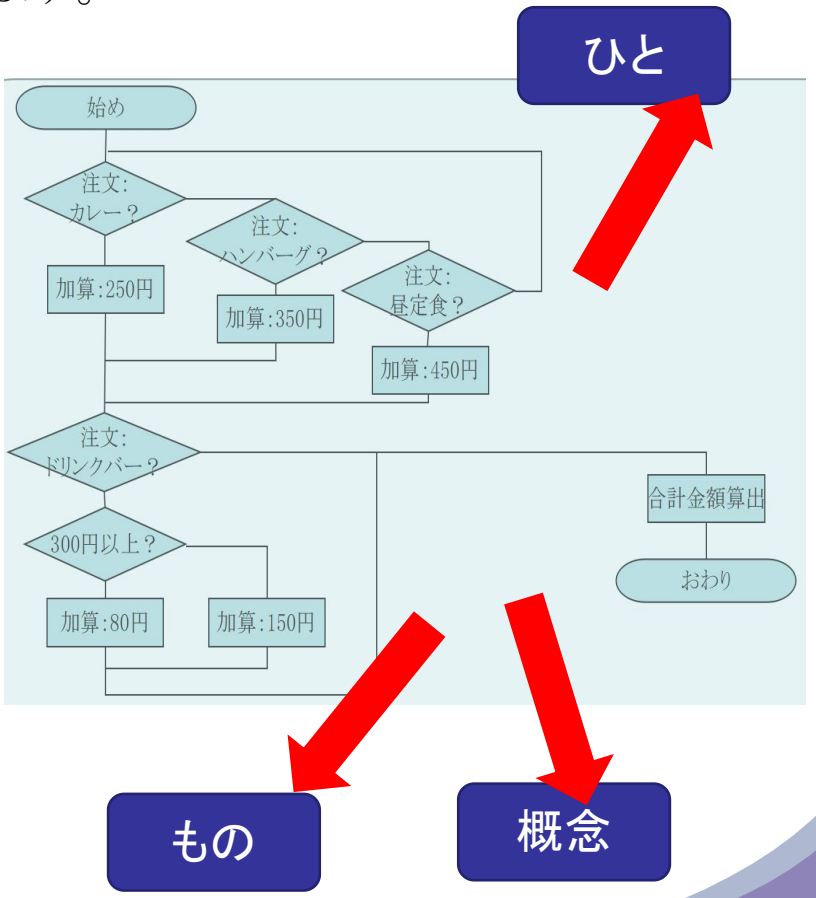
まずは、オブジェクト(つまり、「ひと」「もの」「概念」)を抜き出してプログラミングする流れから解説してゆきます。
先ほどのFamilyRestaurantは、クラス、つまり、オブジェクトと言い換えることができます。

```
public class FamilyRestaurant {
    public static void main(String[] args){
        ?
    }
}
```

手続き型プログラミングとオブジェクト指向プログラミング

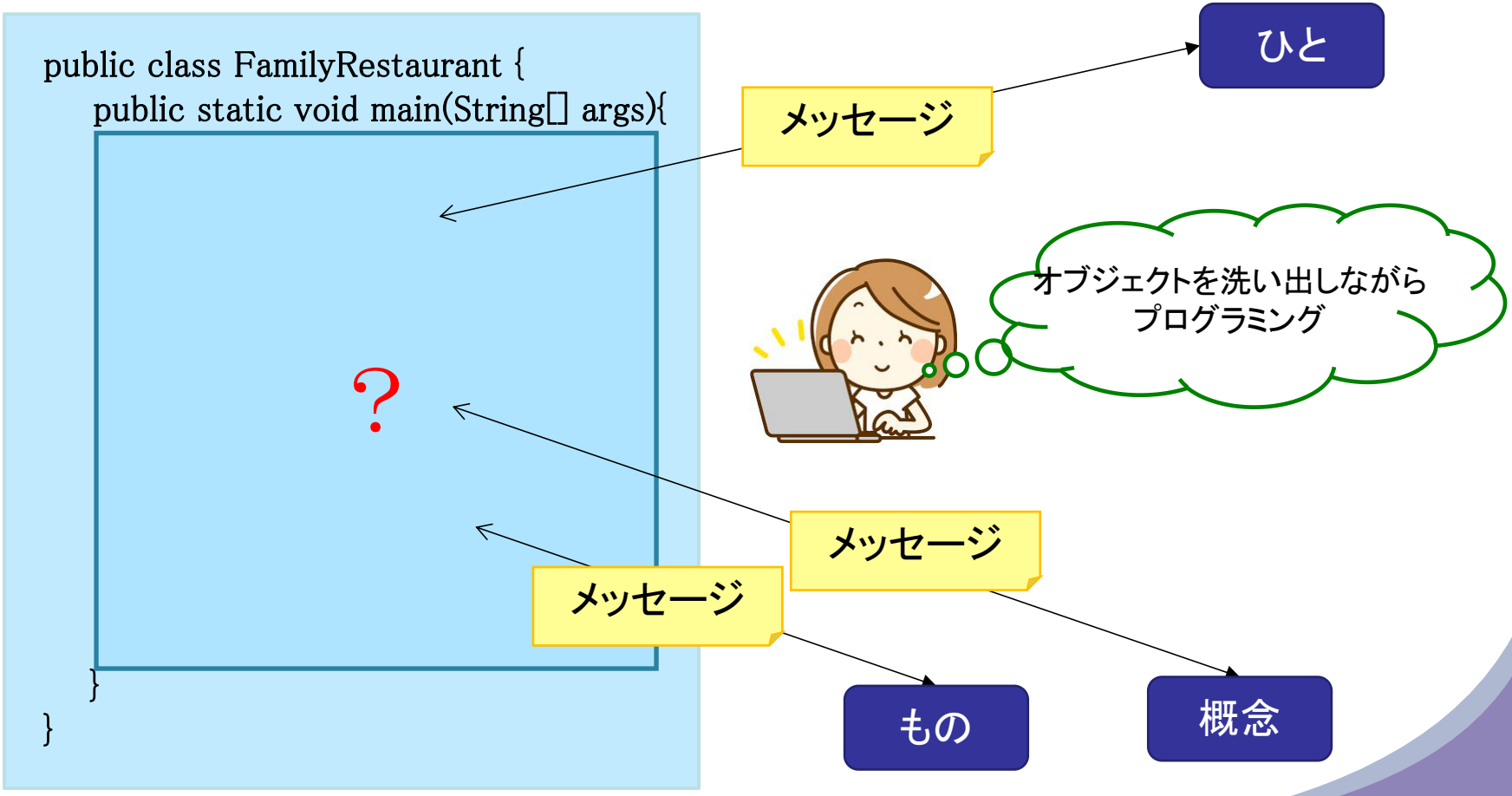
「オブジェクト指向プログラミング」では、フローチャートの先頭から順番にプログラムは記述しません。このフローチャートの処理をおこなっている「ひと」「もの」「概念」は何かを考えて、まずこれを抜き出すことからはじめます。

```
public class FamilyRestaurant {
    public static void main(String[] args){
    }
}
```



