JAVA講座第3回

メソッド

```
package パッケージ名;

import javax.swing.JOptionPane;

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

        void start() {

                内容

        }


}
```

・説明では、上記の雛型を省略しています。ご了承ください。
・ (例題はk3のhpにupしました。コピペして使ってください。一部upされていない問題は自分で書いてください)

内容

- 復習
- メソッド(void型)
- 関数(int型　double型　String型　boolean型　のメソッド)

参照

https://java2010.cis.k.hosei.ac.jp/04-2/material-04/

https://java2010.cis.k.hosei.ac.jp/05-2/material-05/

復習(for文)

- **for**文は、ある変数がある値からある値までの回数分繰り返す機能を持つ。

- **for**文は、繰り返す回数が分かっているときに使う。

- **for((**始める値**);(**継続条件**);(**値の加算もしくは減算**)) {(**繰り返したい命令**)}**で実現できる。

復習(for文)
クラス名：Ex01for

```
int total1 = 0;
for (int i = 1; i < =5 ; i++){
            total1= total1 + i;
}
//total1=1+2+3+4+5
JOptionPane.showMessageDialog(null,total1);
```

i++はi=i+1と同じ
ちなみに　i=i+2　だとiが2ずつ増える

⇒新規定数iを生成、iが1からスタート、{ }内の命令で、total1にiの値を足し、
一通り終わったら、iに1を加算し、iが5以下なら、{ }内の命令を継続。
つまり、iが1～5のときの計5回繰り返す命令。

```
int total2= 0;
for (int i = 5; i > 0 ; i--){
            total2 = total2 + i;
}
//total2=5+4+3+2+1
JOptionPane.showMessageDialog(null,total2);
```

i--はi=i-1と同じ

⇒このように、iを5からスタートして、一通り終わったら、iから1を減算し、
iが0以上のときまで、命令を繰り返すこともできる。

復習(while文)

- **while**文は、ある条件を満たすまで永遠に処理を繰り返す機能を持つ。

- **while**文は繰り返す回数が分からないときに使われるので多くの場面で用いられる。

- **while((条件)){ … }**でてき、**(条件)**を満たすまで**{… }**内の命令を繰り返す。

- 条件を**true(真)**にすると無限ループが完成する。

  ＊脱出には**break;**を使う。

復習(while文)
クラス名：Ex02while

```
int total1 = 0;
int total2 = 0;
int i = 0;
while(i<=5){
      total1=total1+i;
      i++;
}
JOptionPane.showMessageDialog(null,total1); // total1=0+1+2+3+4+5
```

⇒while文の中は、total1にiを足して、iに1を足す、と言う命令を、iが5以下のときは実行し続けるという意味。

```
i=5;
while(true){
      total2=total2+i;
      if(i==0){
          break;
      }
      i--;
}
JOptionPane.showMessageDialog(null, total2); // total2 = 5+4+3+2+1+0
```

⇒while文の中は、、total2にiを足して、iから1を引く、と言う命令を、永遠に実行し続けるが
i==0なら終了するという意味。

条件分岐（if文）

```
if(条件){
    内容1
}else if(ifの条件を満たさないような条件) {
    内容2
}else{       // elseはif、else if の条件を満たさないもの。
    内容3
}
```

こう書くことで、条件によって内容1、内容2、内容3のどれかが実行されます。
内容1、内容2、内容3　が同時に実行されることはありません。

```
if(もし~なら){
    内容1
}else if(そうじゃなくて、~なら){
    内容2
}else{   //全部違うなら
    内容3
}
```

こんな感じ。

条件分岐（if文）
クラス名:Ex03if

```
String input = JOptionPane.showInputDialog("0～100の範囲で点数を入力してください");
int point = Integer.parseInt(input);
if (point >= 80 && point <= 100) {
    JOptionPane.showMessageDialog(null, "高得点です");
} else if (point >= 60 && point < 80) {
    JOptionPane.showMessageDialog(null, "そこそこの点です");
} else if (point >=  0 && point < 60){
    JOptionPane.showMessageDialog(null, "もっとがんばりましょう");
} else {
    JOptionPane.showMessageDialog(null, "0～100の値が入力されました");
}
```

⇒点数が100~80なら***"高得点です"***

60~80 なら***"そこそこの点です"***

0~59　なら***"もっとがんばりましょう"***

それ以外の値入力された場合

"0～100の値が入力されました"

