Java講座　第4回

Let's review!

2012.5.10(Thu)

担当者　清水　浜田

今回は復習！！

今までに覚えた（？）こと

- 入出力・変数・演算子
- 条件分岐
- ループ構文
- メソッド

入出力

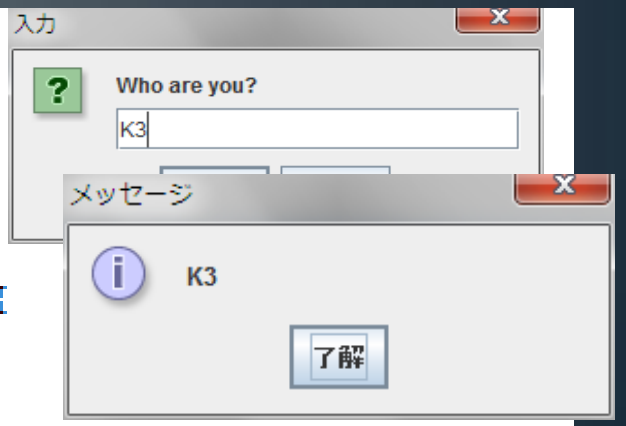
- String **変数名** = JOptionPane.showInputDialog(“**説明**”);
- JOptionPane.showMessageDialog(null, **表示させたいもの**);
- import javax.swing.JOptionPane; **を忘れずに！**

```
package lesson04;

import javax.swing.JOptionPane;

public class Review01 {
    public static void main(String[] args) {
        new Review01().start();
    }

    void start() {
        String aaa = JOptionPane.showInputDialog("Who are you?");
        JOptionPane.showMessageDialog(null, aaa);
    }
}
```



変数

- 変数を作るときは型を設定する（変数を使うときは不要）
  文字列⇒String　整数⇒int(Integer)　実数⇒double
  文字列⇒整数の変換はInteger.parseInt(変数);
  文字列⇒実数の変換はDouble.parseDouble(変数);

```
void start() {
    String input1 = "2012";
    int year = Integer.parseInt(input1);

    String input2 = "5.10";
    double monda = Double.parseDouble(input2);

    JOptionPane.showMessageDialog(null, year);
    JOptionPane.showMessageDialog(null, monda);
}
```

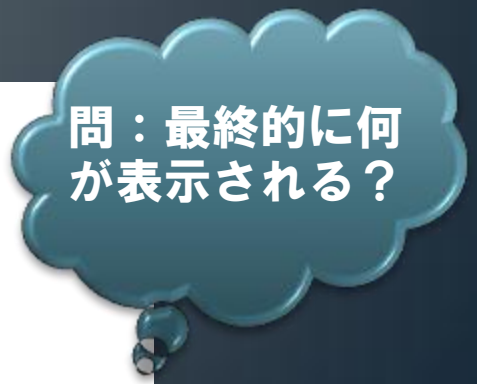
変数をつくったらすぐに代入する必要はない

```
int a;
int b = 10;
a = b;
```

演算子

- 四則　＋ － ＊ /（商） ％（余り）
- (不)等号　＝（左辺に右辺の値を代入） ＝＝（等しい）
  ＞＝（以上） ＜＝（以下） ＞＜
- 論理　＆＆（かつ）　｜｜（または）　！（否定）

```
void start() {
    int a = 10;
    int b = 20;
    int c = a + b;
    int d = c / a * (b - a);
    JOptionPane.showMessageDialog(null, "答えは" + d % b + a + "です。");
}
```



演習その１

・下記のプログラムの空白を埋めて「HelloWorld」とダイアログに表示させるプログラムを完成させよ。


```
package lesson04;

import javax.swing.JOptionPane;

public class HelloWorld {
    public static void main(String[] args) {
        new HelloWorld().start();
    }

    void start(){
        String a = "Hello";
        String b = "World";
        JOptionPane.showMessageDialog(null,_________);
    }
}
```