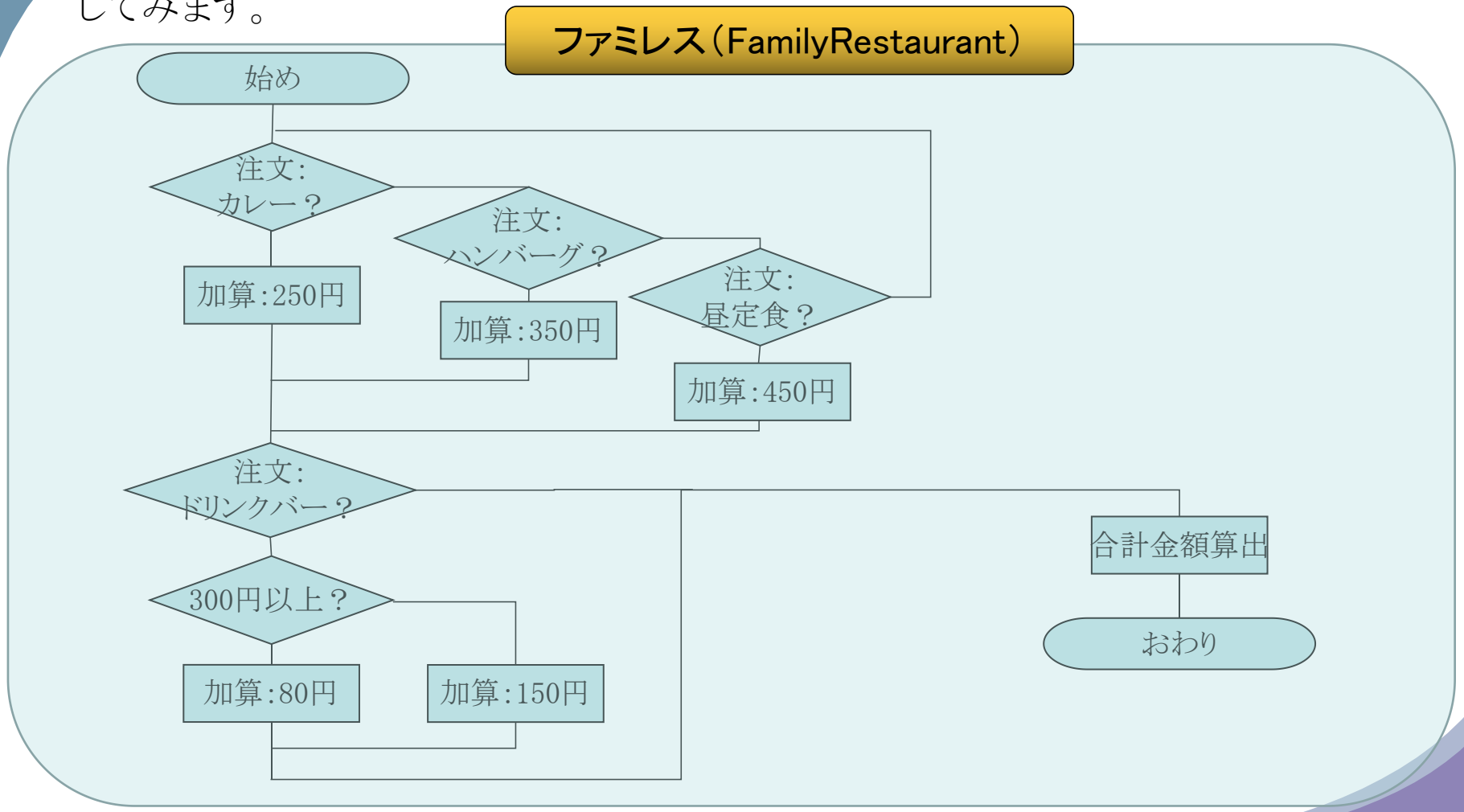
手続き型プログラミングとオブジェクト指向プログラミング

つぎに抜き出した「ひと」「もの」「概念」（つまり、オブジェクト）をプログラミングして、メッセージを使ったやり取りをプログラムしてゆきます。



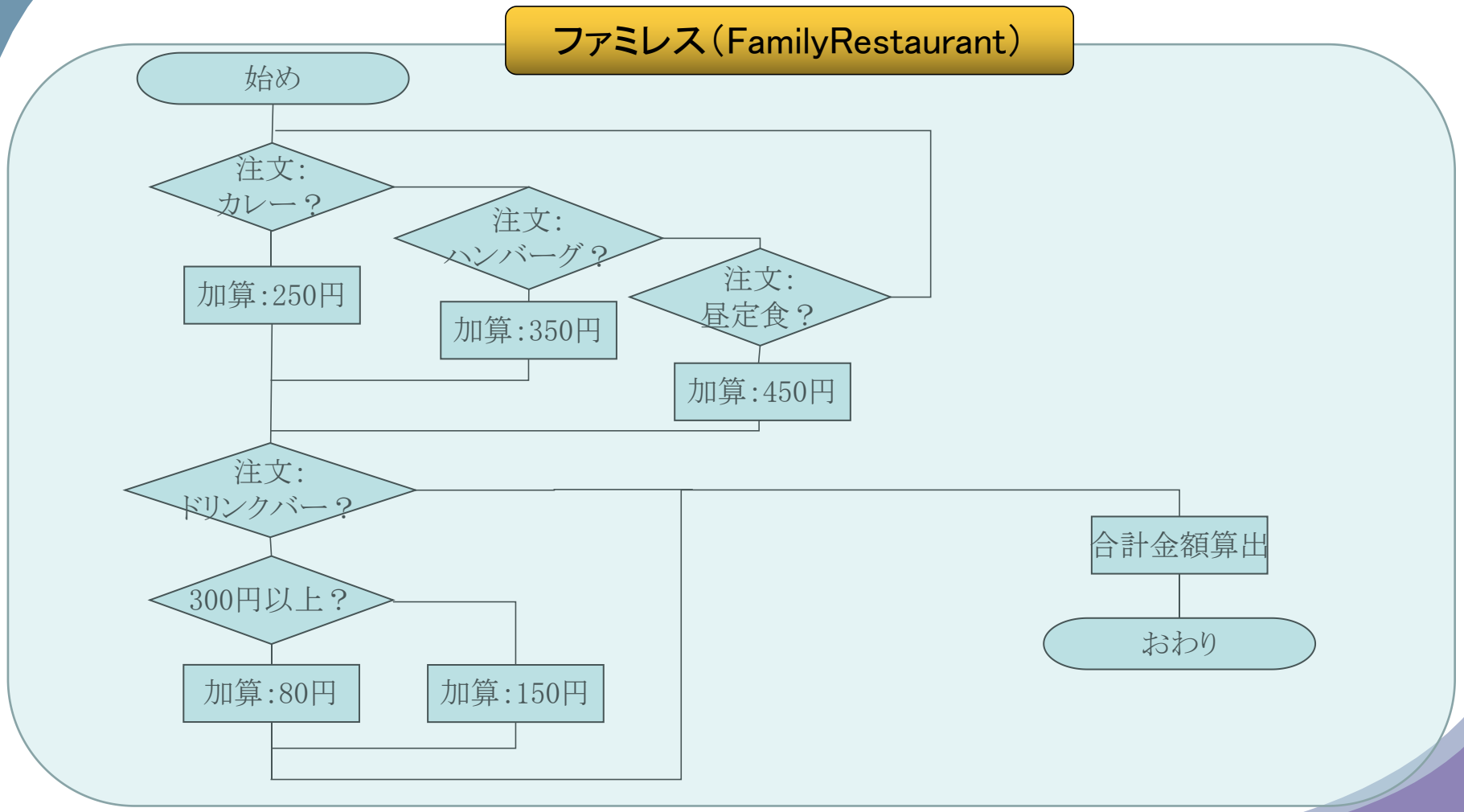
手続き型プログラミングとオブジェクト指向プログラミング

具体的に以下のフローチャートを使って、オブジェクト(=「ひと」「もの」「概念」)を抜き出してみます。



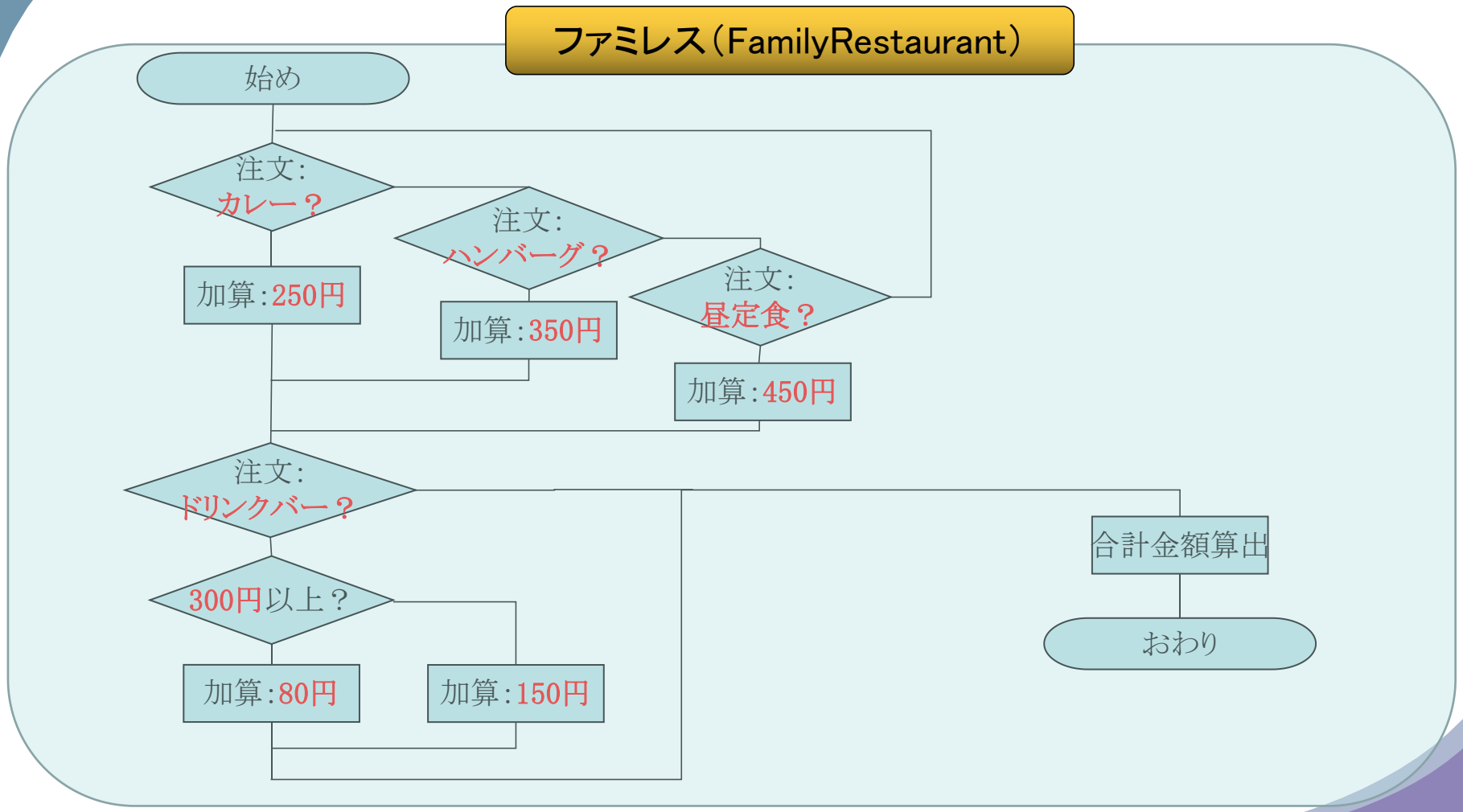
手続き型プログラミングとオブジェクト指向プログラミング

