第3回　手続き指向のプログラム

～同じ仕事は2度書かない～

学習目標

- メソッドを使う利点を説明できる
  - プログラムの意味とメソッドの関係を説明できる
- メソッドを使ったプログラムを書ける
  - メソッドの書法を説明できる
  - 引数のあるプログラムが書ける
  - 返り値のあるプログラムが書ける

3.1.　同じ意味の仕事はカタマリに

3.1.1.　　同じ仕事が何度も

①.前回のプログラムの問題点

前回のプログラムの問題点、というより面倒だったのはどこですか？

みなさんお分かりでしょう。同じ仕事を何度も書かなければならないことです。商品種類の追加を 4 回も行う次のプログラム(例題 3-1)を読みながら、この問題の解決方法を考えましょう。

例題 3-1：同じ仕事が何度も(Example3_1.java)

```
1:      /**
2:      * オブジェクト指向哲学　入門編
3:      * 例題 3-1：同じ仕事が何度も
4:      * 商品種類を追加するプログラム
5:      *
6:      * メインクラス
7:      */
8:      public class Example3_1 {
9:
10:       /**
11:       * メイン
12:       * 商品種類を追加するプログラム
13:       */
14:       public static void main(String[] args) {
15:
16:         //自動販売機プログラムの開始を知らせる
17:         System.out.println("自動販売機が開始しました。");
18:
19:         //商品種類を保存するための配列を定義する
20:         int[] itemTypeArray = new int[10];
21:
22:         //商品種類を保存するための変数を初期化する
23:         //何も入っていないことを-1 として扱う
24:         for(int i=0;i<10;i++){
25:           itemTypeArray[i] = -1;
26:         }
```

```
27:
28:         //商品種類を追加する
29:         int addId = 1001;//コーラ
30:         for(int i=0;i<10;i++){//商品種類が入っていない箱を探す
31:           if(itemTypeArray[i] == -1){//入っていない
32:             itemTypeArray[i] = addId;//書き込む
33:             break;
34:           }
35:         }
36:
37:         addId = 1002;//ソーダ
38:         for(int i=0;i<10;i++){//商品種類が入っていない箱を探す
39:           if(itemTypeArray[i] == -1){//入っていない
40:             itemTypeArray[i] = addId;//書き込む
41:             break;
42:           }
43:         }
44:
45:         addId = 1003;//お茶
46:         for(int i=0;i<10;i++){//商品種類が入っていない箱を探す
47:           if(itemTypeArray[i] == -1){//入っていない
48:             itemTypeArray[i] = addId;//書き込む
49:             break;
50:           }
51:         }
52:
53:         addId = 1004;//DD レモン
54:         for(int i=0;i<10;i++){//商品種類が入っていない箱を探す
55:           if(itemTypeArray[i] == -1){//入っていない
56:             itemTypeArray[i] = addId;//書き込む
57:             break;
58:           }
59:         }
60:       }
61:
62:     }
```

②.同じ仕事は2度書かない

プログラムの中で同じ意味の仕事は２度書きたくないし、書いてはいけません。特に、「コピーペーストをしてからちょっとだけ変更」なんていうのはもってのほかです。必ず解決方法があるはずなので、「コピーペースト」をする前に良く考えましょう。

