10　アルゴリズムを考えながらプログラムを書こう：ポーカーゲームを例にして

前章までで、非常に多くのプログラムを開発してきました。しかし本講義での前章までのプログラミングは、主に文法を覚えるためのものでした。英語の試験に文法問題と長文問題があるように、プログラミングは文法を覚えるだけでなく長文を書けるようになる必要があります。むしろ、プログラミングの現場には、文法の試験などありません。皆さんは研究や仕事でプログラミングの作業に就くことになったら、ひたすら長文を書くことが必要となります。

そこで本章では、自分で長文を考えるように、あえて不親切に、あまりプログラムの詳細を見せずに、処理手順（アルゴリズム）を考えながらプログラムを考えてもらう練習をします。本章では練習の例題として、トランプのポーカーを実現するプログラムを、数段階に分けて開発します。

10.1　ポーカーとは

ポーカーとは、トランプの中から 5 枚を引き、その数字とマークから得られる「ハンド」という組み合わせの強さを競うものです。ハンドには弱い順に、以下のようなものがあります[9]。

ワンペア 同一数字のカード 2 枚のペア一組（残り 3 枚は何でもよい）。

ツーペア 同一数字のカード 2 枚からなるペアが二つ（残り 1 枚は何でもよい）。

スリーカード 同一数字のカード 3 枚（残り 2 枚は何でもよい）。

ストレート 5 枚のカードの数字が連続している。

フラッシュ 5 枚全てが同じマーク（数字は何でもよい）。

フルハウス ワンペアとスリーカードの組み合わせ。

フォーカード 同一数字のカード 4 枚（残り 1 枚は何でもよい）。

ストレートフラッシュ 5 枚のカードが連番で、なおかつ全て同じマーク。

本科目では、計算機が自動的に 5 枚のカードを引き、そのハンドを判定して表示する、というプログラムをつくることにします。

10.2　プログラミングその 1：5 枚のカードを引く

まずテキストエディタで `Poker1.java` というファイルを作ってください。そして `Poker1.java` に以下のようなプログラムを書いてください。

```
// ポーカーゲーム
// その 1： 5 枚のカードを、重複がないように引く
public class Poker1 {

  // 5 枚のカードの番号を表示する
  public static void print(int[] mark, int[] number) {
    (中略)
```

[9] もっと強いハンドに「ロイヤルストレートフラッシュ」がありますが、ここでは省略します。

```
  }

  // 5 枚のカードを引き、マークと番号を決定する
  public static void pick(int[] mark, int[] number) {
    (中略)
    // 結果を表示する
    print(mark, number);
  }

  // main メソッド
  public static void main(String[] args) {
    int[] mark = new int[5];
    int[] number = new int[5];

    // 5 枚のカードを引く
    pick(mark, number);
  }
}
```

ここで、main メソッドの変数 mark には、5 枚のカードのマーク（スペード、クローバー、ハート、ダイヤの 4 種類）のいずれかを、1 から 4 までの整数におきかえて記録することにします。また、変数 number には、5 枚のカードの番号（1 から 13 までの整数）を記録することにします。

pick メソッドの中の（中略）と書いた部分では、3.1 節に出てきた戦争ゲームと同じ要領で、5 枚のカードの各々について、乱数を使ってマークと番号を決定するプログラムを書いてください。

また、print メソッドの中の（中略）と書いた部分では、5 枚のカードのマークと番号を画面に表示するプログラムを書いてください。ただし、スペードを表す 1 は「S」に、クローバーを表す 2 は「C」に、ハートを表す 3 は「H」に、ダイヤを表す 4 は「D」に置き換えて表示してください。

私のプログラムでは、これを実行すると、以下のように表示されます。

```
S 9
S 13
D 8
S 2
H 2
```

これで、「5 枚のカードを引いた」のと同じ効果が出たことになりますが、しかし重要なことに気をつけないといけません。皆さんの現段階のプログラムでは、ひょっとしたら、マークも番号も全く同じカードを 2 枚引いてしまうかもしれません。現実のポーカーでは、それはありえないので、同じカードを 2 枚引かないような判定文をプログラムに加える必要があります。以下に、その処理のヒントを書きます。

i 番目のカードのマークと番号が決まりましたら、j 番目（$0 \leq j \leq (i-1)$）のカードについて、マークと番号を照合してみてください。もし i 番目と j 番目のカードが、マークも番号も両方とも同一だったら、i 番目のカードのマークと番号を、乱数を使ってもう一度決定してください。

