# 上級プログラミング１（第１回）

工学部　情報工学科
木村昌臣

# 教科書

- オブジェクト指向プログラミング

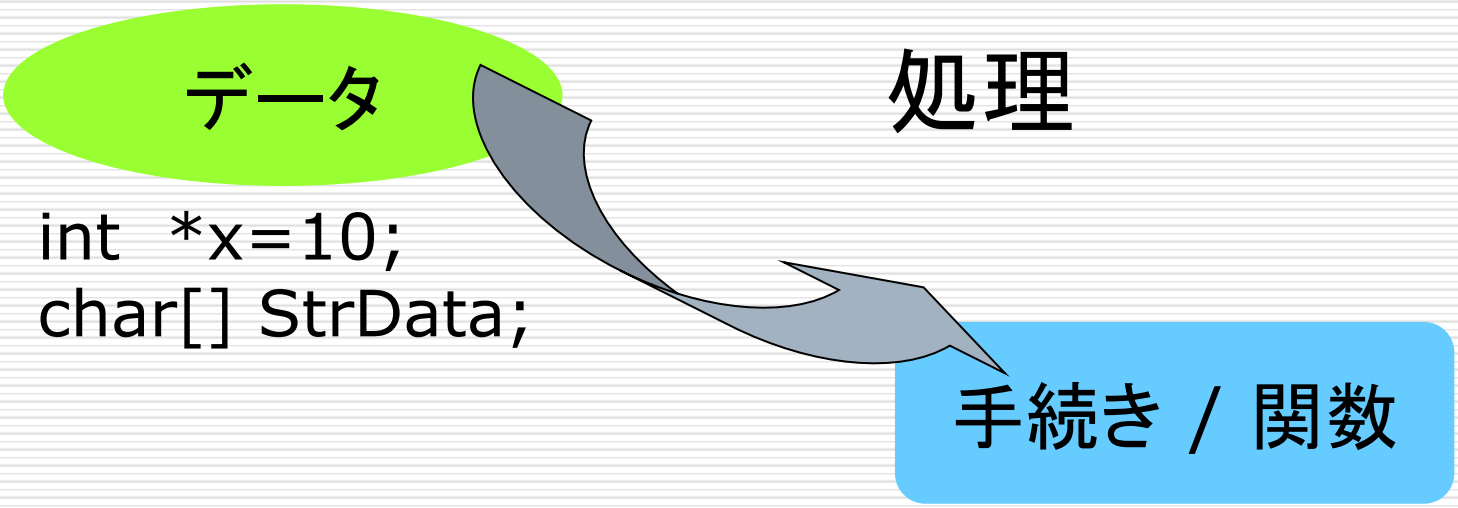

- Java8問題集 理解を深める500問
  - 小テストはここから

# 今日のテーマ

- オブジェクト指向プログラミングとは
- オブジェクト指向言語
- Java概説
- Javaプログラミング入門(1)