＜議論しよう！＞同じ意味の仕事を 2 度書いてはいけない理由

(1)**変更が容易にできるプログラムのために**

(2)**他人が読めるプログラムのために**

(3)**バグの少ないプログラムのために**

3.2.　メソッド

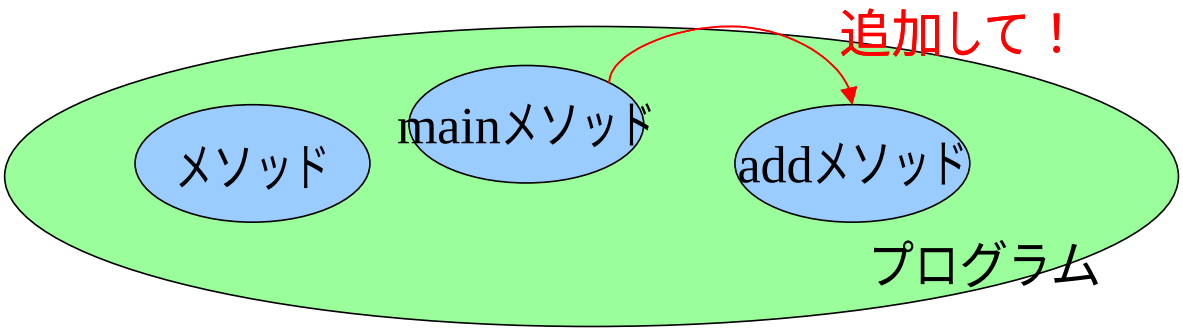
今回は、「メソッド」を使って、仕事をカタマリにする方法を議論していきます。

3.2.1.　　メソッドとは

メソッドは、何度も同じ仕事を書かなくてすむように仕事をまとめて書くことができる仕組みです。

仕事ごとにプログラムを分けると、人間にわかりやすいプログラムになります。
一つのプログラムは、たくさんのメソッドから成り立つことができ、それらのメソッドが仕事を分担して目的を達成することができます。

また、main()もメソッドの一つです。main メソッドはクラスの中に必ず 1 つだけ存在するメソッドで、このメソッドからクラスの仕事は始まります。そしてこの main メソッドから他のメソッドを呼び出して仕事をさせることができます。



例題 3-2：メソッドの利用(Example3_2.java)

```
1:    /**
2:    * オブジェクト指向哲学　入門編
3:    * 例題 3-2：メソッドの利用
4:    * 商品種類を追加するプログラム
5:    *
6:    * メインクラス
7:    */
8:    public class Example3_2 {
9:
10:     /**
11:     * メイン
12:     * 商品種類を追加するプログラム
13:     */
14:     public static void main(String[] args) {
15:
16:       //自動販売機プログラムの開始を知らせる
17:       System.out.println("自動販売機が開始しました。");
18:
19:       //商品種類を保存するための配列を定義する
20:       int[] itemTypeArray = new int[10];
21:
22:       //商品種類を保存するための変数を初期化する
23:       //何も入っていないことを-1 として扱う
24:       for(int i=0;i<10;i++){
25:         itemTypeArray[i] = -1;
26:       }
27:
28:       //商品種類を追加する
29:       add(itemTypeArray,1001);//コーラ
30:       add(itemTypeArray,1002);//ソーダ
31:       add(itemTypeArray,1003);//お茶
32:       add(itemTypeArray,1004);//DD レモン
33:     }
34:
35:     /**
36:     * 商品種類を追加する
37:     */
38:     public static void add(int[] targetArray,int addId){
39:       //商品種類が入っていない箱を探す
40:       for(int i=0;i<10;i++){
41:         if(targetArray[i] == -1){//入っていない
42:           targetArray[i] = addId;//書き込む
43:           break;
44:         }
45:       }
46:     }
47:
48:   }
```

3.2.2.　メソッドを使ったプログラム

①.メソッドを定義する

(1)書式

ここでは商品種類を追加するメソッドを用いて説明します。

```
/**
* 商品種類を追加する
*/
public static void add(int[] targetArray, int addId){
  //商品種類が入っていない箱を探す
  for(int i=0;i<10;i++){
    if(targetArray[i]==-1){
      targetArray[i] = addId;
      break;
    }
  }
}
```

- コメント：(商品種類を追加するメソッド)

メソッドの説明です。メソッドには、必ずどのような仕事をするのか、コメントで書いておきます。これを書くことによってメソッドの中身を全て読まなくてもメソッドの働きがわかり、後で見直すときとても読みやすくなります。

- メソッドの名前：(add)

メソッドの名前です。public static void の後に記述します。public static void はここではおまじないと考えてください。

- 引数：(int[] targetArray, int addId)