を表示します。

```
}else if (point >= 60 ){
} else if (point >=  0 ){
```

と書いても、同じですが、わかりにくいのでやめましょう

日本語	数学表記	JAVA表記
AとBは等しい	A ＝ B	A ＝＝ B
AとBは等しくない	A ≠ B	A ！＝ B
AよりBが大きい	A ＜ B	A ＜ B
AよりBが小さい	A ＞ B	A ＞ B
AがB以上	A ≧ B	A ＞＝ B
AがB以下	A ≦ B	A ＜＝ B
AかつB	A ∩ B	A ＆＆ B
AまたはB	A ∪ B	A ｜｜ B

擬似乱数

- 擬似乱数とは、でたらめに数字を出すことであり、これを用いることでサイコロ振るなどのプログラムを作ることができる。

- **Math.random()**は、**0.0以上1.0未満**の数字を適当に出す命令です。

- **Math.random()は、もちろんdouble**型です。

- 前回の資料では
  double random = Math.random()*6;
  int dice = (int) random + 1;
  でしたが。
  int dice = (int)(Math.random()*6) + 1;
  でも、一緒です。

- ちなみに、(int)というのは、(int)double型の数　と書くことで
  double型の数の小数を切り捨て、int型に直します。このことを**キャスト**といいます

- **(int)Math.random()*6 としてしまうと(int)Math.random()が0になり、**
  (int)Math.random()*6 = 0*6 = 0 となるので、注意してください。

擬似乱数
クラス名：Ex04random

```
int number1 = (int)(Math.random() * 10) + 1;
int number2 = (int)(Math.random() * 10) + 1;
JOptionPane.showMessageDialog(null,number1 + " ×" +
number2 +"=" +number1*number2);
```

⇒number1とnumber2に1から10の値を入れて、
　掛け算するもの。

　Math.*random()*が　*0.0以上1.0未満(1は含まない)*　なので
　＋1しないと、1から10には、なりません。

次のページから、メソッドの説明です。

メソッドって？

- 簡潔に言うと、プログラムを分割して小さなプログラムの部品にする機能。

- 今まではstart(){内容}の内容に全て、書いていた。

例えば、

```
void start(){
        内容1;
          .
          .
          .
       内容100;
}
```

これを　メソッドを使えば

```
void start(){
        内容1から30まで命令を実行するメソッドの名前();       //内容1から30まで命令を実行
        内容31から60まで命令を実行するメソッドの名前();     //内容31から60まで命令を実行
        内容61から100まで命令を実行するメソッドの名前();   //内容61から100まで命令を実行
}

void   内容1から30まで命令を実行するメソッドの名前(){
        内容1から30
}

void   内容31から60まで命令を実行するメソッドの名前(){
        内容31から60
}

void   内容61から100まで命令を実行するメソッドの名前(){
        内容61から100
}
```

おおざっぱに説明するとこうなる。
このように、わけて内容1から100を実行することができる。

メソッド
クラス名：Ex11Quiz(実際に書いてください)

例えば、

名前を入力してあいさつを済ませ、これから問題を始めることを知らせ、問題を表示し答えを入力し、その答えに応じて正解かどうかを表示する。

というプログラムの一例は

```
void start() {
    String name = JOptionPane.showInputDialog("名前を入力してください");
    JOptionPane.showMessageDialog(null, name+"さん。ようこそ。問題を始めます。");
    int number1 = (int)(Math.random() * 10) + 1;
    int number2 = (int)(Math.random() * 10) + 1;
    String input = JOptionPane.showInputDialog(number1 + "×" + number2 +"=?");
    int answer = Integer.parseInt(input);
    if(answer == number1*number2){
        JOptionPane.showMessageDialog(null, "正解です");
    }else{
        JOptionPane.showMessageDialog(null, "不正解です");
    }
}
```

となると思います。

メソッド

- 先ほどの内容をメソッドを使って分割します。
- まず、作る内容を、用途ごとに分けます。

例えば、先ほどの文は

①名前を入力してあいさつを済ませ、これから問題を始めることを知らせ、②問題を表示し答えを入力し、その答えに応じて正解かどうかを表示する。

- だいたい①、②の二つに分けられると思います。

- これら①、②に用途に合ったメソッド名を考え、
  Start(){}の中に書く。

メソッド

```
void start() {
    greeting();    //名前を入力してあいさつを済ませ、これから問題を始めることを表示
    showQuiz(); //問題の答えに応じて正解かどうかを表示する
}
```

