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

今日はこれらを復習して行きます。

入出力

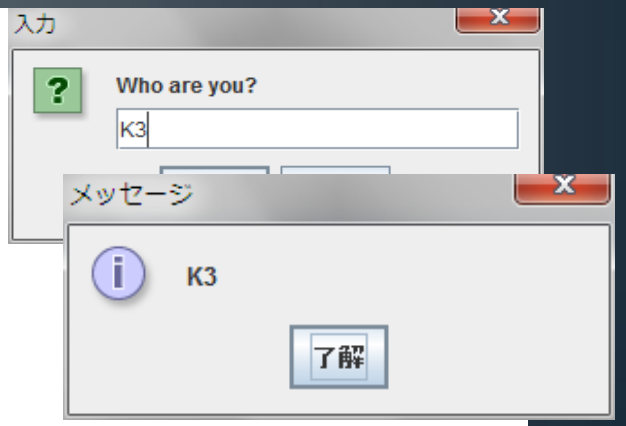
- String **変数名** = JOptionPane.showInputDialog(“**説明**”);
- JOptionPane.showMessageDialog(null, **表示させたいもの**);
- import javax.swing.JOptionPane; **を忘れずに！**

```
package lesson04;

import javax.swing.JOptionPane;

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

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



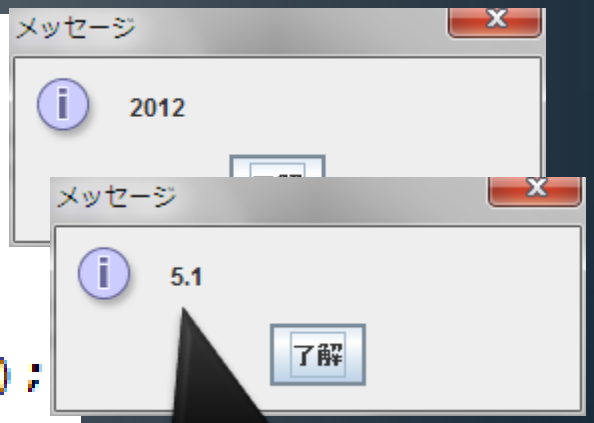
変数

- 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);
}
```



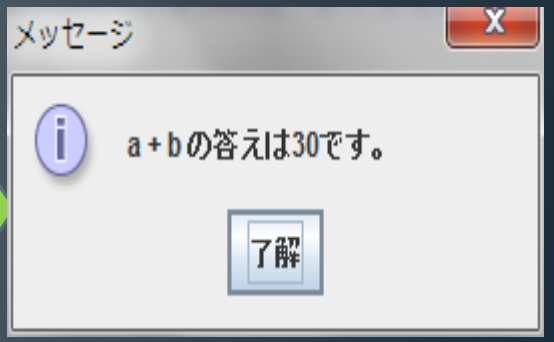
Double型の変数の小数点以下の末端の0は省略される

演算子

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

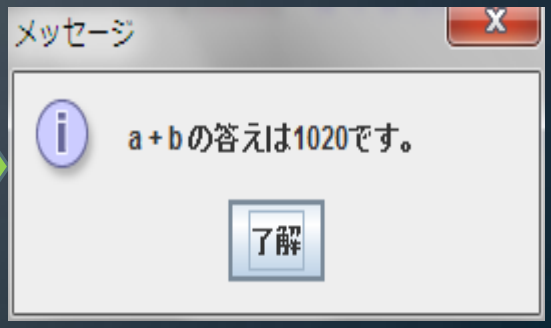
```
void start() {
    int a = 10;
    int b = 20;
    JOptionPane.showMessageDialog(null, "a + b の答えは" + (a + b) + "です。");
}
```

a+bの答えを出力




```
void start() {
    int a = 10;
    int b = 20;
    JOptionPane.showMessageDialog(null, "a + b の答えは" + a + b + "です。");
}
```