まずは、この中で「データ」に該当するものを抜き出します。



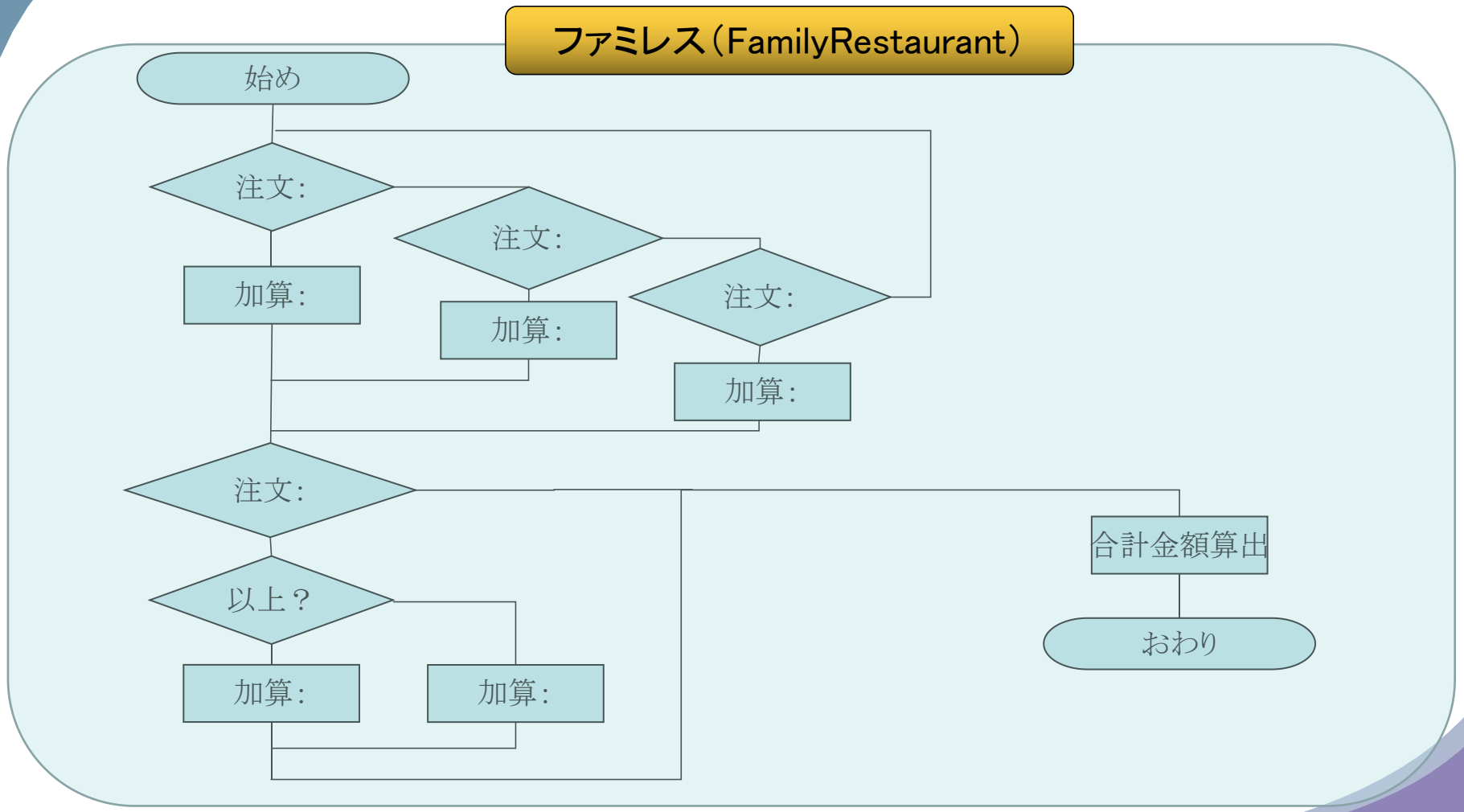
手続き型プログラミングとオブジェクト指向プログラミング

今回は赤色で示したものが、処理をおこなう為の「データ」です。



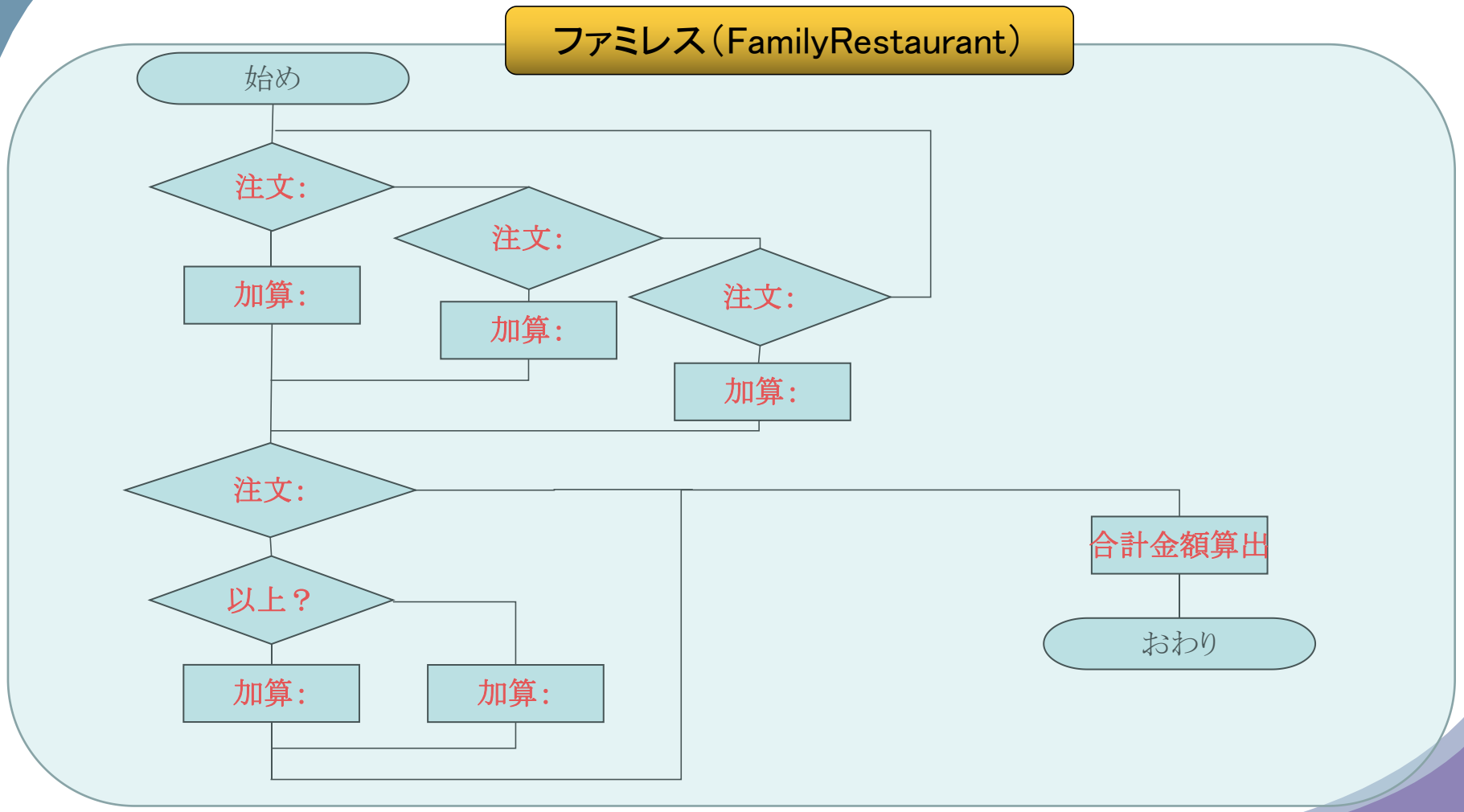
手続き型プログラミングとオブジェクト指向プログラミング

一旦、これらの「データ」は無視しておきます。



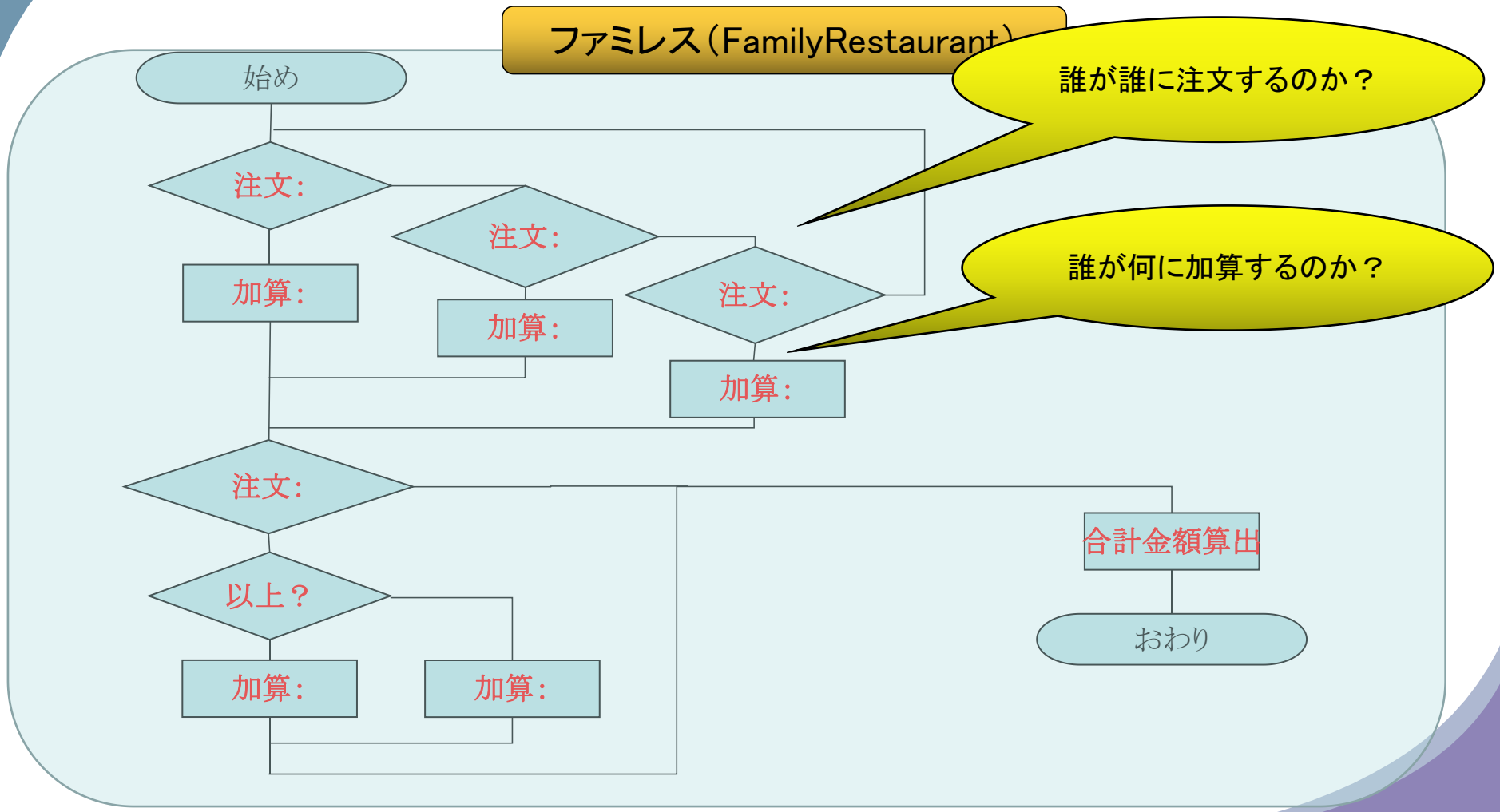
手続き型プログラミングとオブジェクト指向プログラミング

以下の状態から、いよいよ「オブジェクト」を抜き出します。



