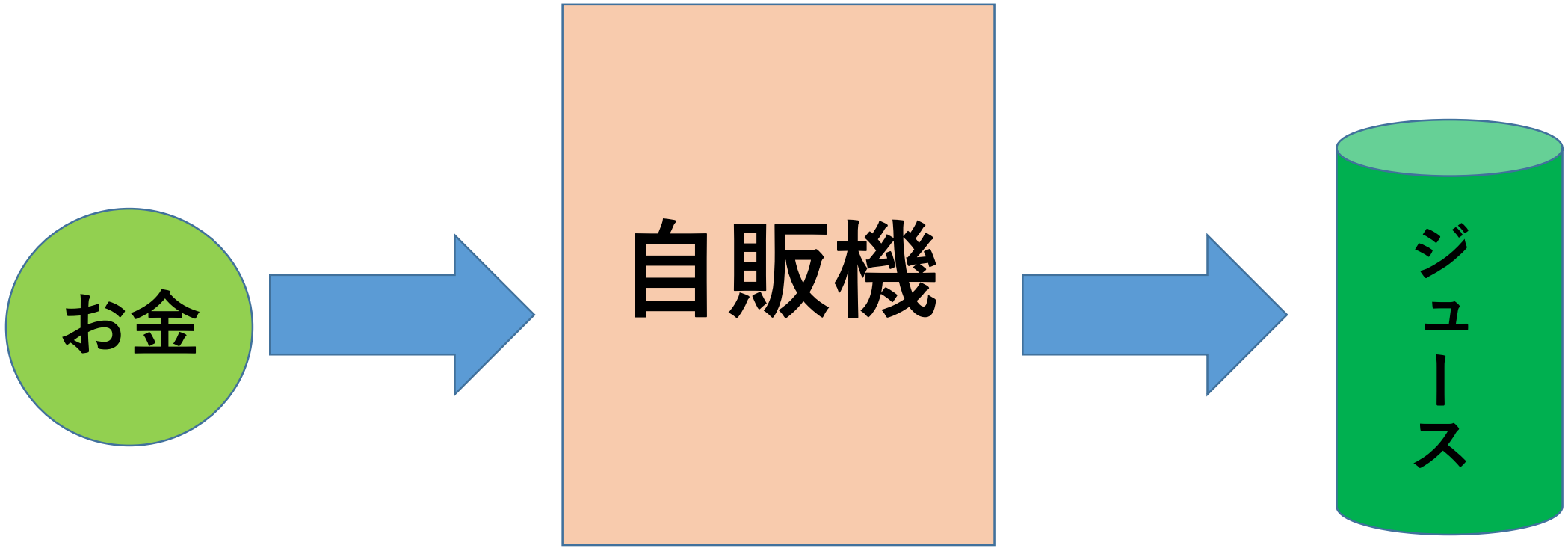
Java 講習7

メソッドその2 (復習)