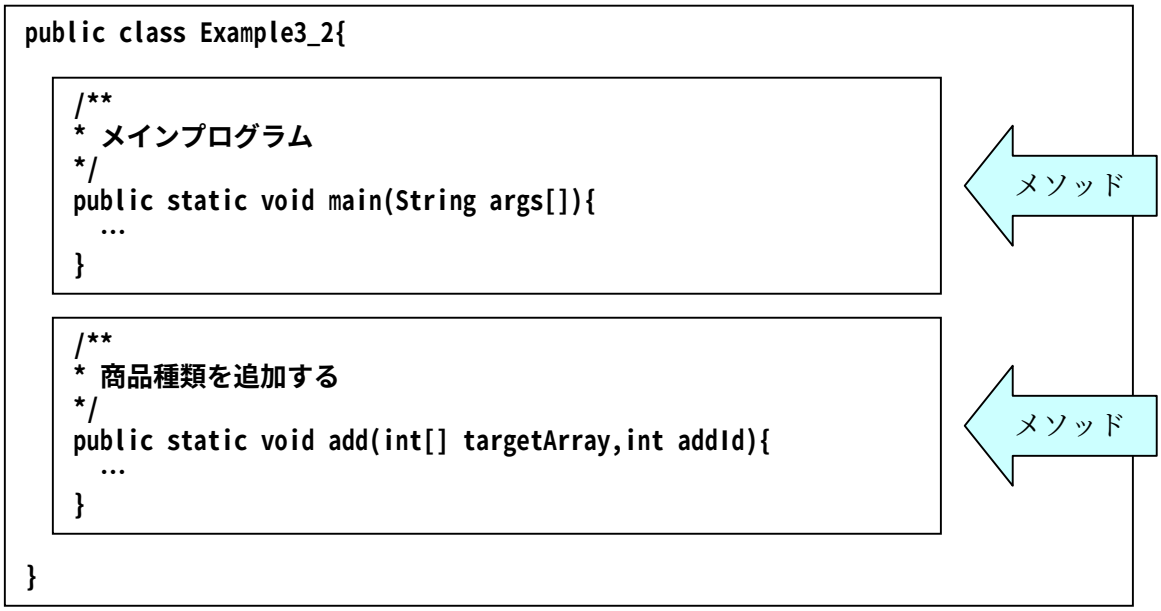
メソッドを呼び出すときに、呼び出し側からメソッドへ引き渡す必要のある数値のことです。詳しくは後述します。

- 仕事：(for(int i=0;i<10;i++)…))

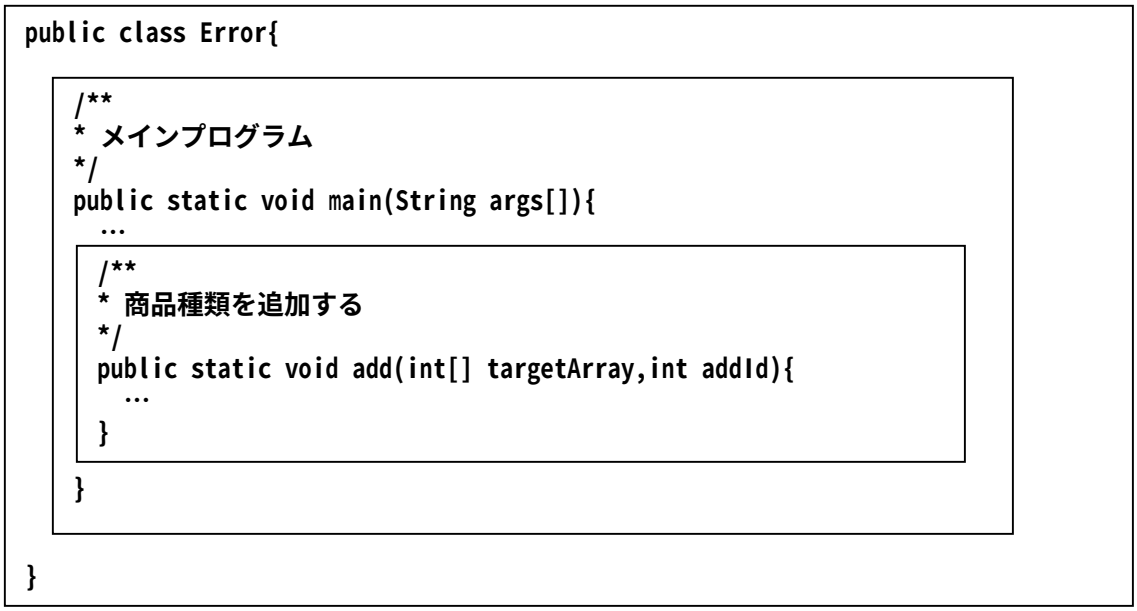
メソッドに実際に動かさせたい内容を書きます。そのメソッドを呼び出すことによってこの仕事をさせることができます。