aとbを別々に出力してしまっている。



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("得点を入力(0-50)");
    int points = Integer.parseInt(conv);

    if(50 >= points && points >= 40) {
        JOptionPane.showMessageDialog(null, "ボーナス+10");
        points += 10;
    }
    else if(40 > points && points >= 20) {
        JOptionPane.showMessageDialog(null, "よくがんばりました");
    }
    else if(20 > points && points >= 0) {
        JOptionPane.showMessageDialog(null, "もっとがんばりましょう");
    }
    else {
        JOptionPane.showMessageDialog(null, "不正な値です");
    }

    if(60 >= points && points >= 0) {
        JOptionPane.showMessageDialog(null, "あなたの得点は" + points + "点");
    }
    else {
        JOptionPane.showMessageDialog(null, "Game Over");
    }
}
```

繰り返しfor文

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

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

これで処理内容が5回ループされる（繰り返される）

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

```
void start(){
    String input = JOptionPane.showInputDialog("ループ回数を入力");
    int num = Integer.parseInt(input);
    int loop = 0;
    for(int i = 0;i < num ;i++){
        loop++;
        JOptionPane.showMessageDialog(null, "ループ"+ loop + "回目");
    }
    JOptionPane.showMessageDialog(null, loop + "回ループしました");
}
```

iは0,1,2,・・・,numまで増えていき、i =numになった瞬間ループを抜ける。最終的にnum回だけfor文の中を処理する。

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) {
            JOptionPane.showMessageDialog(null, "値が1億を越えました");
            break;
        }
        else if(count == 10) {
            JOptionPane.showMessageDialog(null, "ループ回数が10回になりました");
            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。

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 isLoop = true;
    int result = 1;
    int count = 0;

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

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

        if(result > 100000000) {
            JOptionPane.showMessageDialog(null, "値が1億を越えました");
            isLoop = false;
        }
        else if(count == 10) {
            JOptionPane.showMessageDialog(null, "ループ回数が10回になりました");
            isLoop = false;
        }
    }
}
```

このwhile文はboolean型の変数であるisLoop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isLoopがfalseだとループしない。

否定演算子

- If文やwhile文の条件式になっているboolean型の変数の前に！をつけることで値を反転することができる。すなわち、**変数の値がtrueならfalseに、falseならtrueに変わる。**

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

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

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

        if(result > 100000000) {
            JOptionPane.showMessageDialog(null, "値が1億を越えました");
            isLoop = true;
        }
        else if(count == 10) {
            JOptionPane.showMessageDialog(null, "ループ回数が10回になりました");
            isLoop = true;
        }
    }
}
```

変数isLoopの前に！をつけることによって値が反転する。

！によってループを止めるにはisLoopにtrueを代入しなければいけない。

擬似乱数

- Math.random()で0以上で1未満の乱数を生成できる。

```
void start() {
    int sum = 0;