# 今まで勉強してきたプログラミング手法

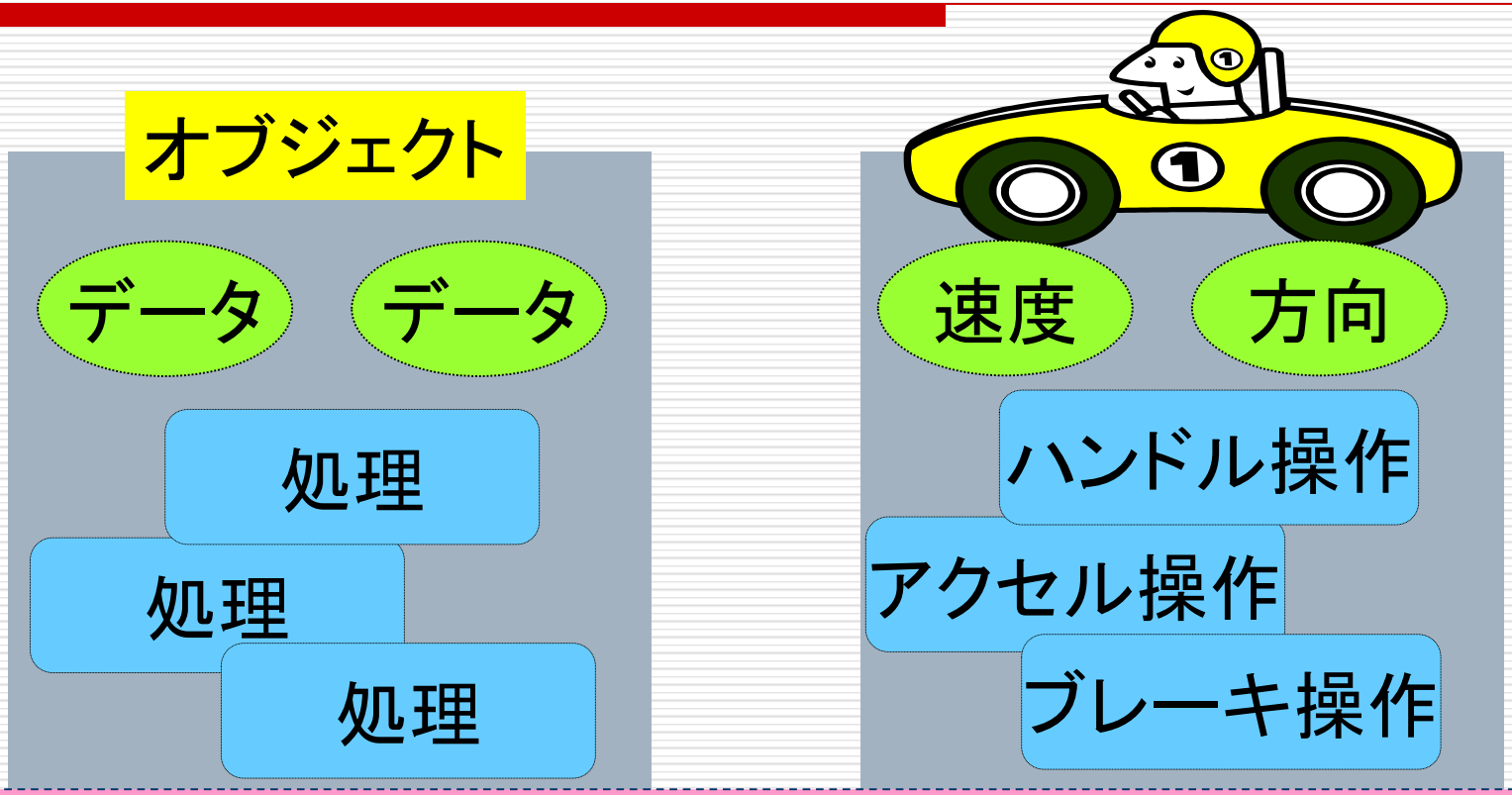
- 構造化プログラミング



# 構造化プログラミングの問題点

- データと関数が独立に存在
  - 関数とデータの関係（どの関数にどのデータを渡すべきか）をプログラマが全て知っておかねばならない
- 機能中心主義
  - 実現すべき機能は要望により変化しやすく、プログラム全体が変更しなければならない可能性がある
  - 実際には、データのほうが変化しにくい

# オブジェクト指向プログラミング




データ構造と処理をまとめた
**オブジェクト**を操作することにより
プログラムを構成

# オブジェクト指向プログラミング

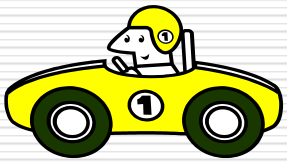
- オブジェクト＝データ＋処理
  - データとそれに対する処理をワンセットに
  - プログラムはオブジェクト（物）への操作として実現