ここまで書き終えたら、Poker1.java の動作を再確認してください。

10.3　プログラミングその 2：5 枚のカードを小さい順に並べ替える

ここでは、ポーカーのハンドを判定するために、5 枚のカードを小さい順に並べ替える処理を書き加えます。この並べ替えの処理を **ソート**ともいいます。ソートは非常に多くのプログラムで頻繁に用いられます。

まず Poker2.java を、以下のように書いてください。そして新しく加わった sotr というメソッドの中に、並べ替えのプログラムを書いてください。

```
// ポーカーゲーム
// その 1： 5 枚のカードを、重複がないように引く
public class Poker2 extends Poker1 {

   // 小さい順に並べ替える
   public static void sort(int[] mark, int[] number) {
     (中略)
     // 結果を表示する
     print(mark, number);
   }

   // main メソッド
   public static void main(String[] args) {
    (中略)
     // 5 枚のカードを引く
     pick(mark, number);
     // 小さい順に並べ替える
     sort(mark, number);
   }
}
```

並べ替えの処理手順は以下の通りです。まず i 番目のカードの番号を見てください。そして j 番目のカードについて、$j = (i+1)$ から $j = 4$ 番目まで順番に、i 番目のカードとの大小を比較してください。もし i 番目のカードより j 番目のカードの番号のほうが小さかったら、i 番目と j 番目のカードを **マーク・番号の両方とも**　入れ替えてください。

マークや番号を入れ替えるプログラムをつくるには、ちょっとした工夫が必要です。ヒントとして、mark と number の他に、もう 1 個変数を用意する必要があります。これを考えてみてください。

i に 0,1,2,3,4 の値をそれぞれ順に代入して、同様な処理を反復すれば、カードは小さい順に並び替えられるはずです。

以上の処理を、sort メソッドの（中略）と書かれた部分に挿入して、動作確認してみてください。私のプログラムでは、これを実行すると、以下のように表示されます。前半が並び替え前の 5 枚、後半がそれを並べ替えた結果です。

```
D 5
H 11
D 10
C 2
D 1
```

```
D 1
C 2
D 5
D 10
H 11
```

10.4 プログラミングその 3：ハンドを判定する (1)

ここでは、ワンペア、ツーペア、スリーカード、ストレート、の各々のハンドが該当するかを判定するメソッドを、Poker3.java の中につくりましょう。

一例としてワンペアが該当するかを判定するメソッドは、以下のように作ってください。

```
  // ワンペアを判定する
 public static boolean onepair(int[] mark, int[] number) {
   if(...)  {
     System.out.println("...  OnePair!");
     return true;
   }
   return false;
 }
```

このメソッドでは、if 文にてワンペアと判定されるようであれば、System.out.println 文を用いてワンペアであることを表示し、return true; の行で true を返します。if 文に入らない場合（ワンペアでない場合）は、return false; の行で false を返します。

ツーペア、スリーペア、ストレートについても、同様にメソッドを作ってください。

ここで、前節で開発した処理により、カードは番号の小さい順に並び替えられています。気がついたかと思いますが、この性質を利用することで、ワンペア、ツーペア、スリーカード、ストレート、いずれも判定のプログラムを簡単に記述できるはずです。

これらができましたら、Poker3.java を以下のように書いてください。

```
// ポーカーゲーム
// その 3： 5 枚のカードを、重複がないように引き、小さい順に並べ替え、いくつかのハンドを判定する
public class Poker3 extends Poker2 {

   // ストレートを判定する
   public static boolean straight(int[] mark, int[] number) {
     (中略)
   }

   // スリーカードを判定する
   public static boolean threecard(int[] mark, int[] number) {
     (中略)
```

```
  }

  // ツーペアを判定する
  public static boolean twopair(int[] mark, int[] number) {
    (中略)
  }

  // ワンペアを判定する
  public static boolean onepair(int[] mark, int[] number) {
    (中略)
  }

  // ハンドを判定する
  public static void judge(int[] mark, int[] number) {
    boolean ret;

    ret = straight(mark, number);
    if(ret == false) {
      ret = threecard(mark, number);
    }
    if(ret == false) {
      ret = twopair(mark, number);
    }
    if(ret == false) {
      ret = onepair(mark, number);
    }
  }

  // main メソッド
  public static void main(String[] args) {
    (中略)
    pick(mark, number);
    sort(mark, number);
    judge(mark, number);
  }
}
```