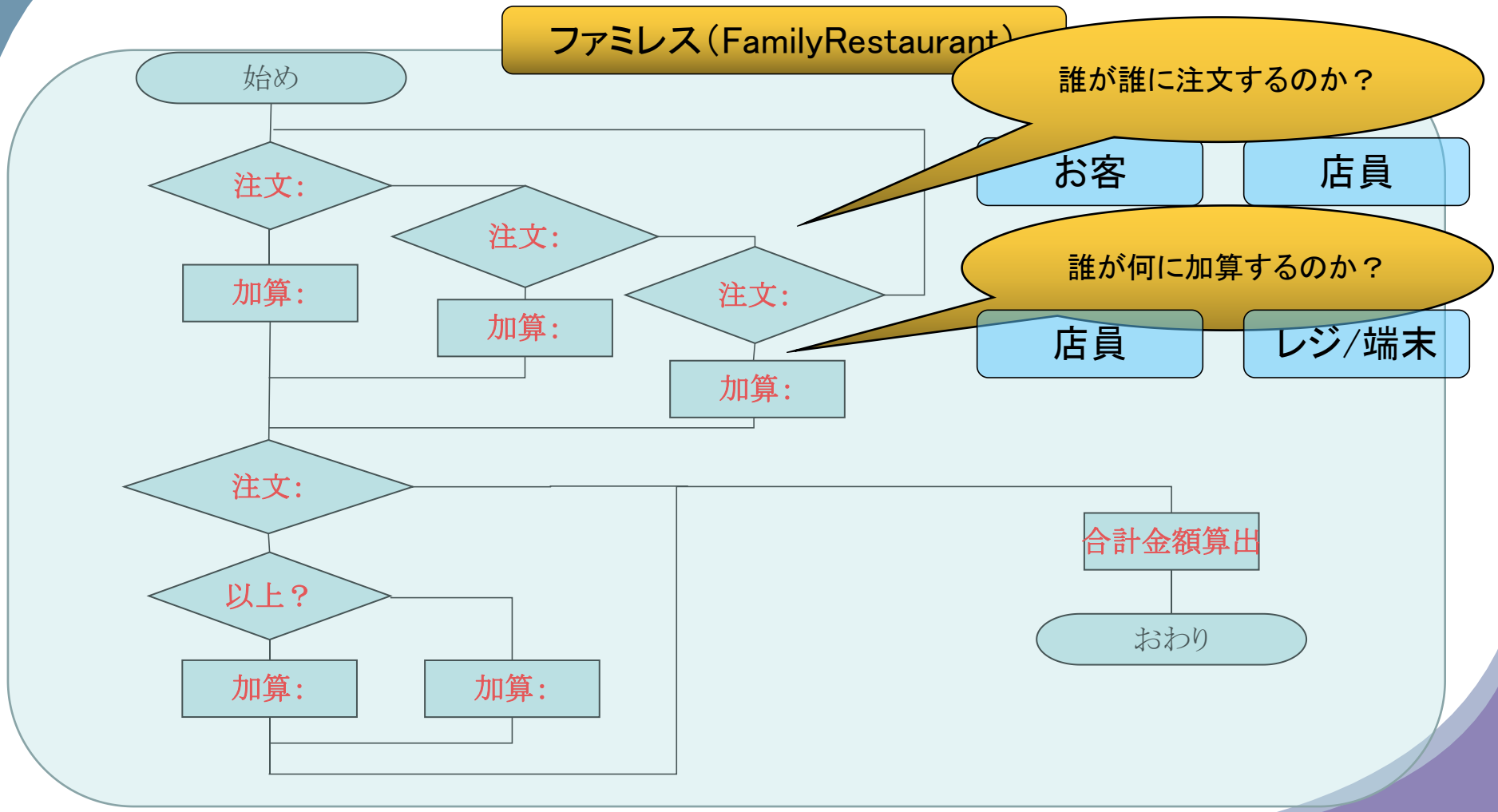
手続き型プログラミングとオブジェクト指向プログラミング

まず、右記のように考えてゆきます。



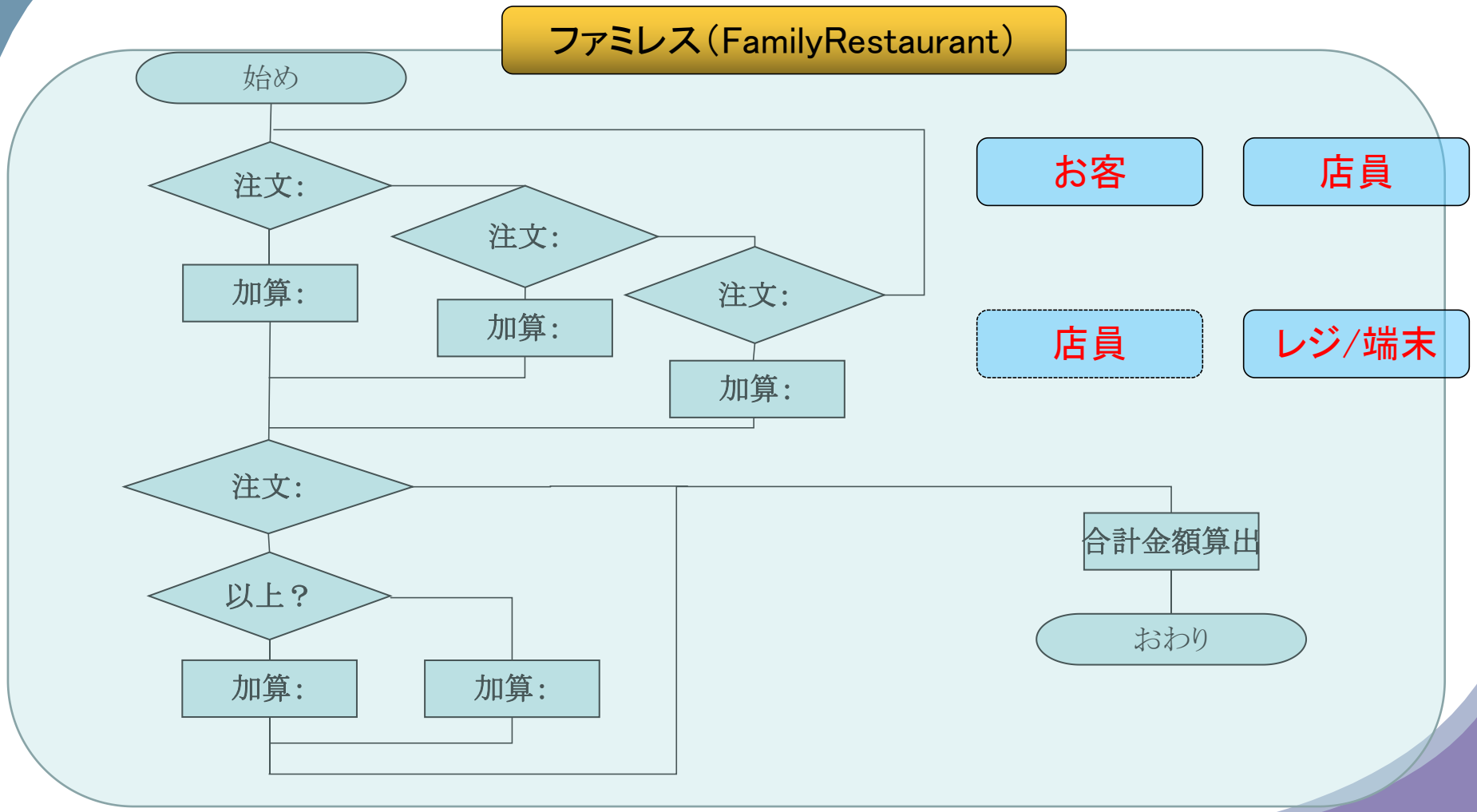
手続き型プログラミングとオブジェクト指向プログラミング

例えば、右図（お客、店員、レジ）のように抜き出しをおこなったとします。



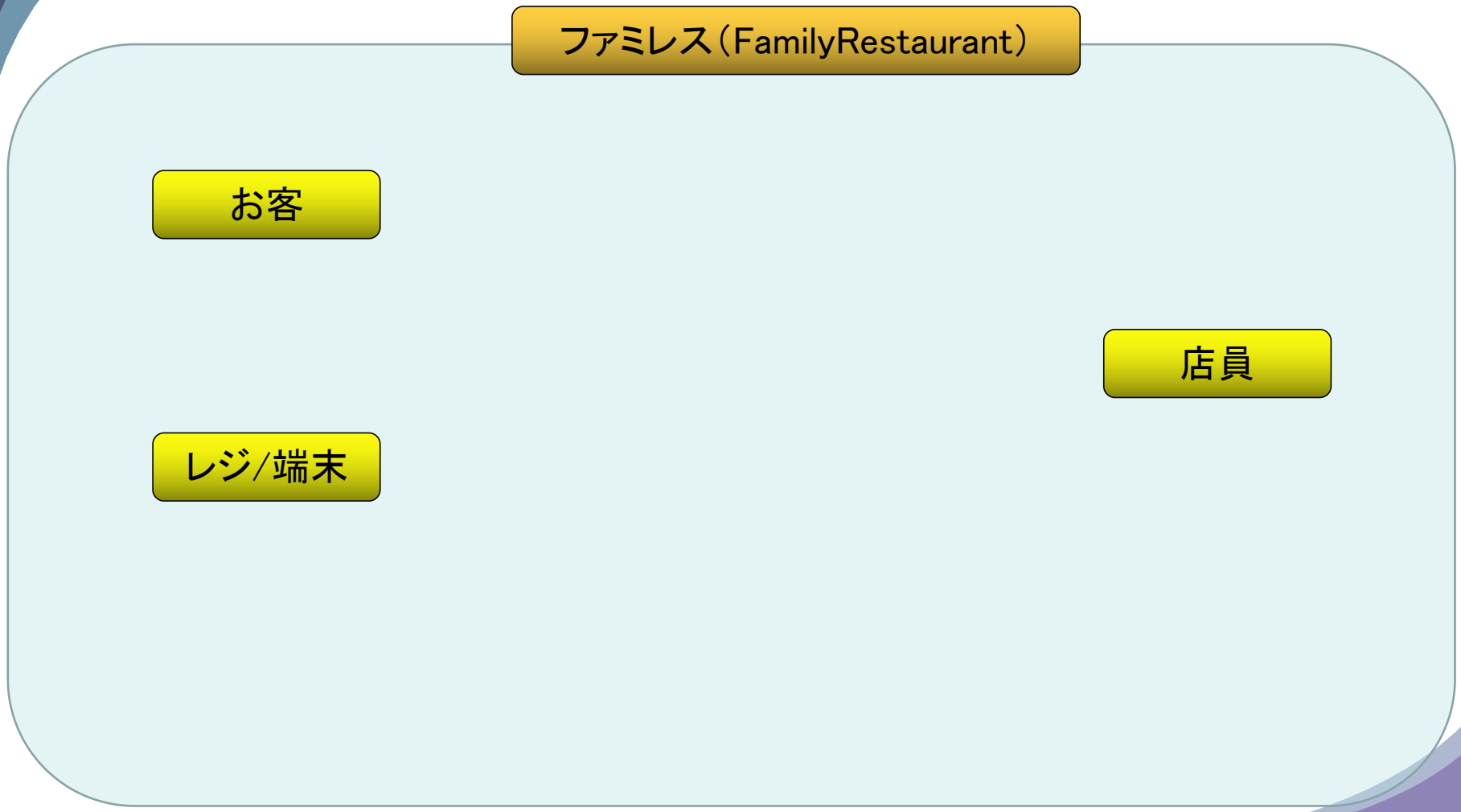
手続き型プログラミングとオブジェクト指向プログラミング

このような場合には、「お客」「店員」「レジ/端末」が「オブジェクト」となります。