(2)メソッドを書く位置

Java では、クラスの中にメソッドを書きます。(メソッドの中にはメソッドを書けません。)



まずい例



メソッドの中にメソッドを入れてはいけません。

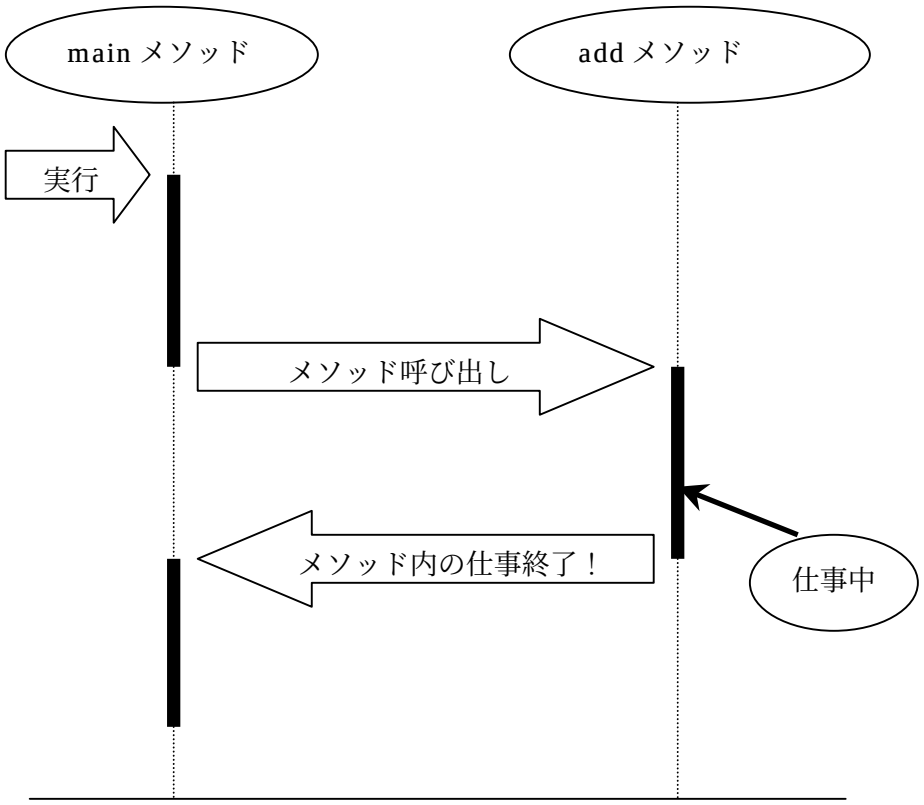
②.メソッドを呼び出す

(1)クラスを実行して呼び出されるのは main メソッドだけ

メソッドは定義しただけでは意味がありません。

プログラムを実行して、自動的に呼ばれるのは、main メソッドだけです。メソッドは、メソッドから呼び出されて仕事をします。

仕事が終わると、呼び出し側が仕事の続きをします。



(2)メソッドを呼び出す

```
public class Example3_2{

  /**
  * プログラム・メイン
  */
  public static void main(String args[]){
    //追加する（メソッドを呼び出す）
    add(itemTypeArray,1001);


  }

  /**
  * 商品種類を追加する
  */
  public static void add(int[] targetArray, int addId){
    ...
  }
}
```

