例 1

class は「フィールド」と「メソッド」を保持する。

```
enum Hand {ROCK, PAPER, SCISSORS}
```

java/Hand.java

```
public class A {
    boolean b;
    int i;
    Hand h;
    public void print(){
        System.out.println(b+" "+i+" "+h);
    }
}
```

java/A.java

```
public class Test0 {
    public static void main(String[] s){
        A x = new A();
        x.b = true;
        x.i = 30;
        x.h = Hand.ROCK;

        A y = new A();
        y.b = false;
        y.i = 40;
        y.h = Hand.PAPER;

        x.print();
        y.print();
    }
}
```

java/Test0.java

例 2

class は new するときにインスタンスが作られる (コンストラクタが呼ばれる)。インスタンスごとに異なる値を保持できる。

```java
public class B {
    boolean b;
    int i, j;
    Hand h;
    public B(){ // コンストラクタ(new時にのみ呼ばれる)
        this.b = true;
        this.i = 10;
        this.j = 20;
        this.h = Hand.SCISSORS;
    }
    public void print(){
        System.out.println(b+" "+i+" "+j+" "+h);
    }
}
```

java/B.java

```java
public class Test1 {
    public static void main(String[] s){
        B x = new B();

        B y = new B();
        y.b = false;
        y.i = 20;

        x.print();
        y.print();
    }
}
```

java/Test1.java

例 3

class はもちろん別のクラスのインスタンスも保持できる。また、クラスは「型」のように機能するので配列にしたりもできる。ただし、実体 (インスタンス) はどこかで作る必要がある。

```java
public class C {
    A a;
    B b;
    public C(){ // コンストラクタ(new時にのみ呼ばれる)
        this.a = new A();
        this.b = new B();
    }
}
```

java/C.java

```java
public class Test2 {
    public static void main(String[] s){
        // 単体のインスタンス
        C obj = new C();

        obj.a.b = true;
        obj.a.i = 11;
        obj.a.h = Hand.ROCK;

        obj.a.print();
        obj.b.print();

        // インスタンスを配列に
        // 注意: Javaの配列のnewの構文
        //       クラス名[] 変数名 = new クラス名[配列の要素数]
        C[] setC = new C[3];

        for(int i=0; i<setC.length; i++){
            setC[i] = new C(); // 各々の要素にインスタンスを作って代入
        }

        for(int i=0; i<setC.length; i++){
            setC[i].b.i = 3*i;
            setC[i].b.j = 4*i;
        }

        for(int i=0; i<setC.length; i++){
            System.out.println("Array element "+i+"th");
            setC[i].b.print();
        }

    }
}
```

java/Test2.java

例 4

各インスタンスは各々独立に値を保持できるので、たとえば、異なるインスタンスを保持していれば同じメソッドで異なる結果を得る。

```
public class D {
    A a;
    B b;
    void set(A a, B b){
        this.a = a;
        this.b = b;
    }
    void run(){
        this.a.b = false;
        this.a.i = 1;
        this.a.h = Hand.PAPER;

        this.a.print();
        this.b.print();
    }
}
```

java/D.java

```
public class Test3 {
    public static void main(String[] s){
        A a;
        B b1, b2;

        a = new A();
        b1 = new B();
        b2 = new B();
        b2.i = 99;
        b2.h = Hand.ROCK;

        D obj = new D();

        obj.set(a,b1);
        obj.run();

        obj.set(a,b2);
        obj.run();
    }
}
```

java/Test3.java

例 5

- 「クラス」を作る ⇔ 部品の「設計図 (雛形)」を作る
- あるクラスの「インスタンス」を 1 こ作る ⇔ 実際に雛形にそって部品を 1 こ作る
- 各々の「インスタンス」は独立に操作でき、独立に情報を保持する ⇔ 「部品」は独立に操作でき、独立に情報を保持できる。「ネジ型番 0001」を 2 こつくって、片方を赤で塗って、もう片方を青で塗って、ということができる。

```java
public class E {
    // 変数を「カプセル化」
    private boolean flag;
    private int counter1, counter2;
    // コンストラクタ
    public E(){
        this.flag = true;
        this.counter1 = 0;
        this.counter2 = 0;
    }
    // メソッド
    public void call(){
        System.out.println("call");
        if(flag)
            this.counter1++;
        else
            this.counter2++;
    }
    public void lost(String msg){
        System.out.println(msg);
        this.flag = false;
    }
    public void print(){
        System.out.println("counter 1:"+counter1+" 2:"+counter2);
    }
}
```

java/E.java

```java
public class Test4 {
    public static void main(String[] s){
        // objのprivate変数に直接アクセスできない/しないが
        // 部品objを使って機能が実現できていることに注意

        E obj;

        obj = new E();

        for(int i=0; i<7; i++) obj.call();
        obj.print();

        obj.lost("Change mode");

        for(int i=0; i<3; i++) obj.call();
        obj.print();
    }
}
```

java/Test4.java

例 6

負けるまでランダム、一度負けたら以降はグーを出し続けるプレイヤ。

```java
public class RandomJankenPlayer {
    private boolean isLose;
    public RandomJankenPlayer(){
        this.isLose = false;
    }
    public void lost(){
        this.isLose = true;
    }
    public Hand showHand(){
        Hand play;
        if(this.isLose){
            System.out.print("ROCK-Mode: ");
            play = Hand.ROCK;
        }else{
            double rnd = Math.random();
            if(rnd < 1.0/3.0){
                play = Hand.ROCK;
            }else if (rnd < 2.0/3.0){
                play = Hand.PAPER;
            }else{
                play = Hand.SCISSORS;
            }
        }
        return play;
    }
}
```

java/RandomJankenPlayer.java

```java
public class Test5 {
    public static void main(String[] s){
        RandomJankenPlayer p1 = new RandomJankenPlayer(); // 「負けグー」
        RandomJankenPlayer p2 = new RandomJankenPlayer(); // 普通のランダム

        Hand h1, h2;

        for(int i=0; i<10; i++){
            h1 = p1.showHand();
            h2 = p2.showHand();
            System.out.println(h1+" vs "+h2);
            if( (h1==Hand.ROCK && h2==Hand.PAPER) ||
                (h1==Hand.PAPER && h2==Hand.SCISSORS) ||
                (h1==Hand.SCISSORS && h2==Hand.ROCK) ){
                p1.lost(); // p1に負けたことを通知
            }
        }
    }
}
```

java/Test5.java