この中の judge メソッドでは、まずストレートを判定します。ストレートに該当しなければ、スリーカードを判定します。スリーカードにも該当しなければ、ツーペア、ワンペア、の順に判定します。このようにポーカーでは、高いハンドから順に判定することになります。

私のプログラムでは、これを実行すると、以下のように表示されます。

```
D 1
S 9
C 6
```

```
H 1
H 11

D 1
H 1
C 6
S 9
H 11

...  OnePair!
```

ここまで開発できましたら、何度も何度も、繰り返し実行してみてください。そして、ハンドが毎回必ず合っているか、よく見てよく確認してください。

10.5 プログラミングその 4：ハンドを判定する (2)

今度は残りのハンドとして、フラッシュ、フルハウス、フォーカード、ストレートフラッシュ、の 4 種類を、Poker4.java に書いてください。前節と同じように、各々のハンドに該当するなら return true; さもなければ return false; として、true または false のいずれかを返します。

Poker4.java には、上述の新しい 4 種類のハンドの他に、judge メソッドと main メソッドをつくってください。main メソッドは Poker3.java と全く同一です。judge メソッドは Poker3.java と似たような作り方で、合計 8 種類のハンドについて判定をしてください。判定する順番は、ストレートフラッシュ、フォーカード、フルハウス、フラッシュ、ストレート、スリーカード、ツーペア、ワンペアの順です。

10.6 プログラミングその 5：各々のハンドが何回ずつ出現するか集計する

最後に、「5 枚のカードを引いて、並べ替えて、ハンドを判定して」という処理を何度も繰り返して、各々のハンドが何回ずつ出現するか集計する、というプログラムをつくりましょう。以下の通り、Poker5.java を作ってください。

```
public class Poker5 extends Poker4 {
   public static int count0 = 0, count1 = 0, count2 = 0, count3 = 0,
            count4 = 0, count5 = 0, count6 = 0, count7 = 0;

   // 役を判定する
   public static void judge(int[] mark, int[] number) {
     (中略)
   }

   // main メソッド
   public static void main(String[] args) {
     (中略)
   }
```

}

このプログラムでは、メソッドの内部に入っていない変数 count0〜count7 を、各々のハンドの出現回数を集計するために使ってください。この出現回数の集計は、judge メソッドの中で行ってください。具体的には、ある特定のハンドに該当するときには、その該当するハンドに対応する変数（count0〜count7 のいずれか）に 1 を加えてください。

この count0〜count7 のように、メソッドの外側に変数を出すことで、メソッドを呼び出して返ってきてからも、変数に代入された値を保存し続けることができるようになります。このプログラムの場合、judge メソッドを何度も呼び出して、何度も返ってきても、集計値は保存し続ける必要があるために、count0〜count7 の各変数をメソッドの外に出すことになります。

main メソッドでは、pick, sort, judge の各メソッドを、何度も繰り返し呼んでください。そして、その反復が終了したら、各々のハンドの出現回数を表示してください。なお、反復回数が非常に多い場合には、画面表示に大きな時間がかかってしまいます。この場合に限り、Poker1.java の print メソッドの中にある System.out.println 文の行頭に//をつけるなどして無効化（コメントアウト）してください。

私のプログラムでは、これを実行すると、以下のように集計結果が表示されます。ちなみに私のプログラムでは、1 万回反復しています。

```
### StraightFlash:  1
### FourCard :  1
### FullHouese :  15
### Flash :  16
### Straight :  31
### ThreeCard :  222
### TwoPair :  484
### OnePair :  4273
```

このプログラムを何度か繰り返してみればわかりますが、上位のハンドになればなるほど、出現回数が少なくなります。ポーカーのハンドが、よくできたものであることが、これでわかるかと思います。