＜考えよう！＞次のプログラムの出力はどうなりますか？

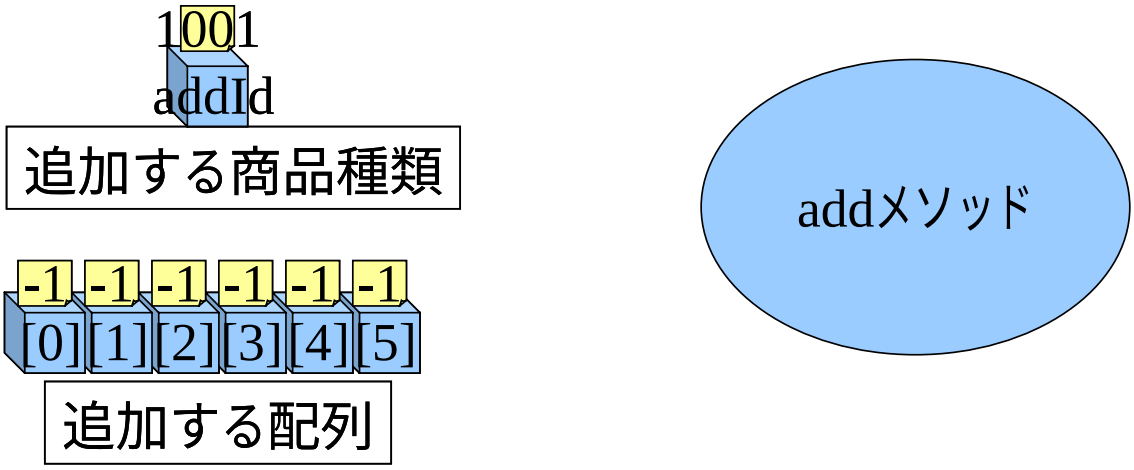
練習問題 3-1:メソッドの呼び出し練習問題

```
/**
* メソッドの呼び出し練習問題
*/
public class Exercise3_1{

  public static void main(String args[]){
    System.out.println("A");
    method1();
    System.out.println("B");
    method2();
    System.out.println("C");
  }

  public static void method1(){
    System.out.println("D");
    method2();
    System.out.println("E");
  }

  public static void method2(){
    System.out.println("F");
  }

  public static void method3(){
    System.out.println("G");
  }

}
```