    while(true) {
        double random = Math.random() * 6;
        int dice = (int)random + 1;
        sum += dice;
        JOptionPane.showMessageDialog(null, dice + "の目が
        if(dice == 1) {
            JOptionPane.showMessageDialog(null, "1の目が出たので終了します。\n" +
                    "出た目の合計は" + sum + "です。");
            break;
        }
    }
}
```

Math.randomで0以上で1未満の乱数を生成し、それを6倍することで0以上で6未満の乱数をつくっている。

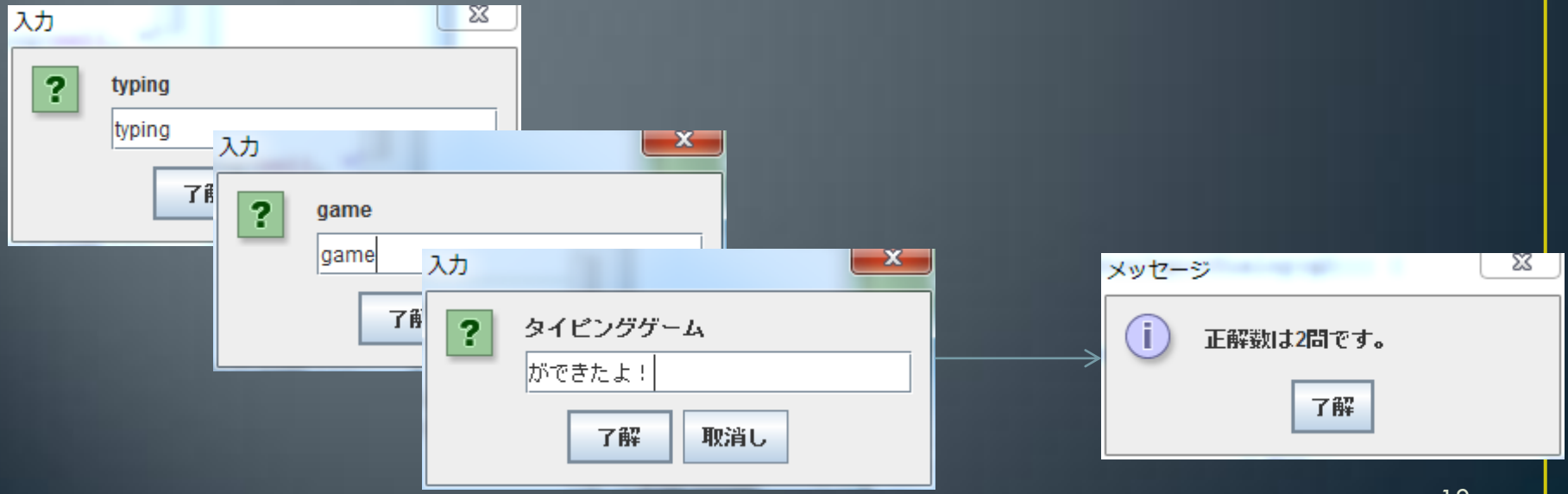
(int)をつけることで小数点以下の数字を切り捨てて整数にしている。これによって0, 1, 2, 3, 4, 5の数字のどれかになるので、1を足してさいころにしている。

演習その１

簡単なタイピングゲームを作ってみよう

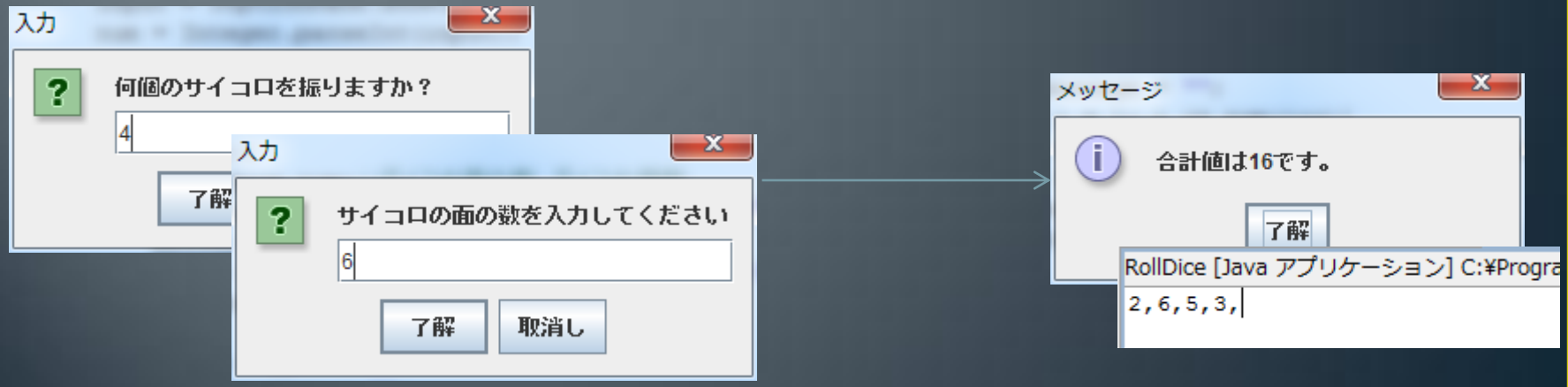
- インプットダイアログに表示した文字列と、入力された文字列が同じかを確かめる。
- これを３回繰り返し、正解数を返す。

実行例：



演習その２

- 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。
- さらに、サイコロ一つ一つの出目をコンソールに表示せよ。
  コンソールへの出力は、System.out.print()もしくは、System.out,println()を使う。
  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
  }
- {}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() {
    JOptionPane.showMessageDialog(null, "1+1=?");
}

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

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

②
void Question() {
    JOptionPane.showMessageDialog(null, "1+1=?");
}

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

Caution！！！

- メソッド同士は同格、
  命令文などはメソッドより格下！！！
  ⇒命令文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() {

    }
}
```