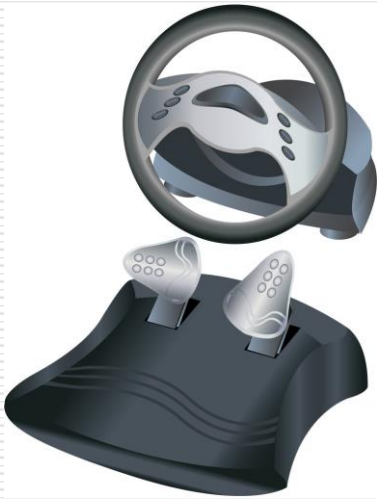
    右折するプログラム＝

    車（オブジェクト）のアクセルを踏む（操作）

    ＋

    車（オブジェクト）のハンドルを右に切る（操作）





  - 操作はデータを変更することがある
- クラス＝オブジェクトの定義

# 代表的なオブジェクト指向言語(1)

- Smalltalk
  - 最初のオブジェクト指向言語
- C++
  - C言語の拡張言語
  - 多重継承
- Java
  - C言語をベースにインターネットに適した言語として開発
  - Unix, Windowsなど様々なプラットフォーム上でプログラムが動作(Write Once, Run Anywhere)
  - 単一継承

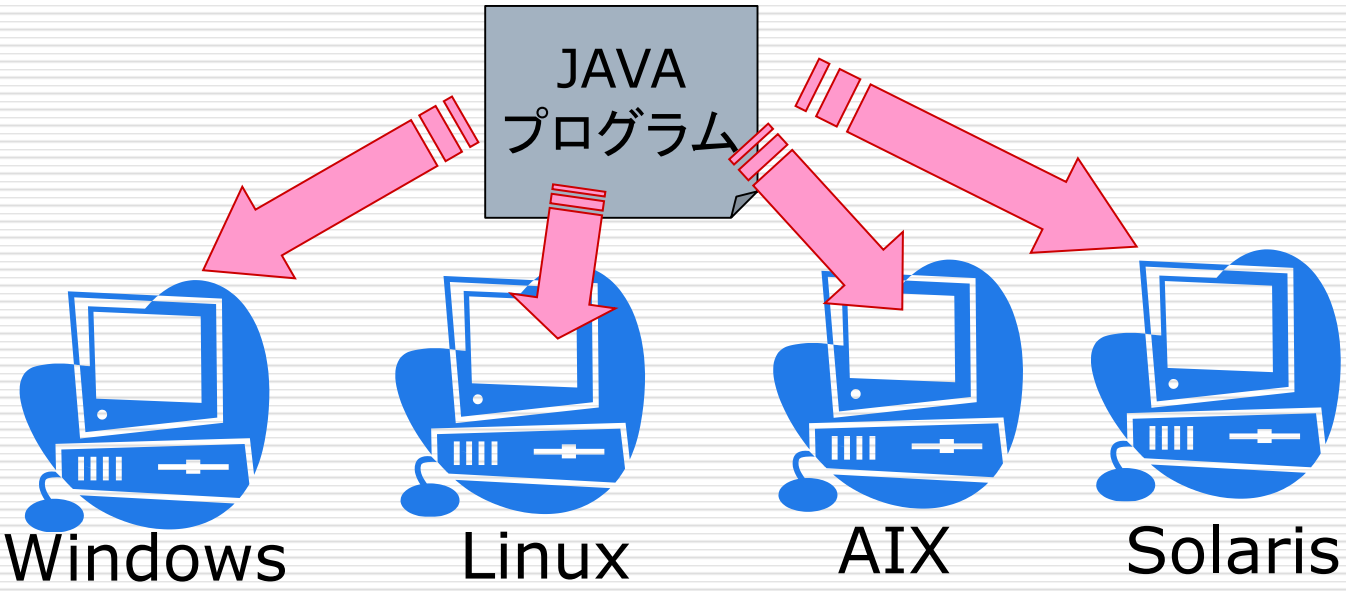
# 代表的なオブジェクト指向言語(2)

- Ruby
  - オブジェクト指向スクリプト言語
- Visual Basic, Visual C++
  - Microsoftのアプリケーション開発言語
- C#
  - MicrosoftのJava likeなアプリケーション開発言語

