● J2Kad10D「ECC 長屋、再び！」

羊を表す Sheep クラスが準備されている。ECC 長屋（1 階建て 5 部屋）へ Sheep を呼び込んで自己紹介させる処理を作成せよ。

① Sheep 型配列（要素数 5）を生成し、for 文を使って、羊の呼び込みと自己紹介を行う処理を作成せよ。
② ①で作成した for 文を拡張 for 文に変更して動作確認せよ。

課題完成時の画面

```
ECC 長屋の住人を募集します！
ベルヌイがやってきた！
ブールがやってきた！
パスカルがやってきた！
デカルトがやってきた！
フックがやってきた！

ECC 長屋の住人が自己紹介します！
「わたしはベルヌイ、よろしくメェ～！」
「わたしはブール、よろしくメェ～！」
「わたしはパスカル、よろしくメェ～！」
「わたしはデカルト、よろしくメェ～！」
「わたしはフック、よろしくメェ～！」
```

Sheep
name
Sheep() intro()

● J2Kad10C「列挙型①」

J2Kad10C にじゃんけんを行う処理が作成されている。勝敗判定を行い、結果を表示する処理を作成せよ。なお、勝敗（WIN、LOSE、DRAW）に関しては新規に Janken クラスを作成し、列挙型（Result）として定義すること。また勝敗判定は Janken クラスに勝敗表（2 次元配列 resultTbl）を作って判定すること。

リスト 1：Result 型の定義（新規クラス「Janken.java」）

```
public class Janken {
    enum Result {
        WIN,
        LOSE,
        DRAW,
    }
    public static Result[][] resultTbl = {
        作成すること
    };
}
```

列挙型 Result や勝敗表 resultTbl は J2Kad10C の中に作ってもよいが、授業ではあとの課題で使いまわしするため、Janken クラスに作っている。

● J2Kad10B「列挙型②」※J2Kad10C の main メソッドをコピーして作成すること

J2Kad10C の main メソッドをコピーし、以下の修正を行え。

① Janken クラスに作成した列挙型 Result に勝敗判定時に表示するメッセージとコンストラクタを追加し、main メソッドの結果表示処理を簡略化せよ。
② Janken クラスにじゃんけんの手（グー、チョキ、パー）を表す列挙型 Hand を追加し、main メソッドの処理を簡略化せよ。なお、可能な限り拡張 for 文を使うこと。

リスト１：勝敗の定義（クラス「Janken.java」）

```
public class Janken {
    enum Result {
        WIN("あなたの勝ちです！"),
        LOSE("あなたの負けです！"),
        DRAW("引き分けです！");
        public String msg;                              // 勝敗メッセージ
        Result(String msg) { this.msg = msg; }          // コンストラクタ
    }
          ⋮
    enum Hand {
        GU("グー", "チョキに勝って、パーに負けます！"),     // 名前、特徴
        CHOKI("チョキ", "パー勝って、グーに負けます！"),    // 名前、特徴
        PA("パー", "グーに勝って、チョキに負けます！");     // 名前、特徴
        public String name;                             // 名前
        public String feature;                          // 特徴
        コンストラクタで名前と特徴を設定する
    }
}
```