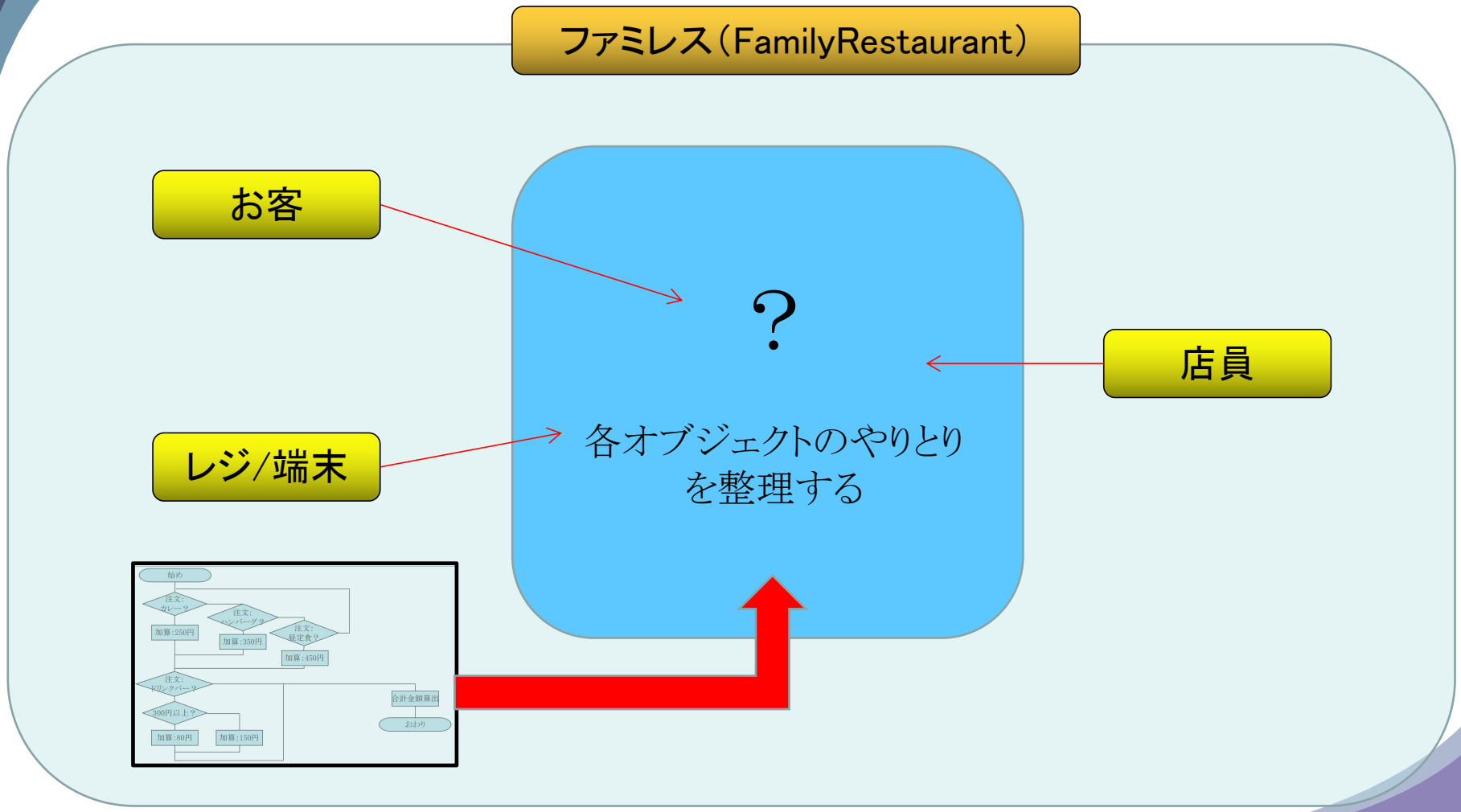
手続き型プログラミングとオブジェクト指向プログラミング

これで「オブジェクト」の抜き出しができました。



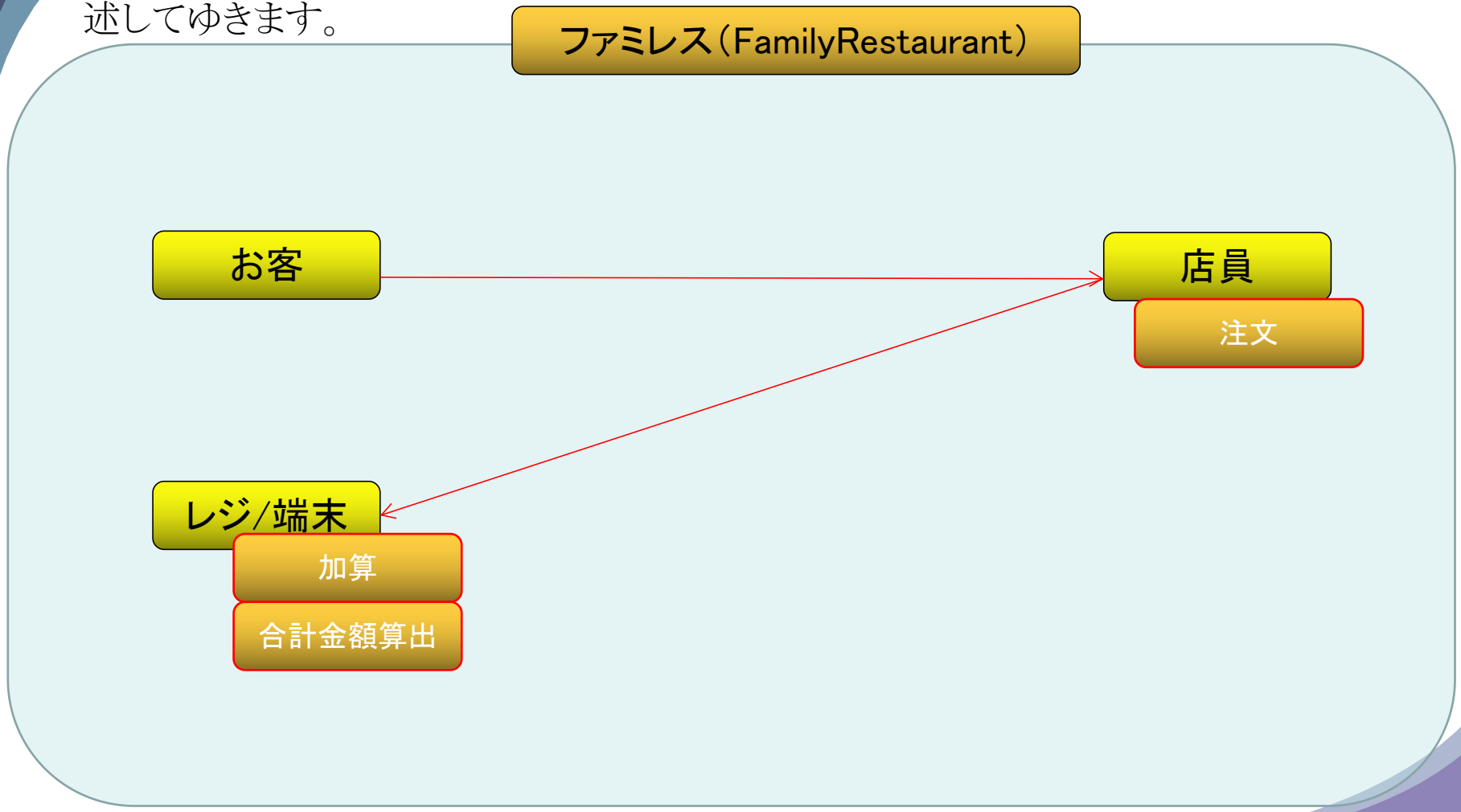
手続き型プログラミングとオブジェクト指向プログラミング

つぎに、フローチャートから各オブジェクトのやりとりを整理してゆきます。



手続き型プログラミングとオブジェクト指向プログラミング

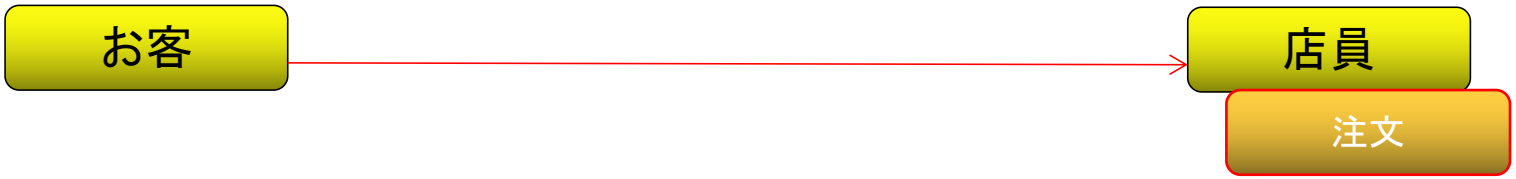
この時、やりとり（注文、加算、合計金額算出）は実際に処理をするオブジェクト側に記述してゆきます。