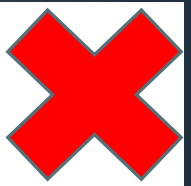
メソッドの中にメソッドを書いちゃだめ！

```
void Question() {

}

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

}
```



メソッドの外に命令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Answer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numberが作られる。そしてAnswerメソッドを実行する。

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

void Question() {  ②
    String input = JOptionPane.showInputDialog("1+1=?");
    int num = Integer.parseInt(input);
    Answer(num);③
}

void Answer(int number) {④
    if(number == 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 = "法政太郎";
    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 namae, int nenrei) {
```

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。1番目の実引数の中身は1番目の仮引数に、2番目の実引数の中身は2番目の仮引数に入る。3番目以降の実引数も同様。

仮引数の型は正確に！

- 下みたいに実引数と仮引数で型が違っているとOUT

```
void Question() {
    String name = "法政太郎";
    int age = 19;
    Answer(name, age);
}
```

実引数がString型なのに仮引数はint型

実引数がint型なのに仮引数はString型

```
void Answer(int nenrei, String namae) {
```

メソッドに値を返す①

- 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。
値を返さないときは今までどおりvoidでいい。

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

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。

メソッドに値を返す②

- 型 変数 = メソッド名(実引数);とすれば、返ってきた値をそのまま変数に入れられる。当然だが、返ってくる値の型と変数の型は同じでなくてはいけない。

```
void start() {
    int ans = Question();
    JOptionPane.showMessageDialog(null, ans);
}

int Question() {
    String input = JOptionPane.showInputDialog("入力した数字+1");
    int num = Integer.parseInt(input);

    return num + 1;
}
```

Questionメソッドを実行し、Questionメソッドから返ってきた値を変数ansに入れる

numに1を足した値をQuestionメソッドを宣言したところに返す。

メソッドに値を返す③

- boolean型の値(trueまたはfalse)を返すことで、if文やwhile文の条件式をメソッドを使って表すことが可能になる。

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

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

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

boolean judge(int nowResult, int nowCount) {
    if (nowResult > 100000000) {
        JOptionPane.showMessageDialog(null, "値が1億を越えました");
        return false;
    }
    else if (nowCount == 10) {
        JOptionPane.showMessageDialog(null, "ループ回数が10回になりました");
        return false;
    }
    else {
        return true;
    }
}
```

このwhile文の条件式は、judgeメソッドを実行しjudgeメソッドから返ってくる値がtrueならループを繰り返し、falseならループをやめる、と言う意味である。なお、judgeメソッドを宣言する際変数resultとcountを実引数に取っている。

Judgeメソッドは仮引数として変数nowResultとnowCountを作り、この2つの変数の値によってfalseを返してループをやめさせるか、trueを返してループを続けるかを判定する。