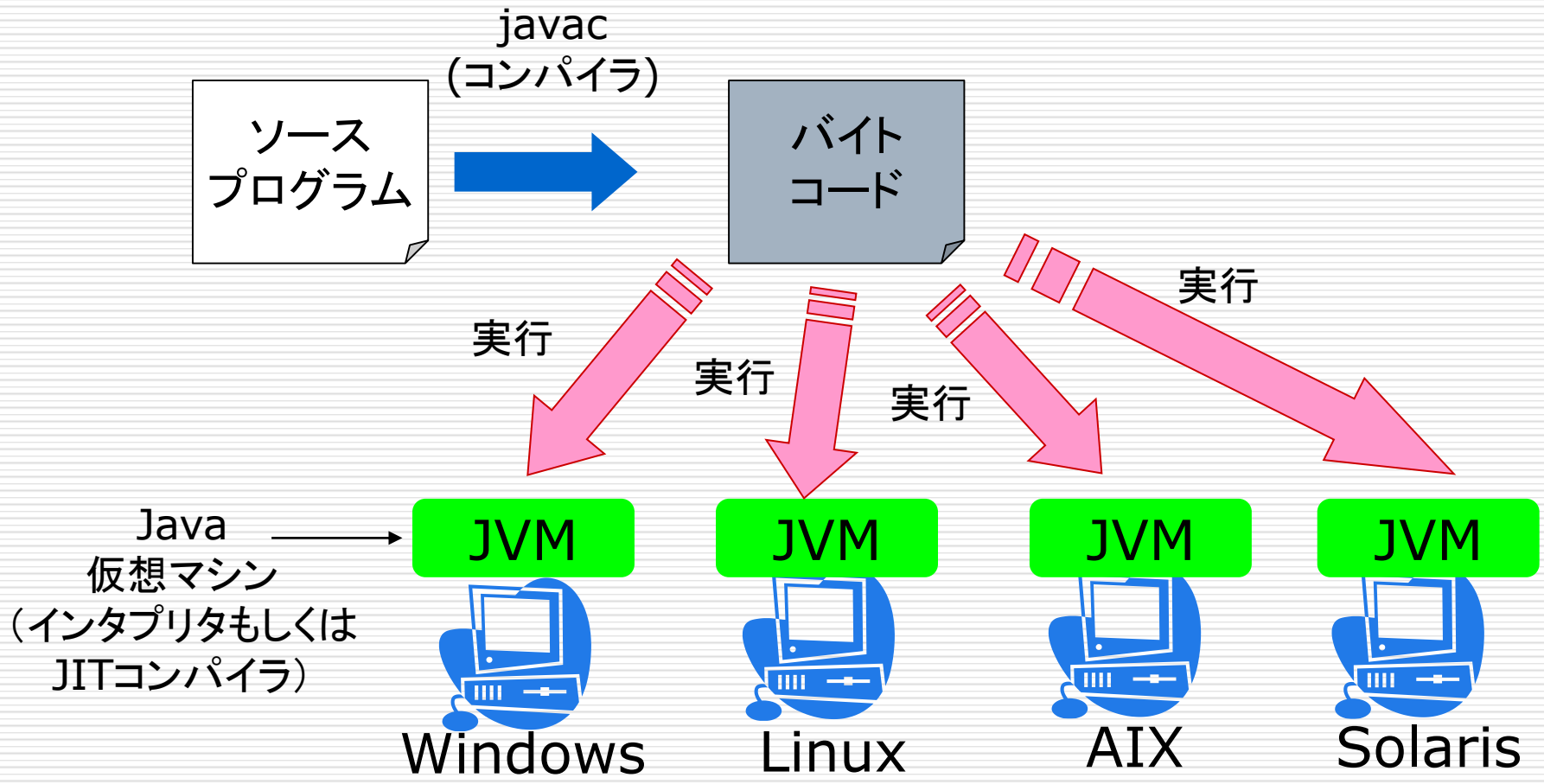
# Java概説

# Javaとは

- 1995年にSun Microsystems(現Oracle)が発表
- 再コンパイルなしに様々なプラットフォーム上でプログラムが動作(Write Once, Run Anywhere)



# Javaプログラム動作の仕組み



# JavaのEdition(用途による)

- Java SE (Standard Edition)
  - 標準版(通常はこれを使う)
- Java EE (Enterprise Edition)
  - エンタープライズシステム向き
  - JavaBeans開発
  - Eclipse foundationへ移管されJakarta EEへ
- Java ME (Micro Edition)
  - 組み込み機器用

# Javaプログラムの開発・実行環境

- JRE (Java Runtime Environment)
  - Java実行環境のみ
  - これだけではプログラムをコンパイルできない
- JDK(Java Developers Kit)
  - Java開発環境(コンパイラなど)
  - JREを含む

