

第15章 メンバへのアクセス制限

目次

- メンバへのアクセス制限
- アクセス修飾子の利用

アクセス修飾子とは

クラスやメンバに付与できるキーワード。
他のクラス、メソッドからのアクセスを許可するかを指定できる。

アクセス修飾子 フィールド宣言;

アクセス修飾子 メソッド宣言{…}

private修飾子

private修飾子が付与されたクラス、メンバは、他のクラスからアクセスできなくなる。
想定外の値を代入される危険性がなくなる。

private修飾子が付与されたメンバ:
→privateメンバ

```
public class Phone {  
    /** 料金 */  
    private int fee;  
    /** データ通信量 */  
    private double data;  
}
```

private修飾子

他のクラスからprivate修飾子が付与されたメンバに直接アクセスしようとする、コンパイルエラーが発生する。

```
phone.fee = -10000;  
phone.data = 9999.99;
```

public修飾子

public修飾子が付与されたクラス、メンバは、他のすべてのクラス、メソッドからアクセス可能となる。

public修飾子が付与されたメンバ:

→publicメンバ

```
public int getFee() {  
    return fee;  
}  
public void setFee(int f) {  
    fee = f;  
}
```

public修飾子

private修飾子が付与されたメンバを操作したい場合は、
publicメソッドを経由する。

```
private int fee;
```

```
public int getFee() {  
    return fee;  
}
```

privateメンバは他のクラスからアクセスできません。
publicメンバは他のすべてのクラスからアクセスできます。

【Sample1501 メンバへのアクセスを制限する】 を作成しましょう



Sample1501のポイント

Phone1501クラスのフィールドfeeとdouble:

private修飾子。他のクラスからはアクセスできない。

setFee()メソッドとsetData()メソッド:

public修飾子。他のクラスからアクセス可能。

```
private int fee;  
private double data;
```

```
public void setFee(int f) {  
}  
  
public void setData(double d) {  
}
```

Sample1501のポイント

setFee()メソッドとsetData()メソッドでは
引数の値が想定通りの値であるかを
チェックするための条件分岐が記述されている。

```
public void setFee(int f) {  
    if (0 <= f) {  
        fee = f;  
        System.out.println("料金を" + fee + "円にしました。");  
    } else {  
        System.out.println(f + "は正しい値ではありません。");  
    }  
}
```

条件を満たす場合:フィールドに代入。

満たさない場合:想定外の値と見なしてエラーメッセージを出力する。

Sample1501のポイント

privateとpublicを利用することで、
想定外の値の受け渡しを防ぎ、
安全に処理が実行されるプログラムが作成できる。

privateとpublicを利用すれば、
想定外の値の受け渡しを防ぐことができます。

アクセス修飾子の種類

修飾子名	アクセスを許可する範囲	記述できる場所
private	同一クラスのみ	フィールド、メソッド、コンストラクタ
指定なし	同一パッケージ内のクラス	フィールド、メソッド、コンストラクタ、クラス
protected	同一パッケージ内のクラスか、 自身を継承したクラス	フィールド、メソッド、コンストラクタ
public	すべてのクラス	フィールド、メソッド、コンストラクタ、クラス

カプセル化

privateメンバへの操作はpublicメンバを経由して行うようにアクセス制限をかけること。

クラス的设计者が、メンバをprivateメンバとpublicメンバに分けておくことで、他の人が既存のクラスを誤った方法で利用してしまう危険性を回避できる。

アクセサ

フィールドへの値の代入、および値の取得に機能が特化したメソッドのこと。

カプセル化を実現する際に必要な要素。

フィールドをオブジェクト内部に隠蔽し、外部から直接参照させない場合に、代わりに外部からアクセスさせるメソッドとしての役割を持つ。

getterメソッド

アクセサのうち、フィールドの値を取得するためのメソッド。

```
public フィールドの型 メソッド名() {  
    return フィールド名;  
}
```

setterメソッド

アクセサのうち、フィールドに値を代入(設定)するためのメソッド。

```
public void メソッド名(型 仮引数名) {  
    this.フィールド名 = 仮引数名;  
}
```


getterとsetterの命名規約

getter:「get」の後ろに頭文字を大文字にしたフィールド名

setter:「set」の後ろに頭文字を大文字にしたフィールド名

(例)private int num のgetter、setter

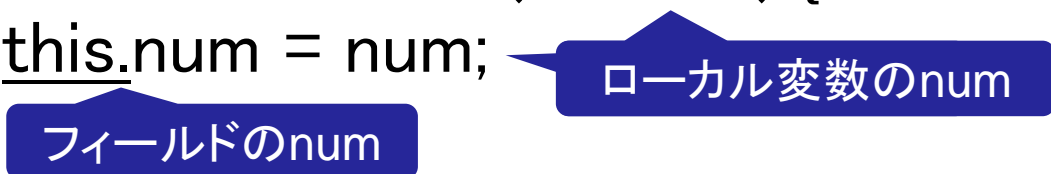
```
public int getNum() {  
    return num;  
}  
public void setNum(int num) {  
    this.num = num;  
}
```

this.の付与

Javaでは、ローカル変数と同名のフィールドが存在する場合、ローカル変数が優先される。

「その変数はオブジェクトに属する(オブジェクトのフィールドである)」と明示するには、フィールドの先頭にthis.を付与する。

```
public void setNum(int num) {  
    this.num = num;  
}
```



【Sample1502 getterとsetter】を作成しましょう

Let's try!



Sample1502のポイント

Human1502クラスにはフィールドnameとgender、
そして各フィールドのgetterとsetterが定義されている。

```
public void setName(String name) {  
    this.name = name;  
}  
  
public void setGender(String gender) {  
    this.gender = gender;  
}
```

Sample1502のポイント

Smample1502クラスのmain()メソッド内では、生成したオブジェクトからsetterを呼び出して、フィールドに値を代入している。

```
human.setName("田中太郎");  
human.setGender("男性");
```

Sample1502のポイント

getterを使用して各フィールドの値を取得し、標準出力している。

```
System.out.println(human.getName());  
System.out.println(human.getGender());
```

章のまとめ

- privateメンバには、クラス外からアクセスできません。
- publicメンバには、クラス外からアクセスできます。
- フィールドとメソッドを1つにまとめ、メンバを保護する機能をカプセル化と呼びます。
- getterはprivateがついたフィールドを他のクラスで呼び出したいときに使用します。
- setterはprivateがついたフィールドに他のクラスから値を代入したいときに使用します。