条件分岐if文

- if(条件１){
  処理内容1
  }
  else if(条件2){
  処理内容2
  }
  ・
  ・
  ・
  else{
  処理内容
  }

```
void start() {
    String conv = JOptionPane.showInputDialog("金いくら持ってる？");
    int money = Integer.parseInt(conv);

    if(money >= 10000) {
        JOptionPane.showMessageDialog(null, "金貸せ");
        money = 0;
    }
    else if(10000 > money && money >= 1000) {
        JOptionPane.showMessageDialog(null, "金持ってるなら消えろ");
    }
    else if(1000 > money && money >= 1) {
        JOptionPane.showMessageDialog(null, "恵んであげよう");
        money += 100000000;
    }
    else if(money == 0) {
        JOptionPane.showMessageDialog(null, "あっ、そう");
    }

    if(money != 0) {
        JOptionPane.showMessageDialog(null, "あなたは今\n"
                + "    " + money + "円\n" + "        持っています");
    }
    else {
        JOptionPane.showMessageDialog(null, "Game Over");
    }
}
```

繰り返しfor文

- for(int i = 0; i < 5; i++) {
  処理内容;
  }

これで処理内容が5回繰り返される

(i = 0, 1, 2, 3, 4)

i<5 iが5より小さい間はループを続ける
i++ ループが終わったらiの値に1を足す

```
void start() {
    int wallet = 0;
    String people = JOptionPane.showInputDialog("参加人数を入力");
    for(int i = 1; i <= Integer.parseInt(people); i++) {
        String money = JOptionPane.showInputDialog("献金額を入力");
        wallet += Integer.parseInt(money);
    }
    JOptionPane.showMessageDialog(null, wallet + "円もらいました。");
}
```

繰り返しwhile文

- while(条件){
  処理内容
  break;
  }

while文の中身は
永遠に繰り返す。
処理をとめるには
①break文を使う→
②条件から外れる
→次頁
と言った方法がある。

```
void start() {
    int result = 1;
    int count = 0;

    String num = JOptionPane.showInputDialog("整数を入力");
    int exp = Integer.parseInt(num);

    while(true) {
        result *= exp;
        count++;
        JOptionPane.showMessageDialog(null, result);

        if(result > 100000000 || count == 10) {
            break;
        }
    }
}
```

break文はループを終了させる働きを持つ。
while文は条件がtrueだと永遠にループする。
問：while文がループしなくなる条件は何か？

論理値boolean

- booleanもStringやint等と同じ変数の型（使い方も同じ）
- Boolean型の変数に代入できるのはtrueまたはfalseのみ。
- if文やwhile文の条件判定などに用いられる。

```
void start() {
    boolean LimitOrNot = true;
    int result = 1;
    int count = 0;

    String num = JOptionPane.showInputDialog("整数を入力");
    int exp = Integer.parseInt(num);

    while(LimitOrNot) {
        result *= exp;
        count++;
        JOptionPane.showMessageDialog(null, result);

        if(result > 100000000 || count == 10) {
            LimitOrNot = false;
        }
    }
}
```

このwhile文はboolean型の変数であるLimitOrNotにtrueが入っているときにループするという条件がついている。逆にLimitOrNotがfalseだとループしない。