# Javaプログラムの形態(1)

- アプリケーション(application)
  - 一番基本的な形態
  - C言語のプログラムのようにmainメソッド(≒関数)が実行されて動く
  - クライアントで実行
- アプレット(applet)
  - Webブラウザ上で実行される形態
  - 起動時間や実行時間が遅く、現在あまり用いられていない
  - クライアントで実行

# Javaプログラムの形態(2)

- サーブレット(servlet)
  - Webアプリケーションサーバー上で実行
  - 入出力はクライアントのブラウザを利用
- JSP (Java Server Page)
  - HTMLにJavaプログラムを埋め込んで動的なWebページを生成する仕組み
  - Webアプリケーションサーバー上で実行
  - 実行時にサーブレットに変換
- Bean
  - 特定の処理を部品としてあらかじめ用意

# この授業では

- もっぱらJavaアプリケーションを扱う
- 上級プログラミング2ではサーブレット、JSPも扱う

# Java プログラミング入門(1)

# まずは簡単な例

```
class SampleClass01{
        static int i=10;
        public static void main(String[] args){
                add10();
                System.out.println(“i=”+i);
        }
        public static void add10(){
                i+=10;
        }
}
```

# プログラムの構成(1)

プログラムは(1つ以上の)クラスとして構成される

```
class SampleClass01{
        static int   i=10;
        public static void main(String[] args){
                add10();  System.out.println("i="+i);
        }
        public static void add10(){
                i+=10;
        }
}
```

# プログラムの構成(2)

クラス≒構造体メンバ(フィールド)
＋メンバを操作する関数(メソッド)

```
class SampleClass01{
    static int   i=10;
    public static void main
            add10();  System.out.println("i="+i);
    }
    public static void add10(){
        i+=10;
    }
}
```

フィールド

メソッド

# プログラムの構成(3:アプリケーションの場合)

クラスのmainメソッドが実行される
(C言語のmain関数のようなもの)

```
class SampleClass01{
        static int   i=10;
        public static void main(String[] args){
            add10(); System.out.println("i="+i);
        }
        public static void add10(){
                i+=10;
        }
}
```

※見慣れないpublicやstaticなどは後の回で説明します

# 実行例

```
class SampleClass01{
        static int i=10;
        public static void main(String[] args){
                add10();   System.out.println("i="+i);    }
        public static void add10(){
                i+=10;
        }
}
```

```
C:¥>  javac  SampleClass01.java
C:¥>  java SampleClass01
        i=20
C:¥>
```

ファイル
SampleClass01.class
が生成される

# C言語と似ているところ

- 変数の型（基本データ型）
  - 整数型: byte, short, int, long
  - 実数型: float, double
  - 文字型: char　　など
    - ただし、文字列はStringクラスのオブジェクト(物)として扱うので注意
- 演算子
- 制御文
  - 基本的に、if文、for文、while文などは書式が同じ

# 例2

```
class Account{
        public int amt=1000;
        public void save(int j){ amt=amt+j; }
        public int show(){ return amt;}
}
```

```
class SampleClass02{
        public static void main(String[] args){
                Account myAccount=new Account();
                myAccount.save(2000);
                int j=myAccount.show();
                System.out.println("預金:"+j);
        }
}
```

# 例2：口座クラス

```
class Account{
        public int amt=1000;
        public void save(int j){ amt=amt+j; }
        public int show(){ return amt;}
}
```

