# **J2Kad09D「ECC長屋、再び！」（実践編P.110「拡張for文」）**

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**

|  |
| --- |
| **Sheep** |
| name |
| Sheep()  intro() |

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～！」

# **J2Kad09C「列挙型①」（実践編P.269「enum宣言」）**

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」）**

列挙型Resultや勝敗表resultTblはJ2Kad09C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。

public class Janken {

enum Result {

WIN,

LOSE,

DRAW,

}

public static Result[][] resultTbl = {

**作成すること**

};

}

# **J2Kad09B「列挙型②」※J2Kad09Cをコピーして作成**

J2Kad09Cをコピーして、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」）**

enumもクラスと同じようにフィールドやメソッドを持つことができる。

WINやLOSEやDRAWはResultのインスタンスのようなイメージ

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（J2Kad09C＆J2Kad09B）**

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！

J2Kad09C

結果表示を作成すること。

# **J2Kad09A「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（←検索）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。

**メゾンECCはこんなかんじ**

|  |  |  |  |
| --- | --- | --- | --- |
|  | maison | | |
| フロア2 |  |
| フロア1 |  |  |
| フロア0 |  |  |  |

**課題完成時の画面**

メゾンECCの住人を募集します！

ポアンカレがやってきた！

ラッセルがやってきた！

ホイヘンスがやってきた！

ファラデーがやってきた！

マッハがやってきた！

パスカルがやってきた！

メゾンECCの住人に自己紹介させます！

「わたしはポアンカレ、よろしくメェ～！」

「わたしはラッセル、よろしくメェ～！」

「わたしはホイヘンス、よろしくメェ～！」

「わたしはファラデー、よろしくメェ～！」

「わたしはマッハ、よろしくメェ～！」

「わたしはパスカル、よろしくメェ～！」

フロア0（3部屋）

フロア1（2部屋）

フロア2（1部屋）

# **J2Kad09S「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。
3. vsHandメソッドを使って勝敗判定を行え。

|  |
| --- |
| ***BasePlayer*** |
| name |
| …  createHand() : Hand  *selectHand() : int* |

|  |
| --- |
| ***Hand*** |
| name |
| …  *+ vsHand(other : Hand) : Janken.Result* |

|  |
| --- |
| **CompPlayer** |
|  |
| …  selectHand() |

|  |
| --- |
| **UserPlayer** |
|  |
| …  selectHand() |

|  |
| --- |
| **Choki** |
|  |
| …  vsHand() |

|  |
| --- |
| **Gu** |
|  |
| …  vsHand() |

|  |
| --- |
| **Pa** |
|  |
| …  vsHand() |

**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！

# **J2Kad09X「ダブルディスパッチ」※J2Kad09Sをコピーして作成**

J2Kad09S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。

|  |
| --- |
| ***Hand*** |
| … |
| …  *+ vsGu() : Janken.Result*  *+ vsChoki() : Janken.Result*  *+ vsPa() : Janken.Result* |

|  |
| --- |
| **Gu** |
|  |
| …  vsGu()  vsChoki()  vsPa() |

|  |
| --- |
| **Choki** |
|  |
| …  vsGu()  vsChoki()  vsPa() |

|  |
| --- |
| **Pa** |
|  |
| …  vsGu()  vsChoki()  vsPa() |

**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**

（J2Kad09S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）。

**例：自分の手（h0）がグーのとき**

Gu

①

Janken.Result vsHand(Hand other) {

return other.vsGu();

}

h0.vsHand(h1);

②

Janken.Result vsGu() { return Janken.Result.DRAW; }

Gu

h0に対しh1との対戦結果を

問い合わせる（ディスパッチ①）。

引数otherに対しGuとの対戦結果を

問い合わせる（ディスパッチ②）。

Janken.Result vsGu() { return Janken.Result.WIN; }

Choki

Janken.Result vsGu() { return Janken.Result.LOSE; }

Pa