③.引数を渡す

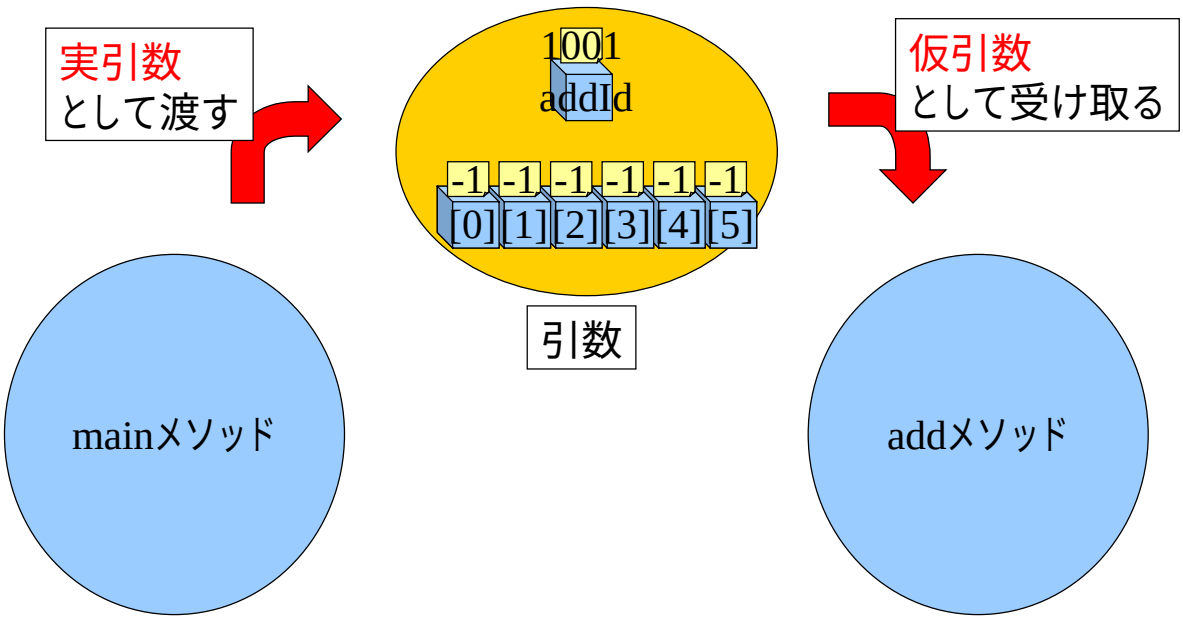
(1)仕事をするには材料が必要

- 仕事をするときに、材料が必要な場合があります。
  ・「追加する」という仕事をするためには、どんな材料が必要ですか？



(2)材料の引き渡し方

メソッドでは、「引数（ひきすう）」を利用して仕事をするための材料の受け渡しを行います。



(3)実引数と仮引数の結合

```
/**
* オブジェクト指向哲学　入門編
* 例題 3-2：メソッドの利用
* 商品種類を追加するプログラム
*
* メインクラス
*/
public class Example3_2 {

  /**
  * メイン
  * 商品種類を追加するプログラム
  */
  public static void main(String[] args) {

    //自動販売機プログラムの開始を知らせる
    System.out.println("自動販売機が開始しました。
");

    //商品種類を保存するための配列を定義する
    int[] itemTypeArray = new int[10];

    //商品種類を保存するための変数を初期化する
    //何も入っていないことを-1 として扱う
    for(int i=0;i<10;i++){
      itemTypeArray[i] = -1;
    }

    //商品種類を追加する
    add(itemTypeArray,1001);//コーラ
    add(itemTypeArray,1002);//ソーダ
    add(itemTypeArray,1003);//お茶
    add(itemTypeArray,1004);//DD レモン
  }

  /**
  * 商品種類を追加する
  */
  public static void add(int[] targetArray,int
addId){
    //商品種類が入っていない箱を探す
    for(int i=0;i<10;i++){
      if(targetArray[i] == -1){//入っていない
        targetArray[i] = addId;//書き込む
        break;
      }
    }
  }

}
```

- 実引数と仮引数の結合

メソッドが呼ばれると、実引数が仮引数に自動的に代入されるので、材料が渡せます。

実引数

メソッドを呼び出すときに実引数を渡します。

・宣言ではありません。実際の値を渡します。

・仮引数に代入する感じで。

仮引数

メソッドを宣言するときに仮引数を宣言します。

・通常の変数の宣言と同じ形にします。

・複数ある場合は「,」で区切ります。

Java Tips ―変数のスコープ(1)―

以下ののソースに変数の有効範囲を書きこんでみましょう。

- メソッド内で宣言した変数はメソッド内でのみ有効

```
public class A{
  public static void main(String args[]){
    int i;
    i = 3;
  }

  public static void methodA(int x){
    i = 3;
  }
}
```

- メソッドの仮引数の有効範囲もそのメソッド内のみ

```
public class A{
  public static void main(String args[]){
    x = 3;
  }

  public static void methodA(int x){
    x = 3;
  }
}
```