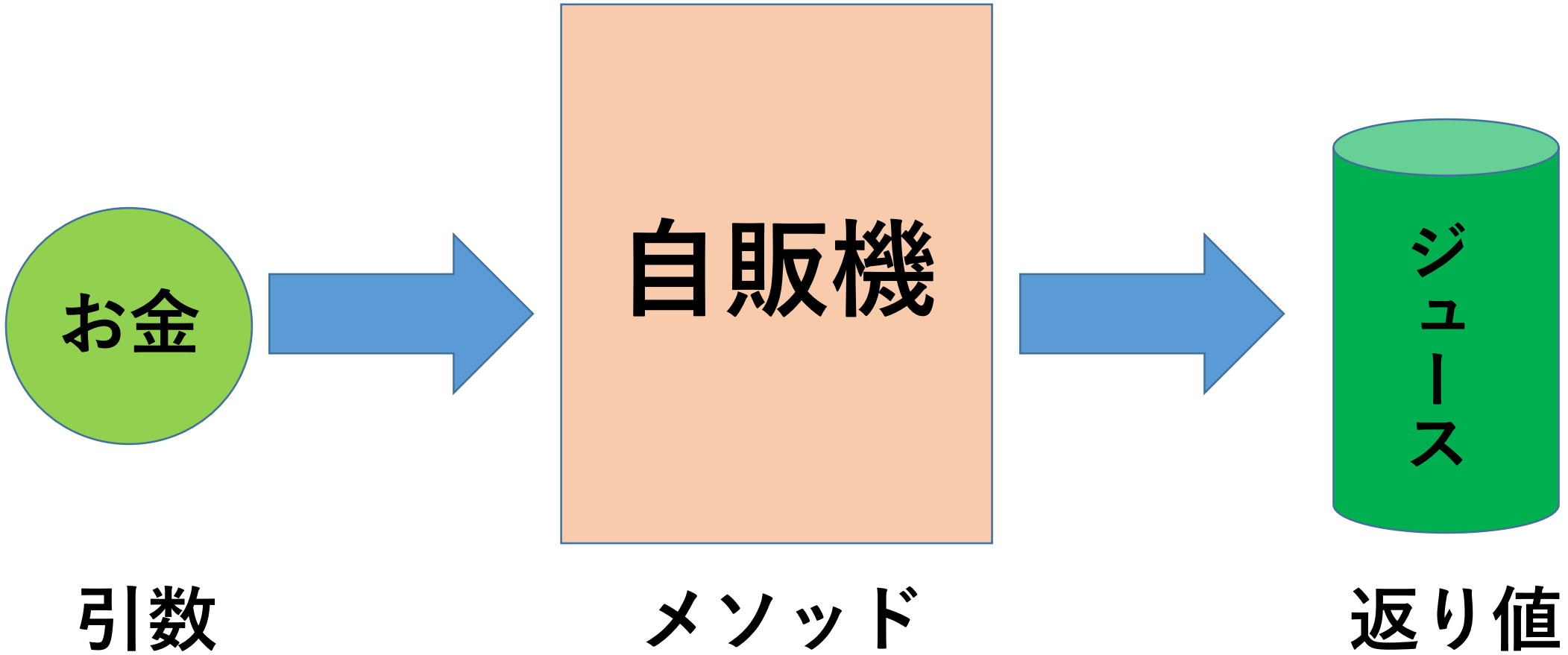
メソッドとは...

- 関数　　ある値を与えると何らかの値を返す



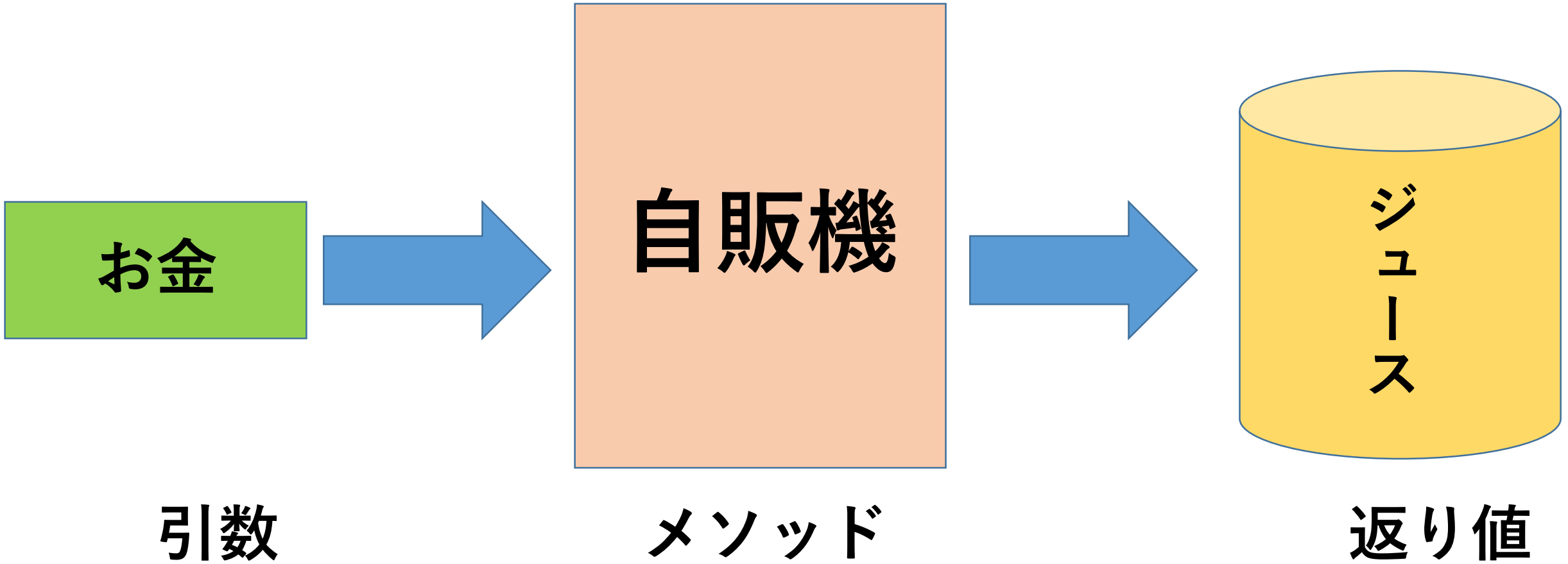
メソッドとは. . .

