

第14章 メソッドの基本

目次

- メソッドの基本
- メソッドの引数
- メソッドの戻り値

メソッドを使用する

メソッドとは、1つ以上の「処理」を1つの機能としてまとめたもの。
定義したメソッドはプログラム処理中で何度も呼び出すことができる。

メソッドをうまく利用することで...

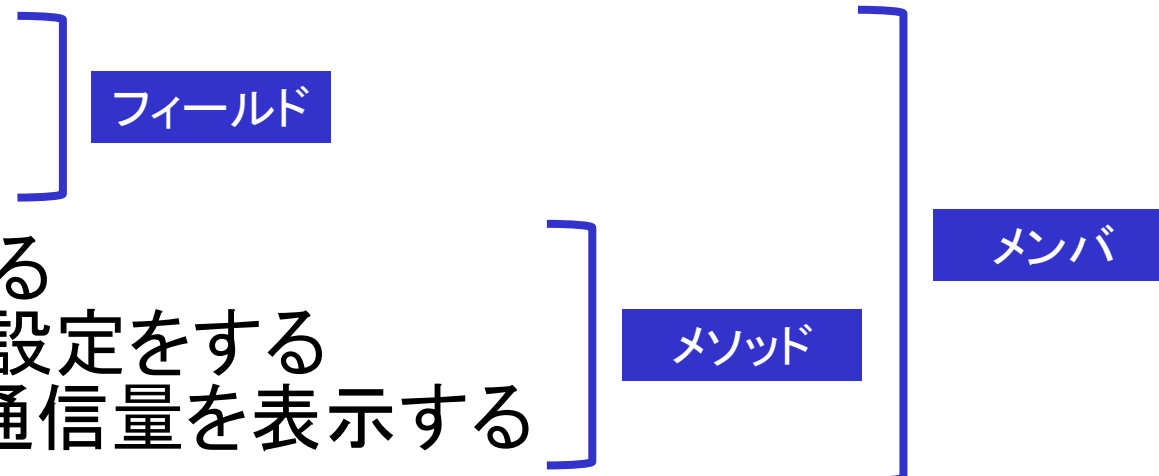
- 効率的にシステム開発を行える。
- 修正や機能改修を行った際の影響を最小限に抑えられる。

メソッドとは、1つ以上の「処理」を1つの機能としてまとめたもの。
メソッドを利用することで、効率よくシステム開発ができます。

メソッドを使用する

メソッドはフィールドと同じくクラスのメンバであり、クラスブロック内に記述する。

```
class 携帯電話{  
    料金;  
    データ通信量;  
    料金設定をする  
    データ通信量設定をする  
    料金とデータ通信量を表示する  
}
```



メソッドを使用する

メソッドは下記構文に沿って記述する。

```
戻り値の型 メソッド名(引数リスト){  
    メソッドが呼び出されたときに実行される処理;  
}
```

メソッドの名前(メソッド名)は変数と同じく識別子のルールに則り命名する。

オブジェクトのメソッド呼び出し

メソッドをクラスに定義すると、生成したオブジェクトからメソッドを呼び出せるようになる。

呼び出されたメソッドは処理を1行目から順番に実行する。
処理が一通り終了すると、呼びだし元に処理が戻る。

オブジェクトのメソッド呼び出し

他クラスからメソッドを呼び出す場合の記述：
オブジェクトを参照している変数名．メソッド名

変数名.メソッド名(引数リスト);

【Sample1401 メソッドを利用する】を作成しましょう



Sample1401のポイント

Phone1401クラス内にshow()メソッドを定義している。
show()メソッドのブロック内にはフィールドfeeとdataの値を
標準出力する処理が記述されている。
このように、メソッド内では同じクラスのフィールドを
呼び出すことができる。

```
void show() {  
    System.out.println("携帯電話の料金は" + fee + "円です。");  
    System.out.println("データ通信量は" + data + "GBです。");  
}
```

Sample1401のポイント

Sample1401クラスのmain()メソッド内では、Phone1401クラスのオブジェクトを生成し、オブジェクトの各フィールドに値を代入した後にshow()メソッドを呼び出している。
その結果、フィールドfeeとdataの値が標準出力される。

```
// 携帯電話クラスのオブジェクトを生成
Phone1401 phone = new Phone1401();
// フィールドに値を代入
phone.fee = 5000;
phone.data = 2.0;
// 料金とデータ通信量を表示
phone.show();
```

【Sample1402 他のメソッドから呼び出す】 を作成しましょう

同じクラス内に定義された他のメソッドから、
メソッドを呼び出すこともできる。

Let's try!