問：while文がループしなくなる条件は何か？

演習その２

簡単なタイピングゲームを作ってみよう

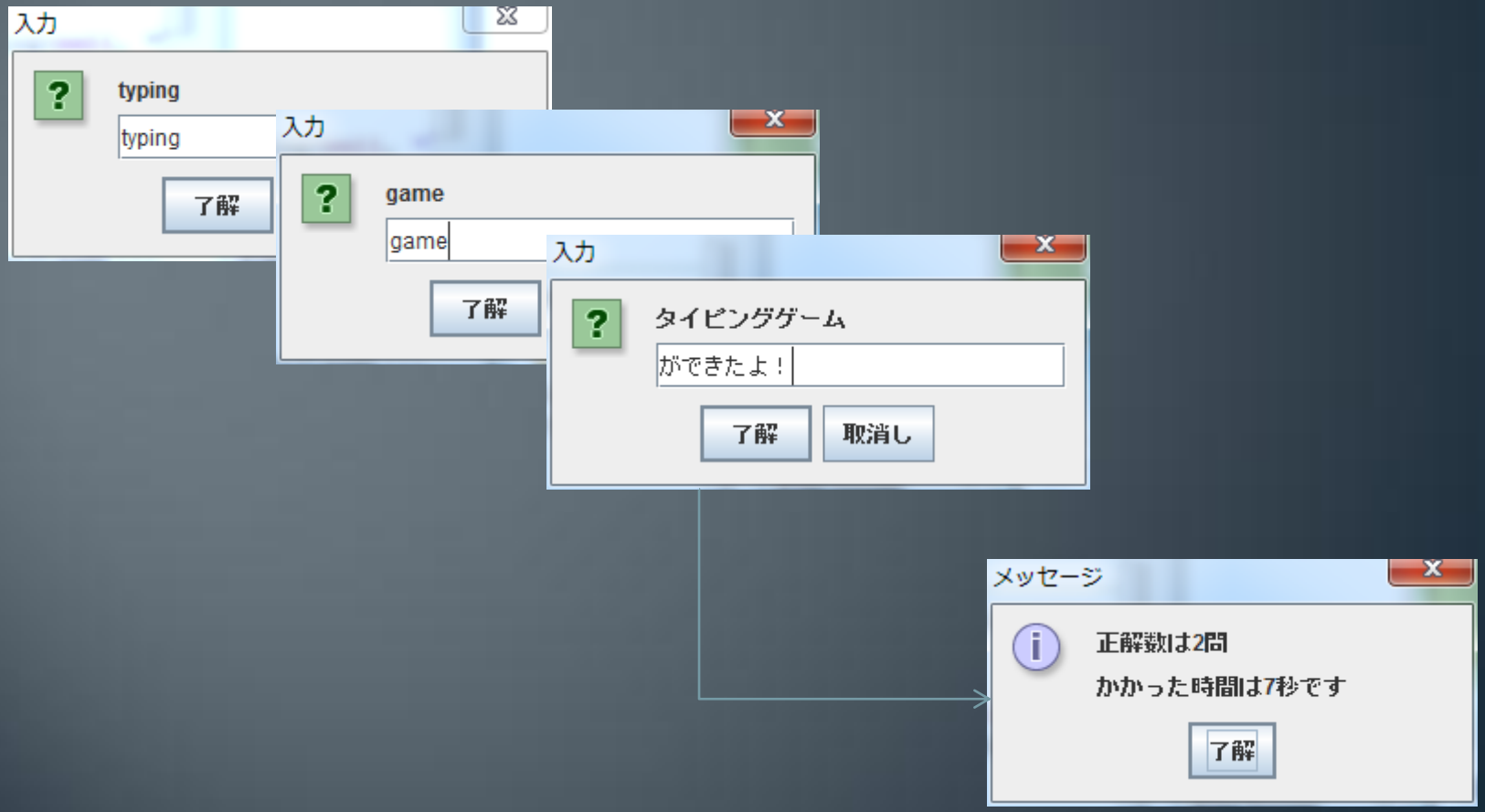
- インプットダイアログに表示した文字列と、入力された文字列が同じかを確かめる。
- これを３回繰り返し、正解数とかかった時間を返す。
  経過時間は、

```
long before = System.currentTimeMillis();
//------------ ここに処理を書く----------------//
long after  = System.currentTimeMillis();
long time = after - before;