- 関数　　ある値を与えると何らかの値を返す



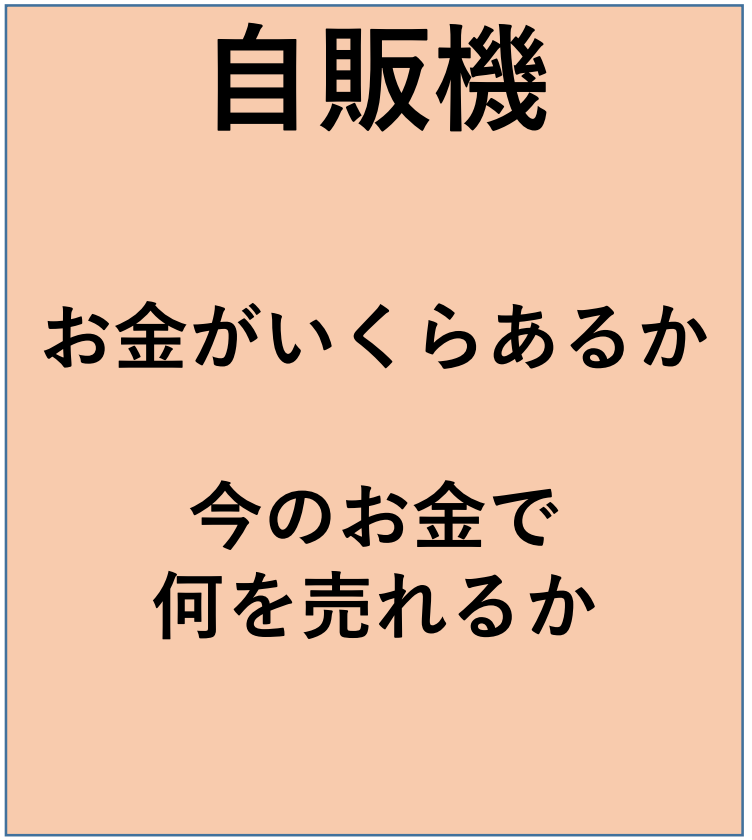
メソッドとは. . .

- 関数　　ある値を与えると何らかの値を返す



メソッドとは. . .

- まとまった処理の塊
  - 値を渡せる
  - 値が貰える



メソッド

メソッドにするメリット

- 何度も同じコードを記述する必要がなくなる
- 処理がまとまっていることでプログラムが見やすくなる

メソッドの使用　～メソッドを呼ぶ～

- メソッドを呼ぶ・・・メソッド(処理の集まり)を使う

メソッド名();　何も値を渡さなくてもいい場合

メソッド名(引数);　値を渡す場合

メソッド名(引数, 引数, ...,引数);