課題完成時の画面（J2Kad10D＆J2Kad10C）

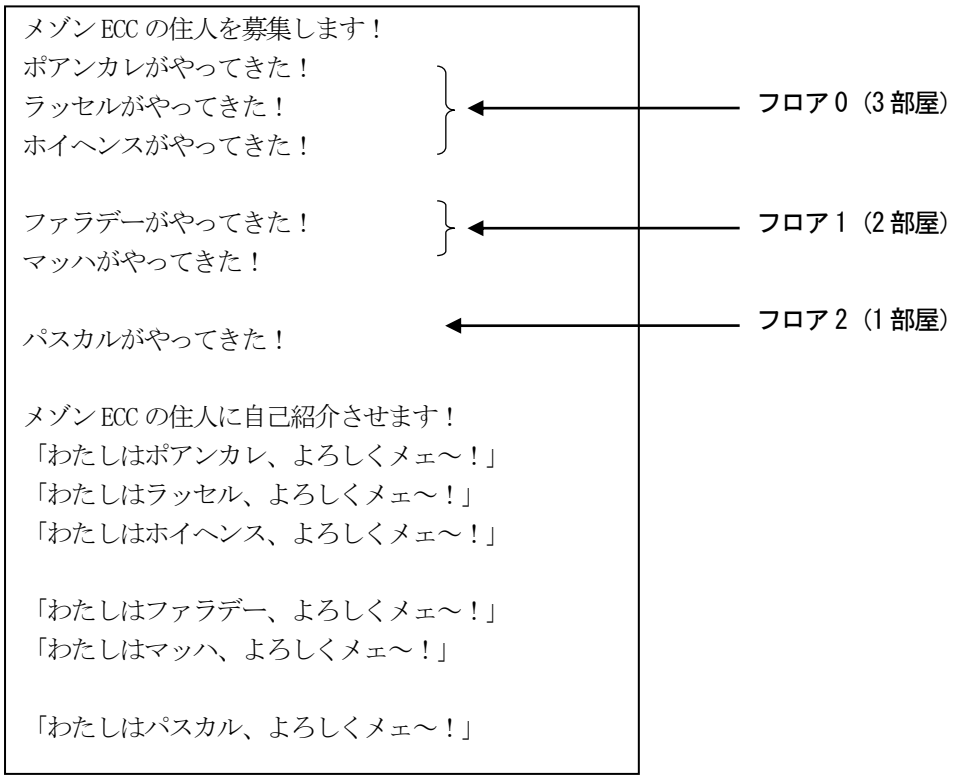
```
じゃんけんをします！
グー：　チョキに勝って、パーに負けます！
チョキ：パー勝って、グーに負けます！
パー：　グーに勝って、チョキに負けます！

何をだしますか？（0：グー、1：チョキ、2：パー、-1：終了）>0
あなたはグーを出した！
コンピュータはパーを出した！
あなたの負けです！

何をだしますか？（0：グー、1：チョキ、2：パー、-1：終了）>1
あなたはチョキを出した！
コンピュータはチョキを出した！
引き分けです！
```

J2Kad10D
結果表示を作成すること。

● J2Kad10A「メゾン ECC、再び！」

ECC 長屋が満室になったので、3 階建てマンション「メゾン ECC」を建てた！ただし北側斜線制限の影響でフロア 0 は 3 部屋、フロア 1 は 2 部屋、フロア 2 は 1 部屋にしなければならない。メゾン ECC へ Sheep を呼び込んで自己紹介させる処理を作成せよ。なお、可能な限り拡張 for 文を使うこと。

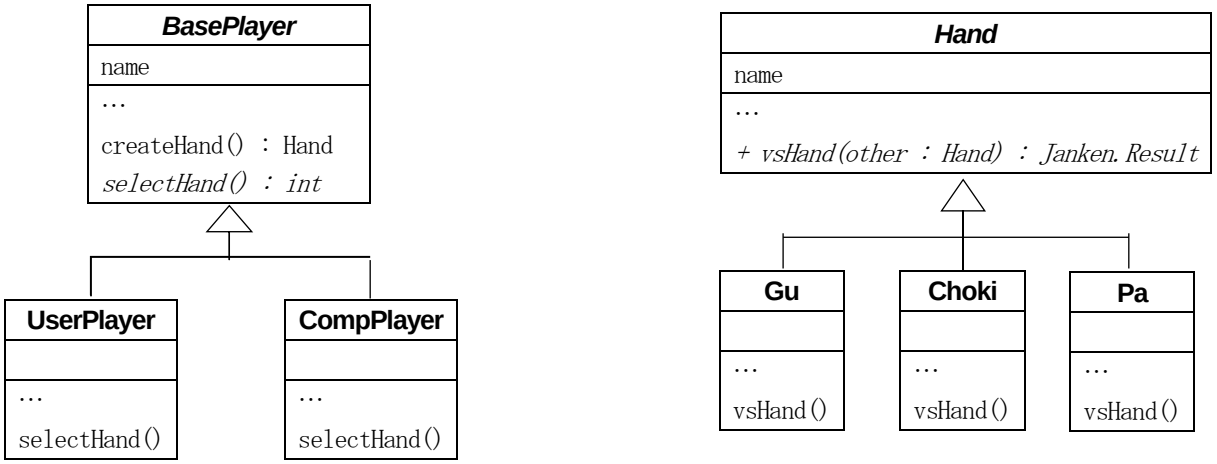
課題完成時の画面



● J2Kad10S「じゃんけん（クラス版）」（入門編 P.211「クラスの確認」）

ユーザーvs コンピュータのじゃんけんゲームの雛形が準備されている。

① UserPlayer・CompPlayer が「手」を選択する selectHand メソッドを実装せよ。
② Gu・Choki・Pa の各クラスが相手に勝てるかどうかを判定する vsHand メソッドを追加せよ。