```

こうするとtimeに処理時間がミリ秒で記憶される。

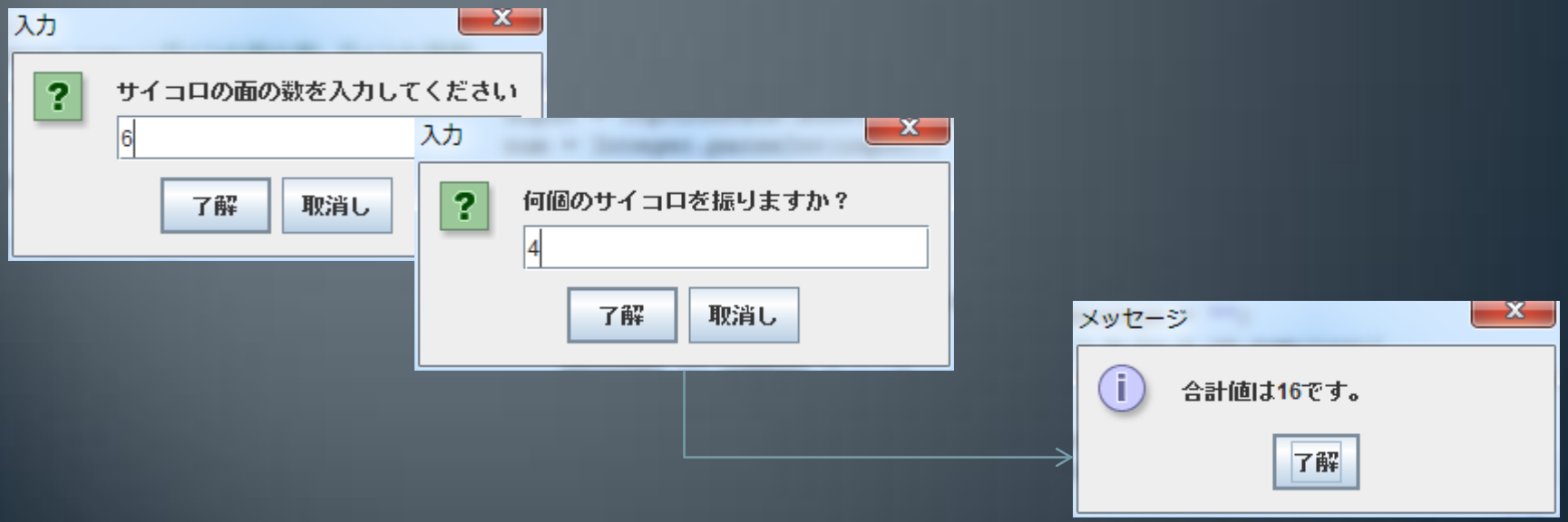
演習２の実行例


入力
typing
typing
了解
入力
game
game
了解
入力
タイピングゲーム
ができたよ！
了解
取消し
メッセージ
正解数は2問
かかった時間は7秒です
了解

演習その３

- n個のm面サイコロを振って、その出た目の合計を出力するプログラムを作成せよ。

実行例：



メソッド

- プログラムをまとめて名前をつけたもの（if文for文みたいな何か命令するものではなくただ単にプログラムをまとめる役割）
- メソッドを使わなくてもプログラムは書けるが、プログラムがわかりやすくなるので必須
- メソッドの例

```
void メソッド１(){
    処理内容１
}
void メソッド２(){
    処理内容2
}
```

メソッドの宣言

- メソッド名();
- これでこのメソッドの中身をこれから実行するという命令になる
- これがないとメソッドの中身を
  いくら書いても実行してくれない

```
void start() {
    Question();
    Answer();
}
```

Questionという名前のメソッドの中身を実行するぞー！

それが終わったらAnswerという名前のメソッドの中身を実行するぞー！

メソッドの中身

- void メソッド名(){
  処理内容
  }
- かぎ括弧の中がメソッドの中身になる
- startメソッドもこのように書きましたね

```
void Question() {
    String input = JOptionPane.showInputDialog("1+1=?");
    Integer.parseInt(input);
}

void Answer() {
    JOptionPane.showMessageDialog(null, "答えは2です");
}
```

Questionメソッドの中身

Answerメソッドの中身

メソッドの全体の流れ①

①startメソッドから実行

②startメソッド内にQuestionメソッドが宣言されているのでQuestionメソッドを実行する

③startメソッド内にAnswerメソッドが宣言されているのでAnswerメソッドを実行する

④プログラムの終了

```
void start() {  ①
    Question();  ②
    Answer();  ③
}

void Question() {  ②
    String input = JOptionPane.showInputDialog("1+1=?");
    Integer.parseInt(input);
}

void Answer() {  ③
    JOptionPane.showMessageDialog(null, "答えは2です");
}
```