複数値を渡す場合

メソッドの使用　　～使用例～

```
printHello();

printf(“Hello”);

sumPrint(5,10);
```

メソッドの使用　～返り値～

メソッド:getSpeed

引数なし / 返り値あり int型

```
getSpeed();

int speed = getSpeed();
```

speedに返り値が入る

```
System.out.println(getSpeed());
```

返り値が出力される

メソッドの作成

変数宣言

[修飾子]△返り値の型△メソッド名(引数の型△変数){処理}

- 返り値がない場合は void
- 引数がない場合はなにも（）の中に入れない

```
[修飾子]△返り値の型△メソッド名(引数の型△変数, 引数の型△変数, ...){
        //処理
}
```

メソッドの作成

変数宣言

[修飾子]△返り値の型△メソッド名(引数の型△変数){処理}

- 返り値がある場合・・・処理の最後に**return△返り値;** が必要

```
[修飾子]△返り値の型△メソッド名(引数の型△変数, 引数の型△変数, ...) {
        //処理
        return 返り値;
}
```

メソッドの作成

public static 返り値の型△メソッド名(引数の型△変数){処理}

```
public△class△Main{
        public△static△void△main(String[]△args){
                //mainメソッド
        }
        public static void△sub(){
        }
}
```

メソッドの中にメソッド
作ることはできない

メソッドの作成

メソッド名は小文字から (マナー)

数字から始めるとコンパイルエラー
(先頭以外なら可能)

メソッドの作成　～例～

public static 返り値の型△メソッド名(引数の型△変数){処理}

メソッド名:printHello

返り値なし

引数なし

```
public△class△Main{
        public△static△void△main(String[]△args){
        }
        public static void△printHello (){
                //処理
        }
}
```

返り値なし

メソッド名

引数なし

メソッドの作成　～例～

public static 返り値の型△メソッド名(引数の型△変数){処理}

メソッド名:sum
返り値の型:int型
引数の型:　int型
引数の数:　2

```
public△class△Main{
        public△static△void△main(String[]△args){
        }
        public static int△sum (int△a,int△b){
                int△tmp = a+b;
                return△sum;
        }
}
```

返り値int型

引数

値を返す

メソッド　～まとめ～

- メソッドとは処理の集まり・関数

- メソッドの作り方

仮引数

```
public static 返り値の型△メソッド名(引数の型△変数){処理}
```

- メソッドの呼び方

```
メソッド名(引数);
```

実引数

メソッド

- System.out. **println();**

メソッド

- System.out. **println(“Hello world”);**

出力をしてくれるメソッド
引数：”Hello world”

メソッド　　演習問題

(1) **以下の仕様でメソッドを作成**

- 引数:　int型
- 処理:　1からnまでの偶数の和を求めて返す。
　　引数が1以上でないなら0を返す。
- 返り値:　int型

(2)　(1)のメソッドを使って引数が -1の時と5の時の返り値を出力するプログラムを作成

メソッド　　演習問題2

(1)2つのint型の引数n,mをとり、n/mのパーセント値(整数)を
返すメソッドを作成

(2)　2つ整数をキーボードから取得し、(1)で作成した
メソッドを使って「１つ目の整数 / 2つ目の整数」の
パーセント値を出力するプログラムを作成

メソッド　　演習問題3(応用)

国語、数学、英語、理科、社会の点数を順番に入力して、配列tensuuに格納する。

配列tensuuをtensuu[0]から順番にメソッドに与える。メソッドの機能は以下の通り

- **メソッドの機能**

与えられた点数が６０点以上なら文字列“goukaku”、それ以外なら“fugoukaku”を返す。

メソッドから返された値を配列gouhiにgouhi[0]から順番に格納する。

配列tensuuに“fugoukaku”が格納されている科目名を出力する。

```
public class Kadai1{
	public static void main(String[] args){
		int a = sumEven(-1);  //-1をsumEvenの変数nへ移動して処理
		int b = sumEven(5);   //-1をsumEvenの変数nへ移動して処理
					//返り値は変数bに入れる
		System.out.println(a);
		System.out.printf("%d¥n",b);
	}
	    //返り値int型        受け取る変数の宣言
	public static int sumEven(int n){
		//1からnまでの偶数の和を求める
		int sum=0;
		for(int i=1;i<=n;i++){
			if(i%2 == 0) sum = sum + i;
		}
		return sum;//値を返す
	}
}
```