メソッドに値を返す④

- 条件式となるメソッドの名前の前に！をつけることで返ってきたboolean型の値を反転することができる。
- すなわち、**返ってきた値がtrueならfalseに、falseならtrueになる。**
- P10と要領は同じ。
- これより上のプログラムは前頁と同じ

```
    while (!judge(result, count)) {
        result *= exp;
        count++;
        JOptionPane.showMessageDialog(null, result);
    }
}

boolean judge(int nowResult, int nowCount) {
    if (nowResult > 100000000) {
        JOptionPane.showMessageDialog(null, "値が1億を越えました");
        return true;
    }
    else if (nowCount == 10) {
        JOptionPane.showMessageDialog(null, "ループ回数が…回");
        return true;
    }
    else {
        return false;
    }
}
```

メソッドjudgeの前に！をつけたことで返ってくる値を反転する。

よって、ループをやめるときはtrue,ループを続けるときはfalseを返さなければいけない。

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;
}
```

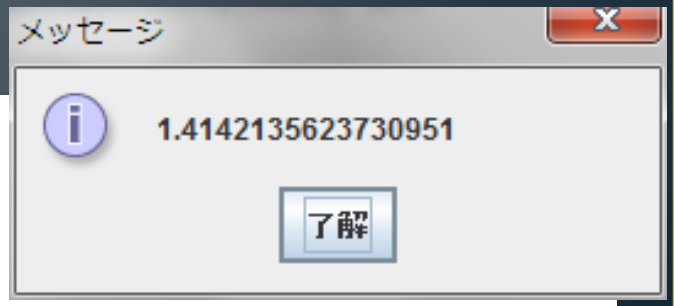
演習その３

- 演習３で作ったサイコロのプログラムを、サイコロの面の数と、振る個数を受け取り、サイコロの目の合計値を返すint型のメソッドにせよ。

演習その４

- 2次方程式ax^2+bx+c=0の解を表示するプログラムを次からの2ページに示すので、赤線が引いてある空欄を埋めなさい。なお、解は実数の範囲で考え、a=0のときの処理は考えない。
- プログラムはstart()メソッドから始まるものとする。
- 2次方程式の解の公式は$x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$
- 判別式は$D = b^2 - 4ac$
- 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);
```



```
void start() {
    String message="aの値を入力してください";
    double numA= ____________;

    message="bの値を入力してください";
    double numB= ____________;

    message="cの値を入力してください";
    double numC= ____________;

    solveEquation(____________);
}

// 入力された値をdouble型で返すメソッド
____________ inputDouble(String message){
    String input = JOptionPane.showInputDialog(message);
    return Double.parseDouble(input);
}

// 判別式を求めるメソッド
double getDiscriminant(double a,double b,double c){
    double discriminant =  b * b - 4 * a * c;
    return ____________;
}
```

```
// 2次方程式の解を求めるメソッド
void solveEquation(double a, double b, double c) {
    double D = getDiscriminant(a, b, c);
    if (              ) {
        JOptionPane.showMessageDialog(null, "解なし");
    } else {
        double solutionPlus = (-b + Math.sqrt(D)) / (2 * a);
        double solutionMinus = (-b - Math.sqrt(D)) / (2 * a);

        if (solutionPlus == solutionMinus) {
            JOptionPane.showMessageDialog(null, "解は" + solutionPlus);
        } else {
            JOptionPane.showMessageDialog(null, "解は" + solutionPlus + "と"+ solutionMinus);
        }
    }
}

// 2次方程式が解を持つかどうかを判定するメソッド
            haveSolutionOrNot(double D) {
    if (D < 0) {
        return false;
    } else {
        return true;
    }
}
```

演習４ヒント

- プログラムを順番に追っていこう。
- コメントアウトをヒントにしてメソッドがどういう働きをしているか考えてみよう。
- 実引数と仮引数の関係は正しいだろうか？
- return文はあるだろうか？あるとしたらその型は？
- If文の条件式にはP9の条件式のような使い方も可能。