Caution！！！

- メソッド同士は同格、
  if文・for文などはメソッドより格下！！！
  ⇒if文などはメソッドの中に書き、メソッドは
  他のメソッドの外に書く！

```
void Question() {
    void Answer() {

    }
}
```

メソッドの中にメソッドを書いちゃだめ！

```
void Question() {

}

for(i = 0; i < 5; i++) {

}
```

メソッドの外にfor文とか書いちゃだめ！

引数

- Javaの仕様上、違うメソッド同士で同じ変数のやり取りはできない

(例)Questionメソッドにある変数numをそのままAnswerメソッドに送ることはできない

```
void Question() {
    int num = 9;
}

void Answer() {
    num += 6;
}
```

Questionメソッドで作った変数numをAnswerメソッドで使いたいのだが・・・そのままではだめか

- でも変数の中身（データ）ならやり取りができる
- やり取りするために引数というものを使う

実引数

- Questionメソッド内で作られた変数numに入っている中身をAnswerメソッドに送りたい
- Questionメソッド内での操作

```
void Question() {
    int num = 9;
    Answer(num);
}
```

Answerメソッドを宣言するとき括弧の中に送りたい変数の名前を書く。
これでAnswerメソッドが起動するときにnumに入っている中身がAnswerメソッドに送られる。
※変数そのものは送られない

- 括弧の中に入っている変数を実引数と言う

仮引数

- Answerメソッドでnumの中身を受け取りたいが、それを入れる変数がAnswerメソッドにはない。そこで新しく変数を作る。
- Answerメソッドでの操作

```
void Answer(int a) {
    a += 6;
}
```

Answerメソッドの中身を書くとき、最初の括弧にnumの中身を入れるための変数を新しく作る。新しく作るので変数の型を書く必要があるが、それは実引数にあわせる。ここで作った変数はこのメソッド内で自由に使用可能。

- 括弧の中に入っている新しく作った変数を仮引数という。
- ここで作った変数は、もとの変数numと別の変数である。

メソッドの全体の流れ②

①startメソッドから実行

②startメソッド内にQuestionメソッドを宣言し、Questionメソッドを実行する。

③Questionメソッドの最後にnumを引数としAnsewerメソッドを宣言し、呼び出す。

④Answerメソッド実行時に、numの中身を入れるためのint型変数berが作られる。

⑤プログラムの終了

```
void start() {
    Question();
}
```

```
void Question() {
    String input = JOptionPane.showInputDialog("1+1=?");
    int num = Integer.parseInt(input);
    Answer(num);
}
```

```
void Answer(int ber) {
    if(ber == 2) {
        JOptionPane.showMessageDialog(null, "正解です。");
    }
    else {
        JOptionPane.showMessageDialog(null, "不正解です。答えは2です。");
    }
}
```

引数を複数取るメソッド

- 引数は2つ以上とることもできる

```
void Question() {
    String name = "清水健志";
    int age = 19;
    Answer(name, age);
}
```

変数nameの中身と変数ageの中身をAnswerメソッドに渡したい。Questionメソッドでは送りたい変数の名前をカンマ区切りで書く。

```
void Answer(String a1, int a2) {

}
```

Answerメソッドではnameとageの中身を入れるための変数を2つ作る。型を間違えないように！

メソッドに値を返す

- 自分自身を呼び出したメソッドに、値を返すことができる。

ここはいままでvoidだった。
値を返すときは返す変数の型を
ここに書く。

```
int Answer()  {
        int  a  =  10;
        return  a;
}
```

return 返す変数の名前;と書くことで値を返せる。
なお、返せる値は1つだけである。

メソッドの全体の流れ③

```
void start() {
    Question();
}
```

```
void Question() {
    String input1 = JOptionPane.showInputDialog("第1問:1+1=?");
    int num = Integer.parseInt(input1);
    String input2 = JOptionPane.showInputDialog("第2問:「東雲」はなんと読む？");

    int right = Answer(num, input2);

    JOptionPane.showMessageDialog(null, right + "問正解しました");
}
```

```
// 入力された回答二つを受け取って、正解かどうかを判定し、正解数を返すメソッド
int Answer(int a1, String a2) {
    int numCorrect = 0;
    if(a1 == 2) {
        numCorrect++;
    }
    if(a2.equals("しののめ")) {
        numCorrect++;
    }
    return numCorrect;
}
```