引数とスコープの関係

スコープで説明したように、add メソッドは main メソッドでのみ有効な「itemTypeArray」を知ることができません。

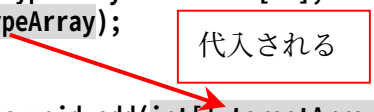
```
public class Example3_2{

  public static void main(String args[]){
    int[] itemTypeArray = new int[10];
    System.out.println(itemTypeArray[3]); //OK
  }

  public static void add(){

    System.out.println(itemTypeArray[3]); //コンパイルエラー

  }
}
```

add メソッドが itemTypeArray を知るためには、教えてもらう必要があります。そこで仮引数を利用するのです。

```
public class Example3_2{

  public static void main(String args[]){
    int[] itemTypeArray = new int[10];
    System.out.println(itemTypeArray[3]);
  }

  public static void add(int[] targetArray){

    System.out.println(targetArray[3]); //OK

  }
```

main メソッドは実引数として値 itemTypeArray を与えます。

```
public class Example3_2{

  public static void main(String args[]){
    int[] itemTypeArray = new int[10];
    add(itemTypeArray);
  }

  public static void add(int[] targetArray){

    System.out.println(targetArray[3]);

  }
}
```

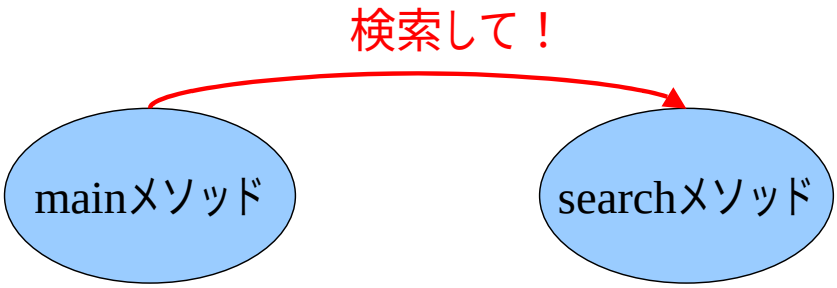


与えられた実引数 itemTypeArray が仮引数 targetArray に代入され、add メソッドで追加するために使えるようになります。

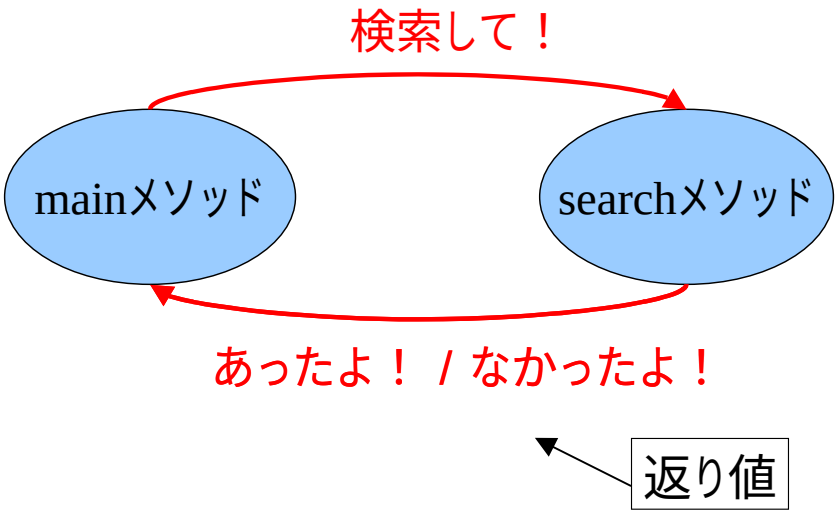
④.返り値のあるメソッド

(1)検索結果を返してほしい（返り値）

追加メソッドを参考にしながら、検索メソッド「search」を作ることを考えます。



main メソッドは検索してもらったら、結果（あった、なかった）を知りたいですね！



メソッドは、仕事の結果を「返り値」として返すことができます。

(2)返り値を返すメソッドを書く

返り値を返す search メソッドを含んだ検索プログラムを例題 3-3 に示します。

例題 3-3：返り値のあるメソッド(Example3_3.java)

```
1:      /**
2:      * オブジェクト指向哲学　入門編
3:      * 例題 3-3：返り値のあるメソッド
4:      * 商品種類を追加、検索するプログラム
5:      *
6:      * メインクラス
7:      */
8:      public class Example3_3 {
9:
10:       /**
11:       * メイン
12:       * 商品種類を追加、検索するプログラム
13:       */
14:       public static void main(String[] args) {
15:
16:         //自動販売機プログラムの開始を知らせる
17:         System.out.println("自動販売機が開始しました。");
18:
19:         //商品種類を保存するための配列を定義する
20:         int[] itemTypeArray = new int[10];
21:
22:         //商品種類を保存するための変数を初期化する
23:         //何も入っていないことを-1 として扱う
24:         for(int i=0;i<10;i++){
25:           itemTypeArray[i] = -1;
26:         }
27:
28:         //商品種類を追加する
29:         add(itemTypeArray,1001);//コーラ