selectHand メソッド（UserPlayer クラス、CompPlayer クラス）

メソッド	UserPlayer	CompPlayer
int selectHand()	0：グー、1：チョキ、2：パー、-1：終了を入力して返す。	乱数で 0、1、2 を返す。

vsHand メソッド（Gu クラス、Choki クラス、Pa クラス）

メソッド	仕様
Janken.Result vsHand(Hand other)	自分と other との対戦結果を返す（例：自分が勝つ場合は Janken.Result.WIN を返す） なお、other が Gu でも Choki でも Pa でもないときは null を返す。

ヒント：other が何のクラス化判定するのは以下のようにする（入門編 P.211「クラスの確認」）

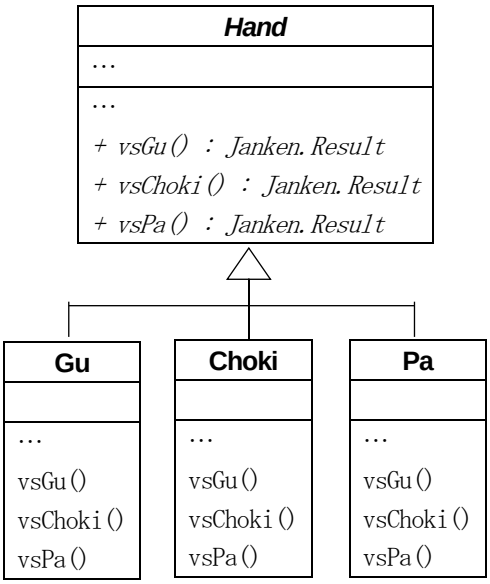
```
if (other instanceof Gu) {
    // other が Gu のとき true
}
```

課題完成時の画面

```
じゃんけんをします！
何をだしますか？（0：グー、1：チョキ、2：パー、-1：終了）>0
あなたはグーを出した！
コンピュータはグーを出した！
引き分けです！

何をだしますか？（0：グー、1：チョキ、2：パー、-1：終了）>1
あなたはチョキを出した！
コンピュータはパーを出した！
あなたの勝ちです！
    ⋮
```

● J2Kad10X「ダブルディスパッチ」※J2Kad10S の main メソッドをコピーして作成すること

J2Kad10S で作成した Gu・Choki・Pa クラスの vsHand メソッドの勝敗判定を、ダブルディスパッチを使って判定するように変更せよ。



追加するメソッド

追加するメソッド	仕様
Janken.Result vsGu()	自分にグーが勝てるとき、Janken.Result.WIN を返す 自分にグーが負けるとき、Janken.Result.LOSE を返す 自分とグーが引き分けるとき、Janken.Result.DRAW を返す
Janken.Result.vsChoki()	自分にチョキが勝てるとき、Janken.Result.WIN を返す 自分にチョキが負けるとき、Janken.Result.LOSE を返す 自分とチョキが引き分けるとき、Janken.Result.DRAW を返す
Janken.Result.vsPa()	自分にパーが勝てるとき、Janken.Result.WIN を返す 自分にパーが負けるとき、Janken.Result.LOSE を返す 自分とパーが引き分けるとき、Janken.Result.DRAW を返す

課題完成時の画面

（J2Kad10S と同じ）

● ダブルディスパッチ（vsHand メソッド）

ダブルディスパッチによって対戦結果を判定します。

① 自分の手（h0）に対し相手の手（h1）を引数にして、勝てるかどうか問い合わせる（ディスパッチ①）。
② 問い合わせを受けた「手」は、受け取った「手」に対し自分が勝てるかどうか問い合わせる（ディスパッチ②）。

「手」クラスには、対戦相手からの問い合わせに返答する関数を追加します（vsGu 関数、vsChoki 関数、vsPa 関数）。

自分の手（h0）がグーのとき