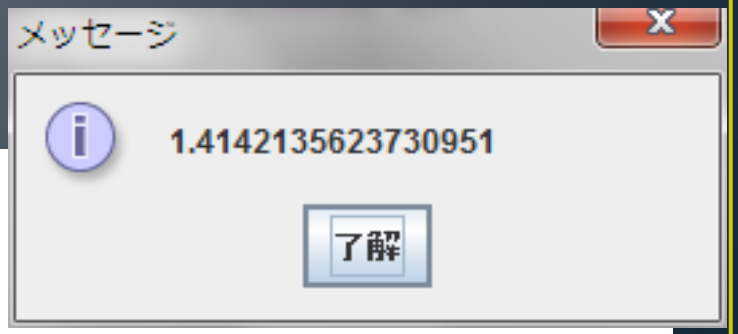
演習その４

- 演習３で作ったサイコロのプログラムを、２つのint型の引数を受け取り、合計値を返すint型のメソッドにせよ。

演習その５

- 2次方程式ax^2+bx+c=0の解を表示するプログラムを作りなさい。a,b,cの値はそれぞれ入力させること。なお、解は実数の範囲で考え、a=0のときの処理は考えなくてよい。

- ちなみに、Math.sqrt(変数);で変数の平方根を求めることができる。




```
int a = 2;
double b = Math.sqrt(a);
JOptionPane.showMessageDialog(null, b);
```

演習５ヒント

- まず解があるかどうかを判定するとよい。Boolean型の変数を作って解があるならtrue、ないならfalseを代入する。
- わかっていると思うが、2次方程式の解の公式は

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- 全部startメソッドの中に書いてもいいが、わかりにくくなると思うので、