30:         add(itemTypeArray,1002);//ソーダ
31:         add(itemTypeArray,1003);//お茶
32:         add(itemTypeArray,1004);//DD レモン
33:
34:         //商品種類を検索する
35:         boolean isFound = search(itemTypeArray,1001);//コーラを検索
36:         System.out.println(isFound);//検索結果を表示
37:
38:       }
39:
40:       /**
```

```
41:       * 商品種類を追加する
42:       */
43:       public static void add(int[] targetArray,int addId){
44:         //商品種類が入っていない箱を探す
45:         for(int i=0;i<10;i++){
46:           if(targetArray[i] == -1){//入っていない
47:             targetArray[i] = addId;//書き込む
48:             break;
49:           }
50:         }
51:       }
52:
53:       /**
54:       * 商品種類を検索する
55:       */
56:       public static boolean search(int[] targetArray,int searchId){
57:         //一つ一つ商品種類を探す
58:         for(int i=0;i<10;i++){
59:           if(targetArray[i] == searchId){//見つかった
60:             return true;
61:           }
62:         }
63:
64:         //見つからなかった
65:         return false;
66:       }
67:     }
```

(3)返り値をどの型で返すのかを決める

今回の場合、「あった、なかった」なので、真偽値型の boolean を使うことにしましょう。

true　= あった false　= なかった

(4)返り値のあるプログラムの書式

メソッド宣言の返り値を boolean にします。

追加メソッドの時は、「void」でした。
void とは、「返り値がない」という意味なのでした。

```
/**
* 商品種類を検索する
*/
public static boolean search(int[] targetArray,int searchId){
  //一つ一つ商品種類を探す
  for(int i=0;i<10;i++){
    if(targetArray[i] == searchId){//見つかった
      return true;
    }
  }
  //見つからなかった
  return false;
}
```

return[値]で返り値を返します。

また、返り値を返すとメソッドは終了します。

Q：このプログラム間違ってない？
見つかって true を返した後、for 文を抜けたら、false も返してしまわないですか？

A：間違っていません。
値を返した時点で、メソッドは終了し、main に戻るので、見つかったときもうまく動きます。

3.3.　プログラムの意味とメソッド

メソッドを使って、仕事を分担していくのが、手続き指向の基本です。どのように分担していけばよいですか？

3.3.1.　　プログラムの意味

ここでいうプログラムの意味とは、第 1 回で行ったプログラムの目的と似たようなもので、人間から見た「プログラムの意味」のことです。プログラムは人間が目的を果たすためにかかれますから、プログラムの意味ごとにまとめておくのは重要なことですね。

目的の階層構造を、もう一度いままでやってきた商品種類の管理プログラムを題材にして考えてみましょう。どのような階層構造になりますか？そして、それはメソッドとどのように関係していますか？考えてみましょう。

-
  -
  -
  -
  -

練習問題

＜記述問題＞

☆記述問題 3-1

メソッドを使う利点を自分の言葉で説明せよ。また、重複コードになっていてもメソッドとしてまとめるべきでない時はどのような時か考えよ。

＜プログラム問題＞

☆プログラム問題 3-1

プログラム問題 2-1 で作ったプログラムを、メソッドを使って書き直せ。仕様も同様とする。