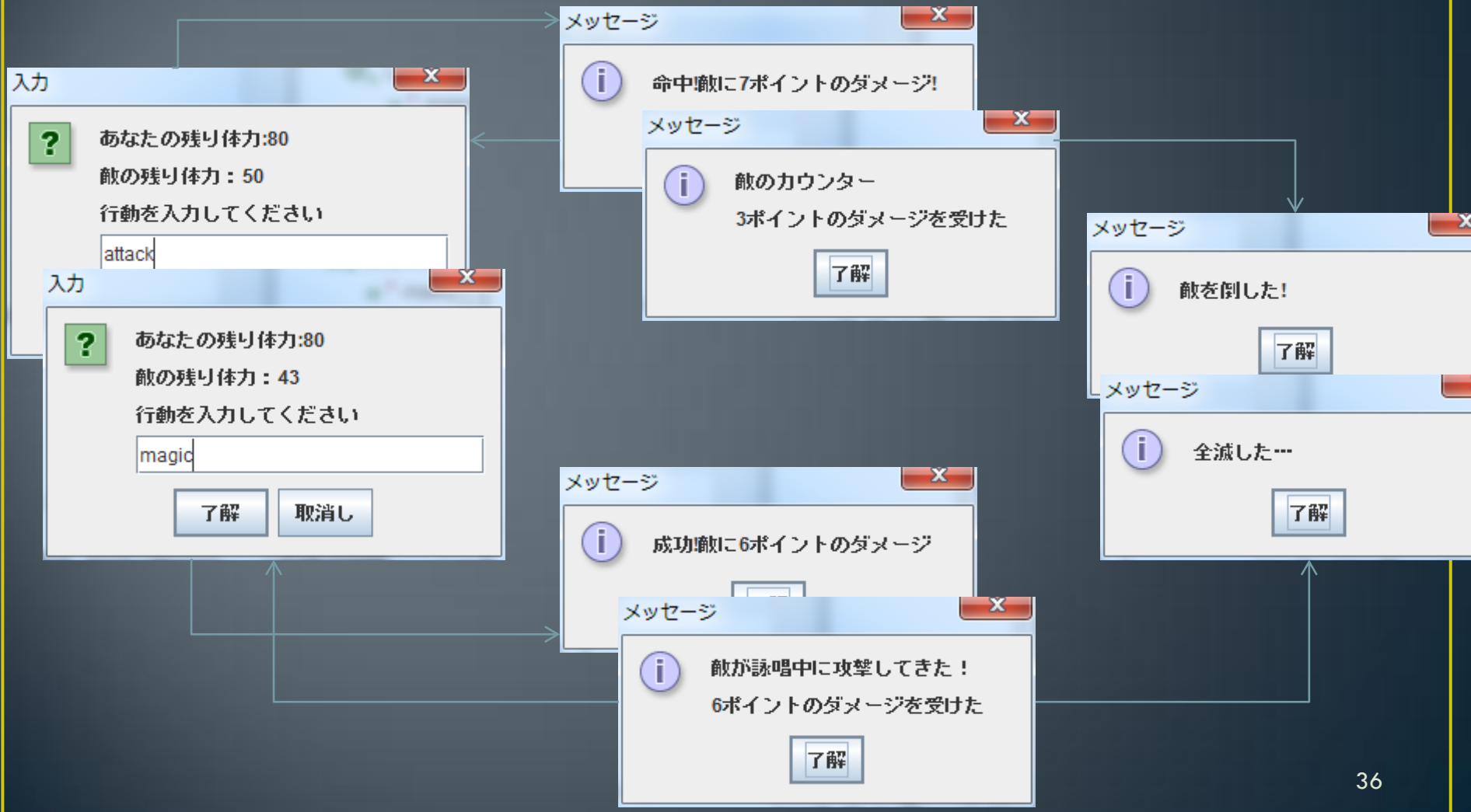
演習その５

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５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５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5/14（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。