  ①入力された文字列を整数として返すメソッド
  ②解があるかどうかを判定するメソッド
  ③解の公式を使って解を求め表示するメソッド
  を作るとよい。

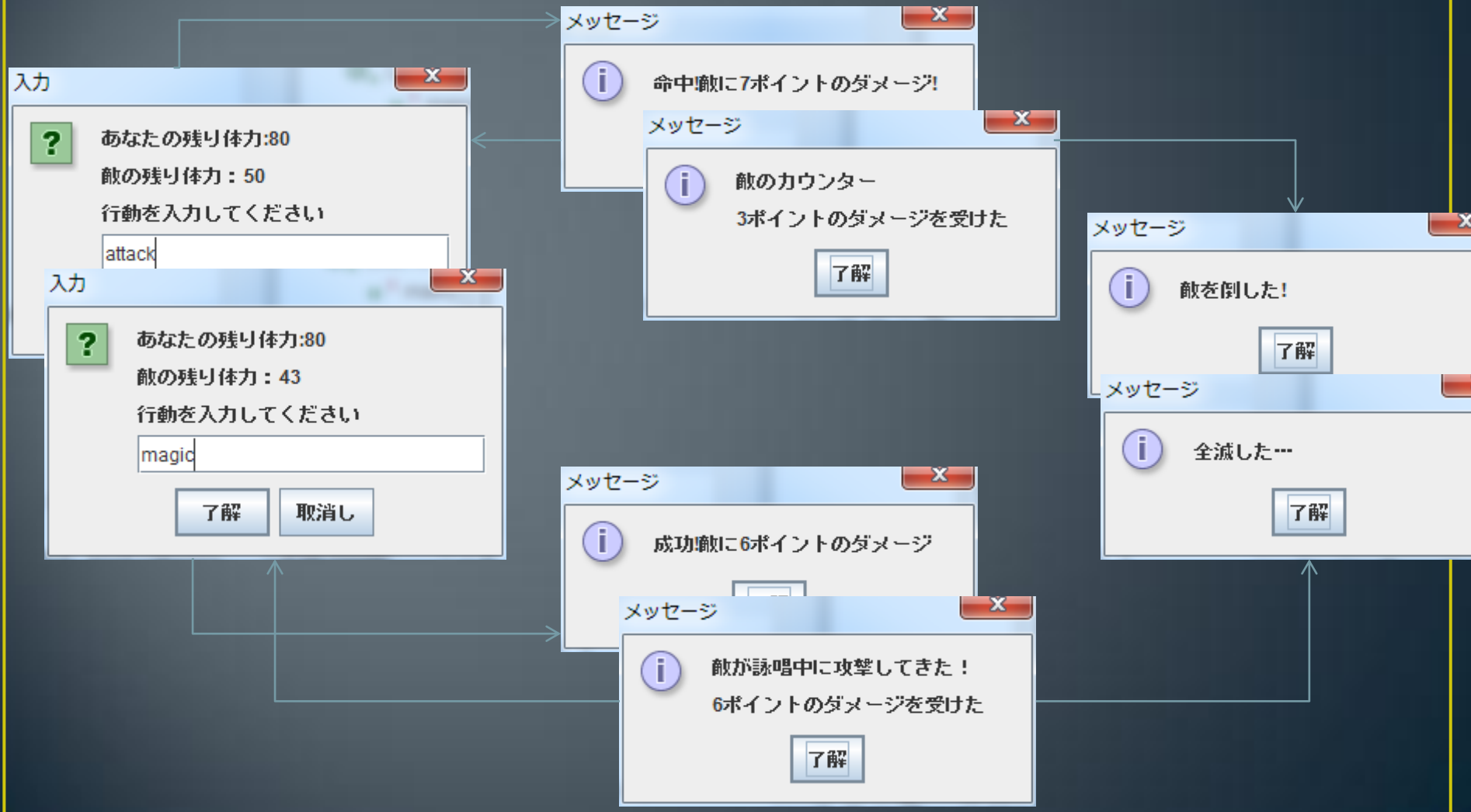
演習その６

簡単な戦闘システムを作ってみよう

- 最初にプレイヤーのHPと、敵のHPを決める。
- プレイヤーに行動を入力させて、それによって処理を分岐させる。
- サイコロで出目勝負をして、負けた方のHPを減らす。
  この時のサイコロの面の数、個数を処理によって変える。
  出目勝負には演習4で作った関数を使うと良い
- これをプレイヤーか敵のHPが０以下になるまで繰り返す。

これが出来たらいろいろ改造してみよう

演習６実行例


入力
あなたの残り体力:80
敵の残り体力：50
行動を入力してください
attack
メッセージ
命中!敵に7ポイントのダメージ!
メッセージ
敵のカウンター
3ポイントのダメージを受けた
了解
入力
あなたの残り体力:80
敵の残り体力：43
行動を入力してください
magic
了解
取消し
メッセージ
敵を倒した!
了解
メッセージ
全滅した…
了解
メッセージ
成功!敵に6ポイントのダメージ
メッセージ
敵が詠唱中に攻撃してきた！
6ポイントのダメージを受けた
了解

演習６ヒント

プログラムの概形の一例を挙げると

```
void start(){
  //ループ内で使う変数の初期化//
  // プレイヤーのHPと敵のHPを決める
  //---------------------//
  while(敵のHPが０以上 かつ プレイヤーのHPが０以上){
    プレイヤーの行動を入力させるダイアログを表示
    if(入力が行動1を表すとき){
      // 行動１の処理 //
    }else if(入力が行動2を表すとき){
      // 行動2の処理 //
    }
  }
  if(敵のHPが0以下){
   // 敵を倒した時の処理 //
  }else if(プレイヤーのHPが0以下){
   // 倒された時の処理 //
  }
}
```

- これで復習は終わりです。
- ここまで理解すればプログラミング入門１の内容の4分の3はできたも同然！
- メソッドは難しいかもしれませんが、これからプログラムを書く上で欠かせないのでがんばって勉強しましょう。
- 次回は7/14（月）に行います。内容は配列とクラスです。
- 次回の内容も非常に重要です。
- また会いましょう。