Sample1402のポイント

Phone1402クラスのオブジェクトを生成して、そのオブジェクトのshow()を呼び出している。show()メソッドのブロック内の1行目ではmessage()メソッドを呼び出しているため、まずmessage()メソッドの処理が実行され、下記の文言が出力される。

これから携帯電話の情報を表示します。

Sample1402のポイント

message()メソッドの処理が完了すると、
呼出し元であるshow()メソッドの処理に戻る。
そして、message()メソッドの呼び出し処理の次に
記述された処理に移り、下記2行の文言が出力される。

携帯電話の料金は5000円です。
データ通信量は2.0GBです。

メソッドの引数


引数とは、呼び出し先のメソッドに渡す値のこと。
以下の下線部分が引数にあたる記述。

```
void searchName(String name) {  
    System.out.println("アドレス帳から" + name + "さん  
    を検索します。");  
}
```

メソッドの引数

引数を利用する場合は、引数を受け取るための型と変数を()内に指定する。

```
void searchName(String name) {  
    ...  
}
```



仮引数

仮引数とは、引数を受け取るための変数のこと。
仮引数で受け取った値をメソッド内で利用する。
サンプルコードでは、searchName()メソッド内で
仮引数「name」の値を画面出力するという処理が記述されている。

```
System.out.println("アドレス帳から" + name + "さんを検  
索します。");
```


実引数

実引数とは、呼び出し元から渡されて、
仮引数が受け取る実際の値のこと。

実引数を渡す場合は、メソッドの「()」内に実引数を指定する。
その際、必ず仮引数と同じ型の値を指定する。
サンプルコードではsearchName()メソッドに渡す
実引数として田中を指定している。

```
phone.searchName("田中");
```

メソッドに渡す値を「実引数」といいます。
実引数を受け取る変数を「仮引数」といいます。

【Sample1403 引数をもつメソッドを呼び出す】 を作成しましょう

Let's try!



Sample1403のポイント

Phone1403クラスのsearchName()メソッド、call()メソッドは受け取った引数を画面出力するメソッドである。どちらのメソッドにも仮引数が指定されている。

```
void searchName(String n) {  
    System.out.println("アドレス帳から" + n + "さんを  
        検索します。");  
}
```

```
void call(int pn) {  
    System.out.println(pn + "へ電話をかけます。");  
}
```

Sample1403のポイント

引数を持つメソッドにJavadocコメントを記述する場合、
「@param」というJavadocタグを使用して引数の情報を明記する。
「@param」に続けて、「仮引数名」と「仮引数に代入される
値の意味」を記述する。

```
/** 名前を検索  
 * @param n 名前  
 */
```

Sample1403のポイント

main()メソッド内では、searchName()メソッドを呼び出す際に値「田中」「鈴木」「佐藤」を実引数として渡している。

また、call()メソッドには

「1234567890」「1122223333」という実引数を渡している。

```
phone.searchName("田中");  
phone.searchName("鈴木");  
phone.searchName("佐藤");
```

```
phone.call(1234567890);  
phone.call("1122223333");
```

Sample1403のポイント

下記のサンプルコードのように、
実引数として変数を指定することもできる。

```
String name = "田中";  
int phoneNumber = 1234567890;  
  
phone.searchName(name);  
phone.call(phoneNumber);
```

複数の引数を持つメソッド

メソッドには引数を複数個持たせることができる。
その場合、仮引数と仮引数の間を「,(カンマ)」で区切る。

```
void entryAddressBook(String n, int pn) {  
    ...  
}
```

メソッドには複数の引数を定義することができます。

複数の引数を持つメソッド

複数の引数を持つメソッドを呼び出す際は、
実引数を「,(カンマ)」で区切って指定する。
実引数を指定するときは、仮引数と同じ型、順番で指定する。
引数の型が合わない場合、コンパイルエラーが発生する。

```
phone.entryAddressBook("田中", 1234567890);
```


複数の引数を持つメソッド

仮引数と異なる数の実引数を指定した場合も、コンパイルエラーが発生する。

```
// 誤ったメソッドの呼びだし  
phone.entryAddressBook("田中");
```

【Sample1404 複数の引数を持つメソッド】 を作成しましょう

Let's try!