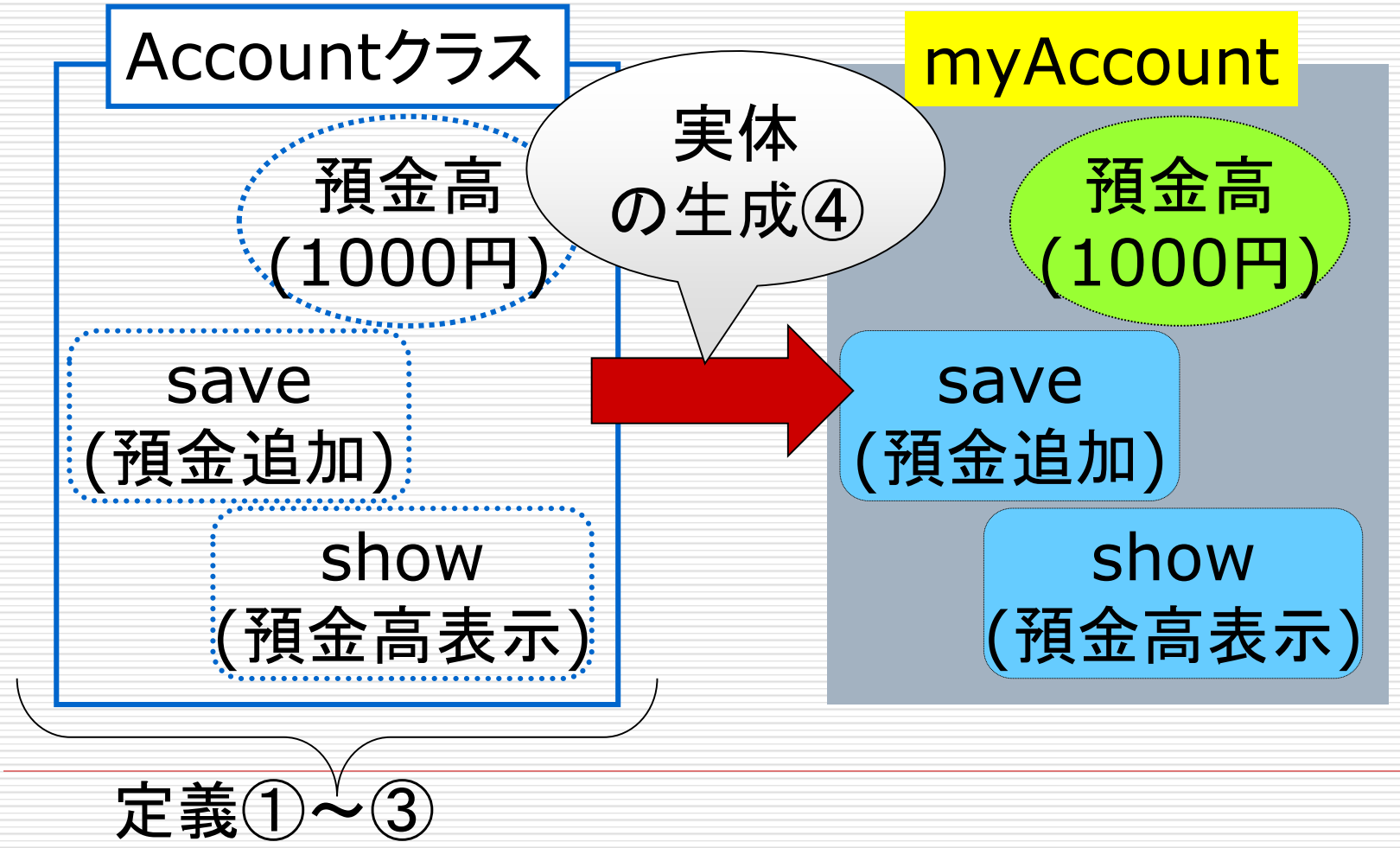
① 預金高を表す変数を定義。1000円とする
② saveメソッドが呼ばれると、引数の額だけ加算する
③ showメソッドが呼ばれると預金高が返される

# 例2: SampleClass02クラス

```
class SampleClass02{
        public static void main(String[] args){
            Account myAccount=new Account();・・・④
            myAccount.save(2000);   ・・・⑤
            int j=myAccount.show();  ・・・⑥
             System.out.println(“預金:”+j);     ・・・⑦
        }
}
```

④ 口座オブジェクト(口座の実体)を作り、初期化
⑤ 2000円預ける　　⑥ 現在の預金高をjに代入
⑦ jの値を表示する

# 例2のイメージ(1)


Accountクラス
預金高
(1000円)
save
(預金追加)
show
(預金高表示)
実体
の生成④
myAccount
預金高
(1000円)
save
(預金追加)
show
(預金高表示)
定義①～③

# 例2のイメージ(2)


表示⑦

預金:j=3000