手続き型プログラミングとオブジェクト指向プログラミング

なお、どのオブジェクトに追加すればよいか、判断しづらい場合もあります。
この場合には、実際に「処理する」という言葉を当てはめてみると判断しやすくなります。

×お客は注文を処理する
○店員は注文を処理する



⇒お客さんは、注文を店員に伝えますが、実際にこの注文を処理しません。
　よって、ここでは店員に「注文」を追加しておきます。

手続き型プログラミングとオブジェクト指向プログラミング

補足：

プログラミング用語として、「処理」と「機能」という言葉があります。

それぞれは意味が異なっています。

「処理」・・・・動き、つまり実際の行動を意味します。
「機能」・・・・動く為の能力、つまり仕組みを意味します。

手続き型プログラミングとオブジェクト指向プログラミング

補足：

例えば、スマートフォンには電源をつける機能があります。
実際にこの機能を使うと、スマートフォンは電源をつける処理をします。

「処理」と「機能」は、これから「オブジェクト指向プログラミング」をおこなう上で、大切な使い分けすべき用語でもあります。

手続き型プログラミングとオブジェクト指向プログラミング

補足：

ここで練習してみましょう。
次のうち、実際に作業をおこなったのはどちらでしょうか？

1. 店員が端末に加算する機能

2. 店員が端末に加算する処理

手続き型プログラミングとオブジェクト指向プログラミング

補足：

⇒解答：
2．店員が端末に加算する処理

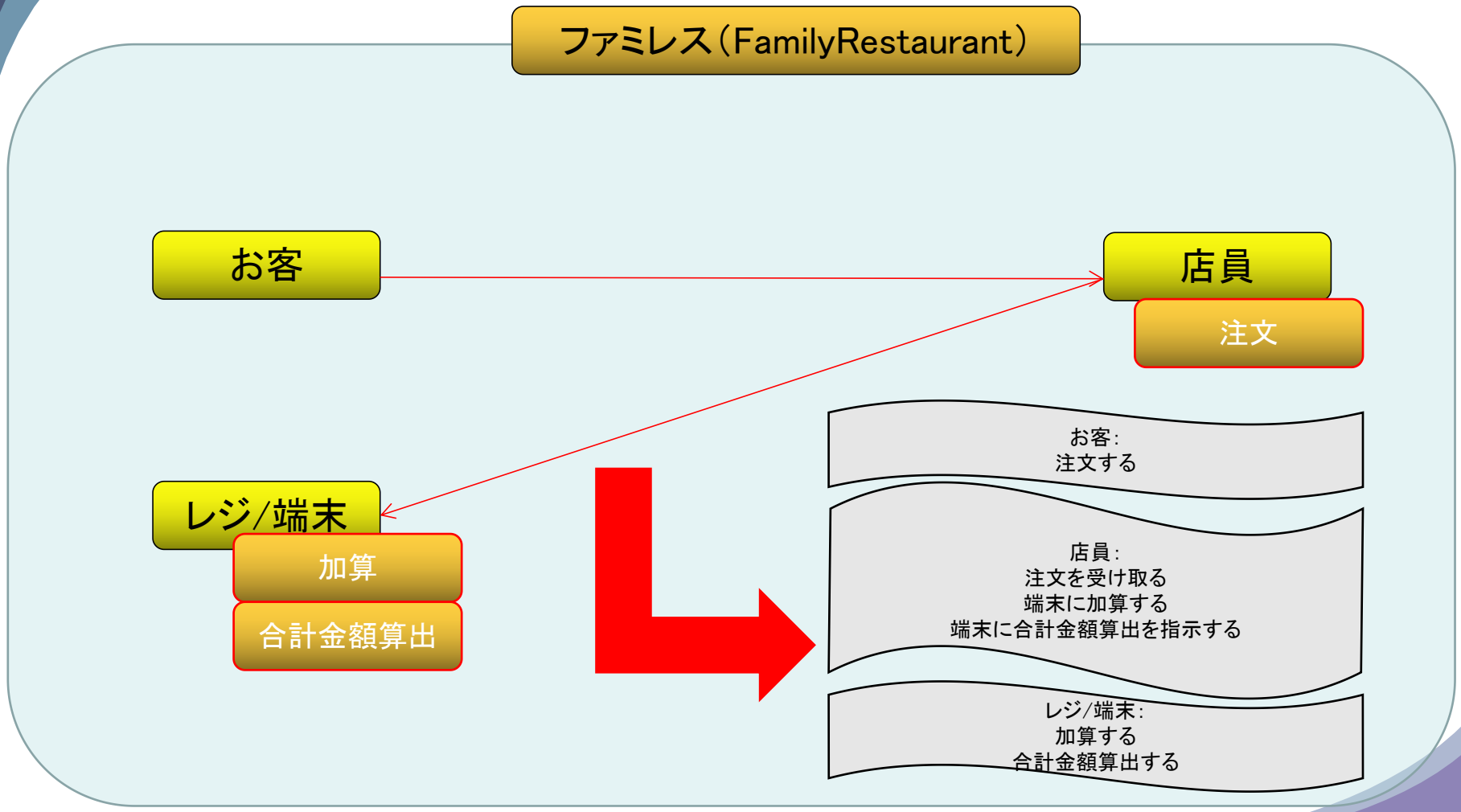
1．の端末に加算する機能は、端末にこの仕組みがあることを示しています。

これに対して、

2．の端末に加算する処理は、「実際にこの機能を使って」作業つまり端末に加算する動きを示しています。

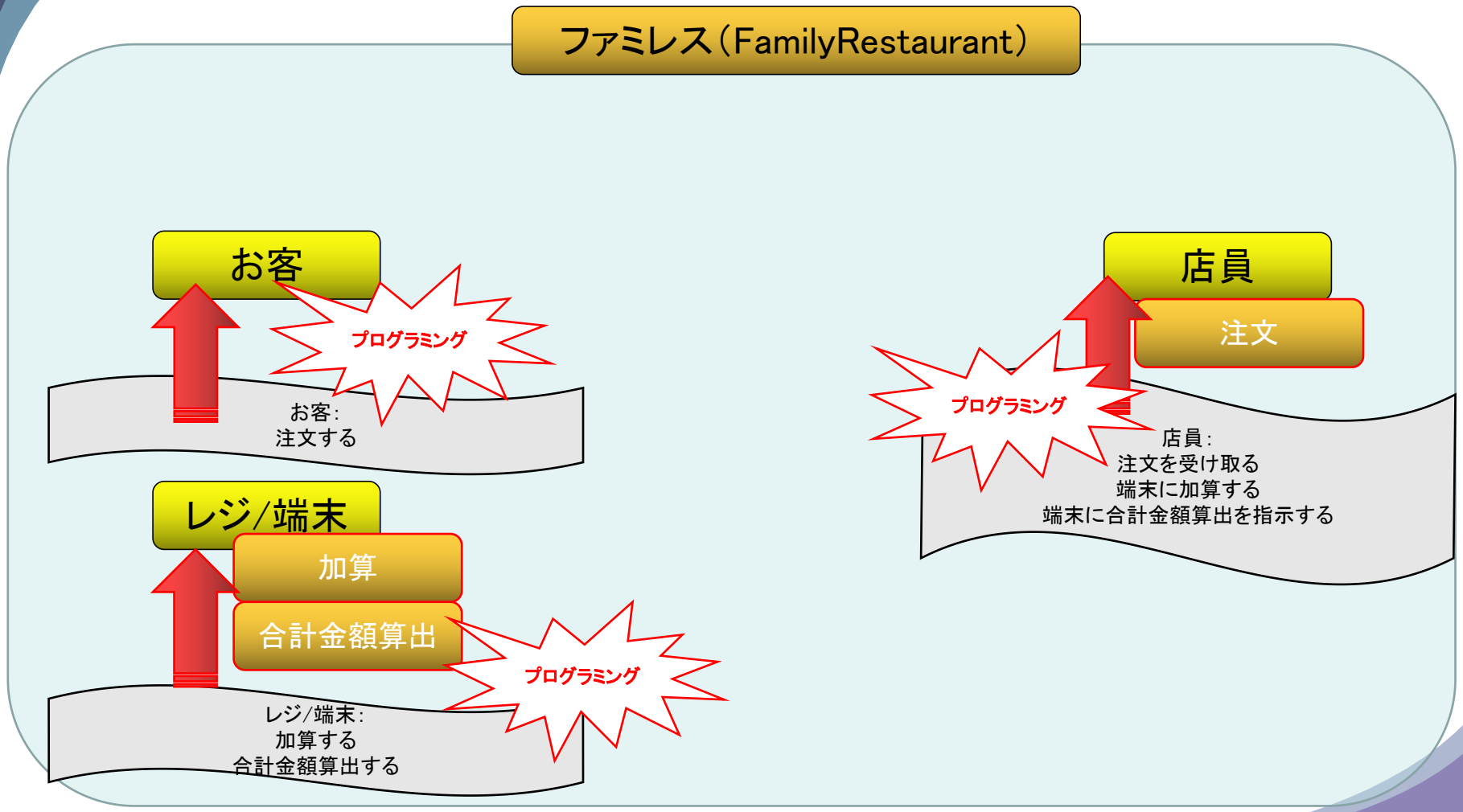
手続き型プログラミングとオブジェクト指向プログラミング

それでは、各オブジェクトをつかって、それぞれの機能を整理してゆきます。



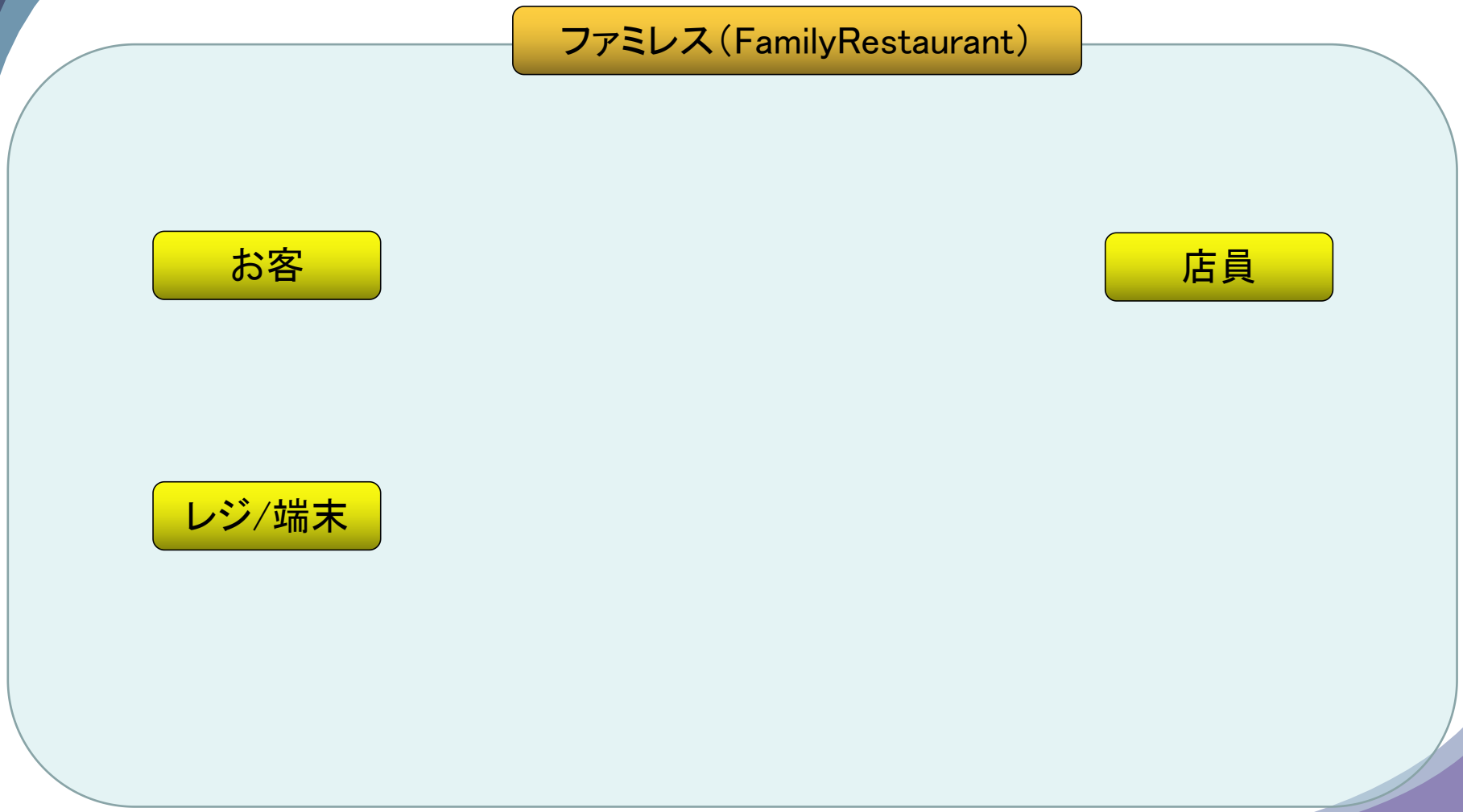
手続き型プログラミングとオブジェクト指向プログラミング

機能を整理できましたら、これをオブジェクト毎にプログラミングしてゆきます。



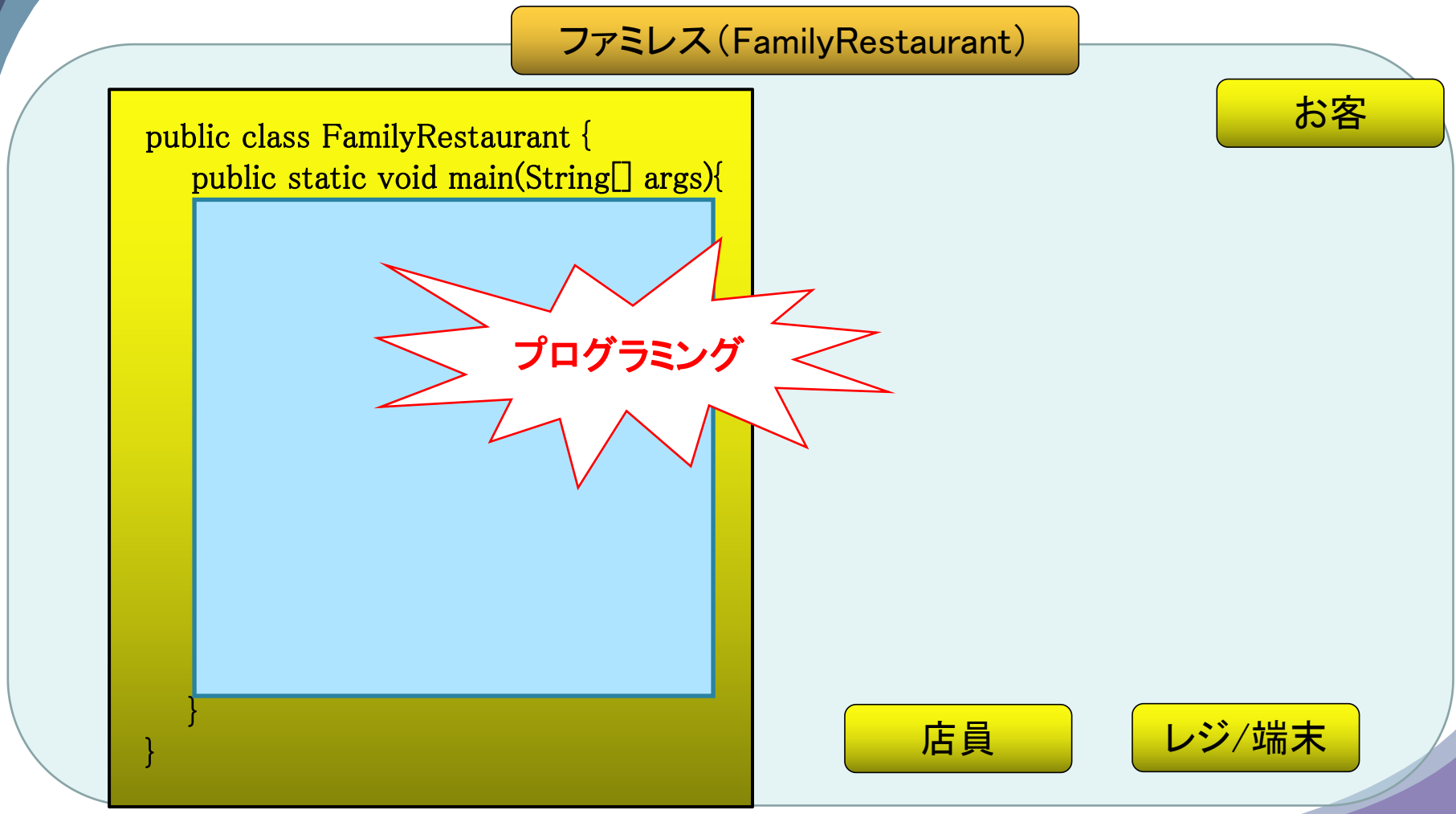
手続き型プログラミングとオブジェクト指向プログラミング

プログラミングできたら、「お客」「店員」「レジ/端末」のオブジェクトは出来上がりです。