Sample1404のポイント

Phone1404クラスのentryAddressBook()メソッドでは、
第1引数(1つ目の引数)にString型、
第2引数(2つ目の引数)にint型の仮引数が定義されている。

```
void entryAddressBook(String n, int pn) {  
    System.out.println("アドレス帳に名前:" + n + "さ  
        さん、電話番号:" + pn + "を登録します。");  
}
```

Sample1404のポイント

Sample1404クラスのmain()メソッドでは、Phone1404クラスのオブジェクトからentryAddressBook ()メソッドを呼び出している。その際、「()」内にあらかじめ用意したString型、int型の変数を指定している。

このように記述することで、各変数の値が実引数として、仮引数に渡される。

```
String name = "田中";  
int phoneNumber = 1234567890;  
// 複数の引数を持つメソッドを呼び出す  
phone.entryAddressBook(name, phoneNumber);
```

【Sample1405 参照型を引数に持つメソッド】 を作成しましょう

引数には、intやbooleanなどの基本型だけでなく、
配列やオブジェクトといった参照型を指定することもできる。

Let's try!



Sample1405のポイント

Human1405クラスには、buyPhone()メソッドが定義されている。
このメソッドでは、引数として受け取ったPhone1405クラスの
フィールドphoneNameを出力している。

```
void buyPhone(Phone1405 p) {  
    System.out.println(p.phoneName + "を購入しまし  
    た。");  
}
```

Sample1405のポイント

Sample1405クラスのmain()メソッドでは、
Phone1405クラスのオブジェクトを2つ作成し、
それぞれのオブジェクトのフィールドに値を代入している。

```
// 1つ目の携帯電話クラスのオブジェクトを生成
Phone1405 phone1 = new Phone1405();
phone1.phoneName = "ApplePhone";

// 2つ目の携帯電話クラスのオブジェクトを生成
Phone1405 phone2 = new Phone1405();
phone2.phoneName = "AndroidPhone";
```

Sample1405のポイント

その後、Human1405クラスのbuyPhone()メソッドを2回呼び出している。

その際、作成した2つのPhone1405クラスのオブジェクトを実引数として渡している。

```
// 参照型を引数に持つメソッドを呼び出す  
human.buyPhone(phone1);  
human.buyPhone(phone2);
```


引数のないメソッドの定義

引数のないメソッドを定義する場合は、
「()」内には何も記述しない。

```
void show() {  
    System.out.println(“携帯電話の料金は” + fee +  
        “円です。”);  
    System.out.println(“データ通信量は” + data +  
        “GBです。”);  
}
```

引数のないメソッドの定義

引数のないメソッドを呼び出す際も、「()」内には何も記述しない。

引数を指定しないメソッドを定義すると、
値を渡さないで呼び出せます。

```
phone.show();
```

戻り値の仕組み

戻り値とは、メソッドから返される値のこと。

戻り値を返すには、メソッドの定義の中に
「戻り値の型」を指定する。

そして、メソッドのブロック内でreturn文を記述し、
返したい値を指定する。

```
戻り値の型 メソッド名（引数リスト） {  
    . . .  
    return 戻り値;  
}
```



戻り値の仕組み

- ・戻り値は1つの値(式)しか指定できない。
→もし複数の値を返したい場合は、
それらの値を一旦配列の中に代入し、
その配列を戻り値として返すなどの工夫が必要。

戻り値を利用すると、呼び出し元に情報を返すことができます。

【Sample1406 戻り値を持つメソッド】を作成しましょう

Let's try!



Sample1406のポイント

sayFullName()メソッドは引数として受け取った2つの文字の連結結果を、calcBmi()メソッドは受け取った身長、体重から算出したBMIの計算結果を戻り値として返す。

```
double calcBmi(double weight, double height) {  
    double bmi = weight / ((height / 100) *  
        (height / 100));  
    return bmi;  
}
```

```
String sayFullName(String sei, String mei) {  
    String fullName = sei + mei;  
    return fullName;  
}
```