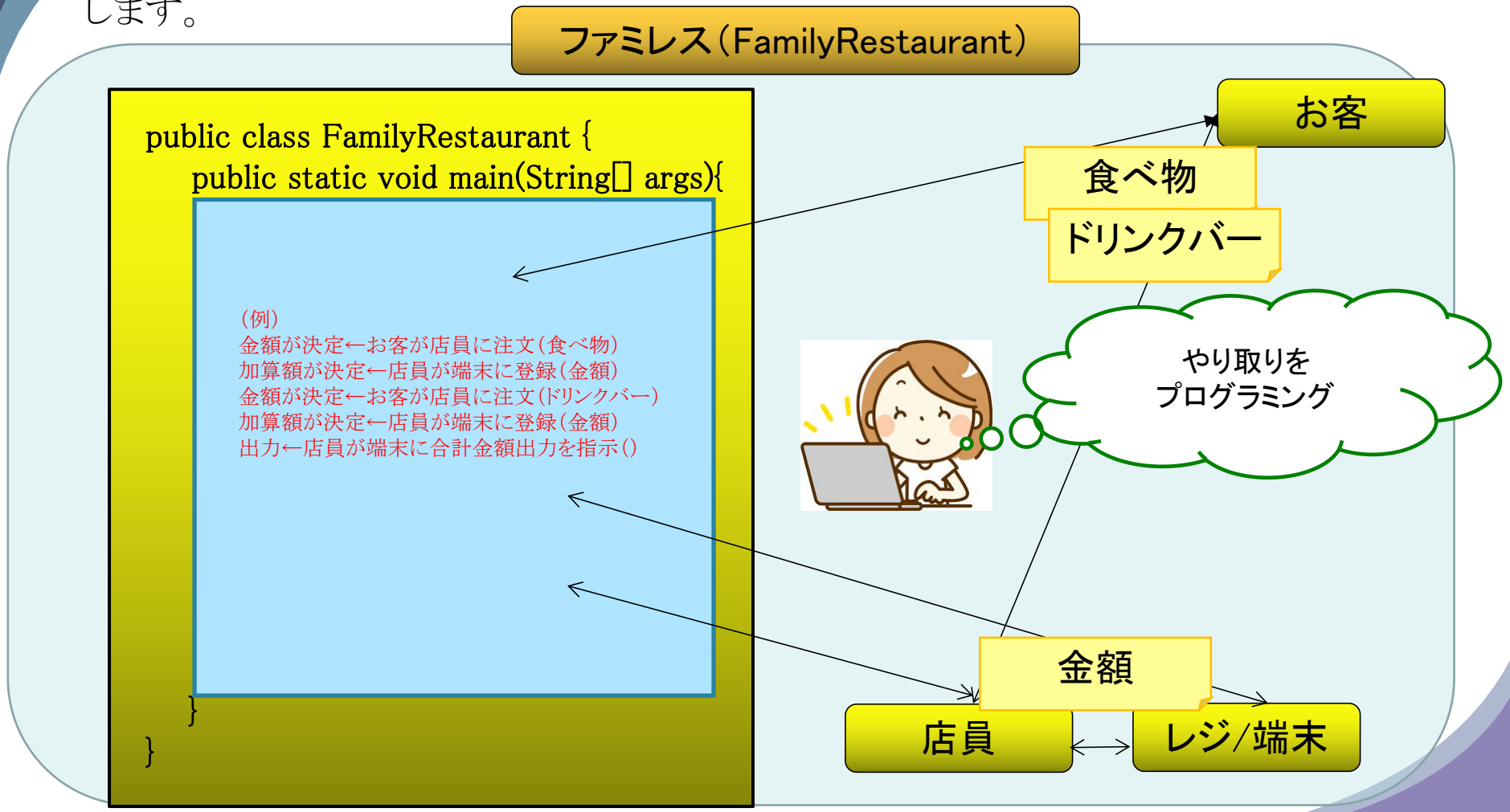
手続き型プログラミングとオブジェクト指向プログラミング

最後にFamilyRestaurantクラスをプログラミングしてゆきます。



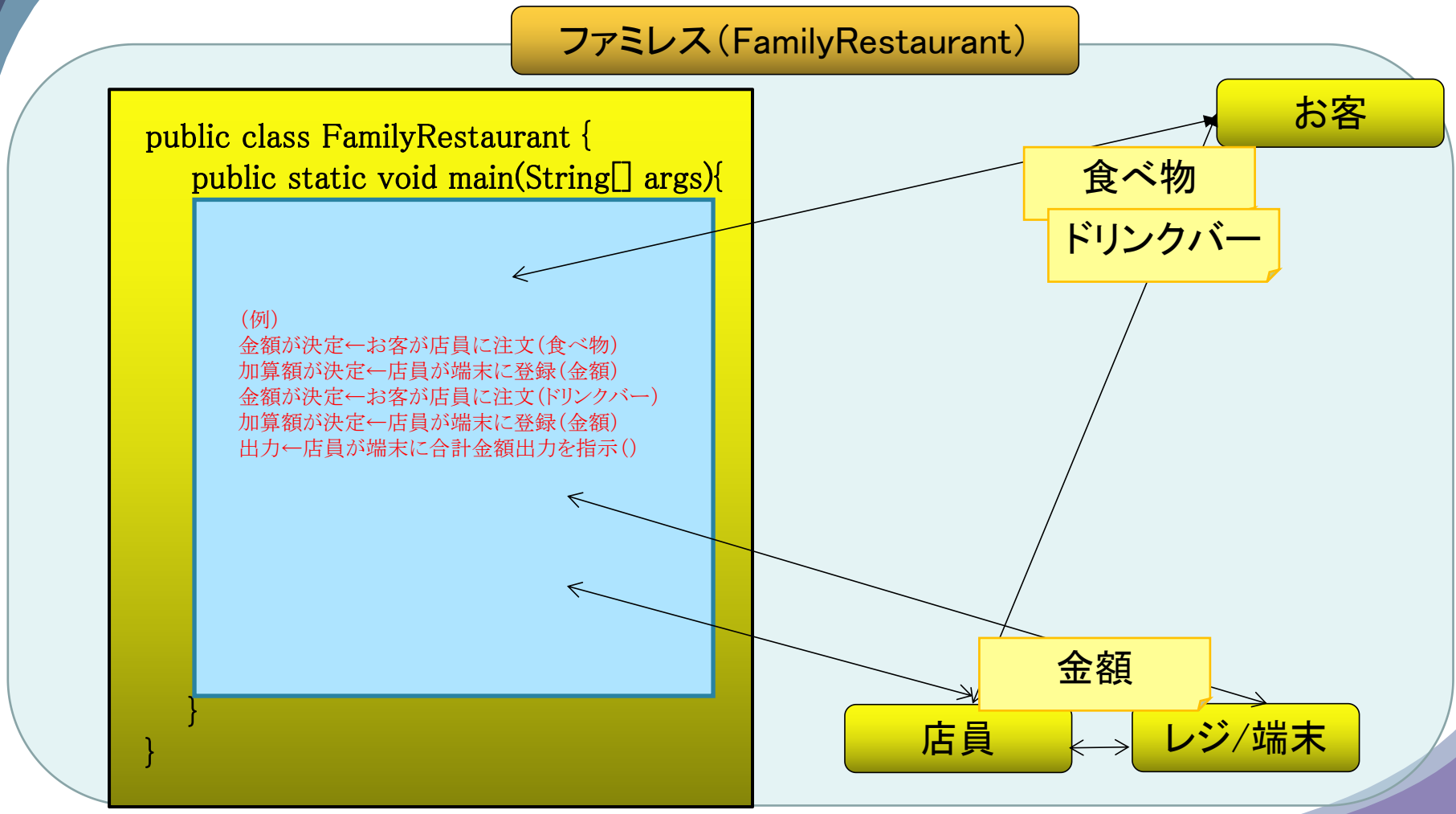
手続き型プログラミングとオブジェクト指向プログラミング

FamilyRestaurantクラスには、オブジェクト同士や、メッセージのやり取りをプログラミングします。



手続き型プログラミングとオブジェクト指向プログラミング

このオブジェクト同士や、メッセージのやり取りを書き終えるとすべて完成です。



手続き型プログラミングとオブジェクト指向プログラミング

「オブジェクト指向プログラミング」はこのように、

・オブジェクト（つまり、「ひと」「もの」「概念」）を抜き出してプログラミング
・オブジェクト同士のやりとりや、メッセージのやり取りをプログラミング

をおこなってプログラムを完成します。

手続き型プログラミングとオブジェクト指向プログラミング

また「オブジェクト指向プログラミング」には、様々なやりとりの仕組みが準備されています。

以降は、この仕組みに触れながら、具体的にプログラミングを進めてゆきます。

以上