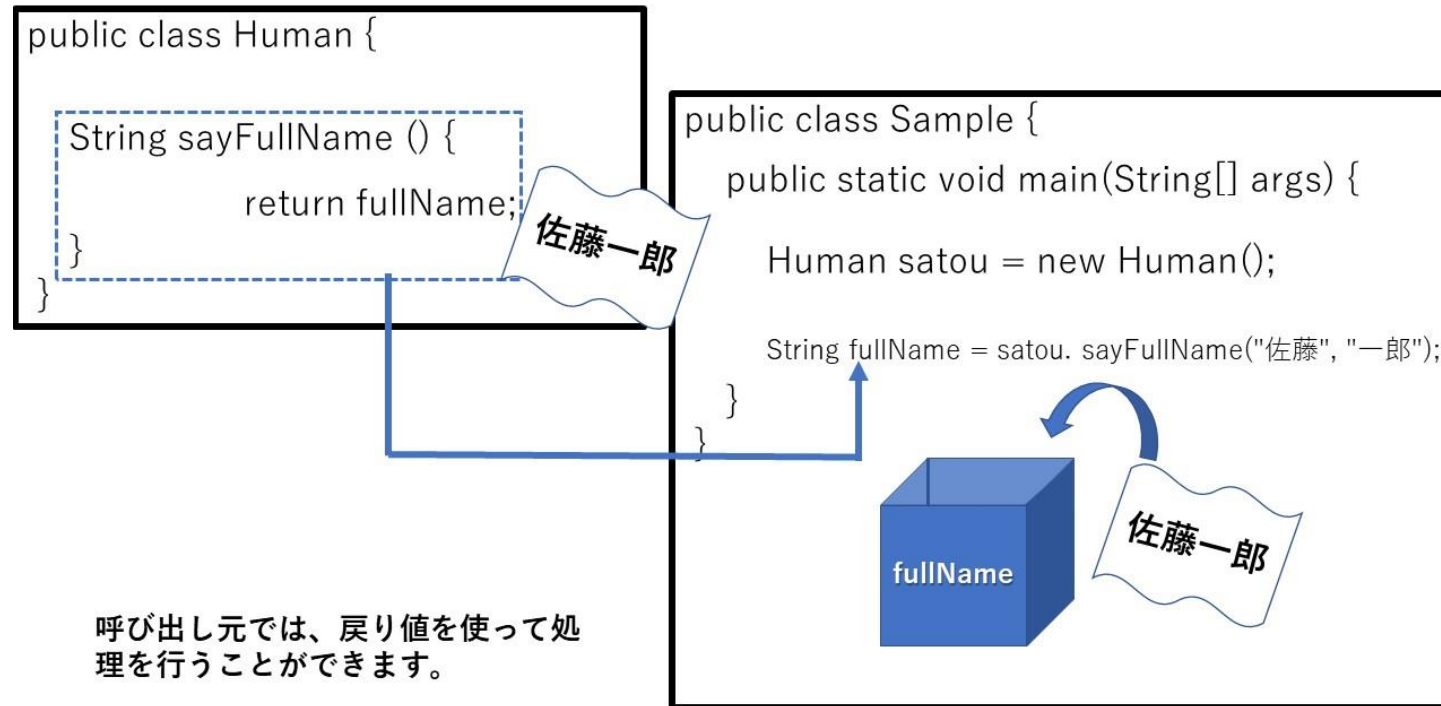
Sample1406のポイント

Sample1406クラスのmain()メソッドでは、sayFullName()メソッドとcalcBmi()メソッドを呼び出している。戻り値を以降の処理で利用したい場合、メソッドの呼び出し結果を変数に代入するよう記述する。

```
//メソッドを呼びだし、戻り値を変数に代入  
String fullName = satou.sayFullName("佐藤", "一郎");  
double bmi = satou.calcBmi(70.0, 175.0);
```

Sample1406のポイント

戻り値の受け渡しのイメージ



【Sample1407 参照型を引数に持つメソッド】を作成しましょう

戻り値には、intやbooleanなどの基本型だけでなく、配列やオブジェクトといった参照型を指定することもできる。

Let's try!



Sample1407のポイント

Human1407クラスには、
tellHobbies()メソッドが定義されている。
このメソッドは、String型の配列hobbiesの値を
戻り値として呼び出し元に返す。

```
String[] tellHobbies() {  
    String[] hobbies = {"ランニング", "映画鑑賞", "読書"};  
    return hobbies;  
}
```

Sample1407のポイント

Sample1407クラスのmain()メソッドでは、
tellHobbies()メソッドを呼び出し、
戻り値を変数showHobbiesに代入している。
その後for文を利用し、変数showHobbiesの各要素を出力している。

```
// メソッドを呼び出し、戻り値を変数に代入  
String[] showHobbies = satou.tellHobbies();  
  
// 変数の中身を繰り返して表示  
for (int i = 0; i < showHobbies.length; i++) {  
    System.out.println((i + 1) + "つ目の趣味  
    は" + showHobbies[i] + "です。");  
}
```

戻り値を持たないメソッド

メソッドが戻り値を持たないことを示すには、
戻り値の型の部分に「void」と記述する。
voidは「なにもない」という意味を持つ。

```
public void show() {
```

戻り値の型の部分にvoidと記述

戻り値のないメソッドにはvoidを指定します。

章のまとめ

- メソッドを定義して、一定の処理をまとめることができます。
- メソッドを呼び出すと定義しておいた処理が行われます。
- 引数を使って、メソッドに値を渡すことができます。
- メソッドの定義内で値を受け取る変数を、仮引数と呼びます。
- メソッドを呼び出すときに渡す値を実引数と呼びます。
- 戻り値を使うと、呼び出し元に情報を返すことができます。