しかし、これだけでは、もちろんダメです。**void start() {}** に加え、

```
void greeting() {
    内容
}

void showQuiz() {
    内容
}
```

このように、書くことで**start(){}内で** **greeting メソッド** の内容と **showQuiz メソッド** の内容 が 実行できます。ちなみに**start(){}** 自体メソッド です。startメソッドと呼びます。

メソッド

ちなみに・・・

```
void start() {
        greeting();
        showQuiz();
        greeting();
        showQuiz();
        greeting();
        showQuiz();
}

void greeting() {
        内容
}

void showQuiz() {
        内容
}
```

このように、startメソッド内で、何度もメソッドを実行しようとすると、その数だけ実行できます。
これが、メソッドを作る利点になることもあります。

メソッド

ちなみに・・・

```
void start() {

}

void greeting() {
      内容
}

void showQuiz() {
      内容
}
```

このように、startメソッド内に何もないと、何も表示されません。
何も書かないと、startメソッド内を実行するだけです。
前のページの内容を具体的に書くと・・・

- クラス名：Ex12QuizMethod

```
void start() {
    greeting();
    showQuiz();
}

void greeting() {
    String name = JOptionPane.showInputDialog("名前を入力してください");
    JOptionPane.showMessageDialog(null, name+"さん。ようこそ。問題を始めます。");
}

void showQuiz() {
    int number1 = (int)(Math.random() * 10) + 1;
    int number2 = (int)(Math.random() * 10) + 1;
    String input = JOptionPane.showInputDialog(number1 + "×" + number2 +"=?");
    int answer = Integer.parseInt(input);
    if(answer == number1*number2){
        JOptionPane.showMessageDialog(null, "正解です");
    }else{
        JOptionPane.showMessageDialog(null, "不正解です");
    }
}
```

- 実行結果は先ほどのプログラム同じ。

流れと用語の説明

```
void start() {
    greeting();
    showQuiz();
}
```

それぞれ、greetingメソッドの呼び出し
showQuizメソッドの呼び出しという

```
void greeting() {
    String name = JOptionPane.showInputDialog("名前を入力してください");
    JOptionPane.showMessageDialog(null, name+"さん。ようこそ。問題を始めます。");
}
```

それぞれ、
Greetingメソッドの
宣言
showQuizメソッド
の宣言という

```
void showQuiz() {
    int number1 = (int)(Math.random() * 10) + 1;
    int number2 = (int)(Math.random() * 10) + 1;
    String input = JOptionPane.showInputDialog(number1 + "×" + number2 +"=?");
    int answer = Integer.parseInt(input);
    if(answer == number1*number2){
        JOptionPane.showMessageDialog(null, "正解です");
    }else{
        JOptionPane.showMessageDialog(null, "不正解です");
    }
}
```

ちなみに・・・

```
void start() {
    greeting();①
    showQuiz();②
}

void showQuiz() {②
    int number1 = (int)(Math.random() * 10) + 1;
    int number2 = (int)(Math.random() * 10) + 1;
    String input = JOptionPane.showInputDialog(number1 + "×" + number2 +"=?");
    int answer = Integer.parseInt(input);
    if(answer == number1*number2){
        JOptionPane.showMessageDialog(null, "正解です");
    }else{
        JOptionPane.showMessageDialog(null, "不正解です");
    }
}

void greeting()①{
    String name = JOptionPane.showInputDialog("名前を入力してください");
    JOptionPane.showMessageDialog(null, name+"さん。ようこそ。問題を始めます。");
}
```

- (showQuizメソッドとgreetingメソッドを入れ替えた。)
  このように書いても、結果は変わらない。

しかし・・・

```
void start() {
    showQuiz();  ①
    greeting();  ②
}

void showQuiz() {①
    int number1 = (int)(Math.random() * 10) + 1;
    int number2 = (int)(Math.random() * 10) + 1;
    String input = JOptionPane.showInputDialog(number1 + "×" + number2 +"=?");
    int answer = Integer.parseInt(input);
    if(answer == number1*number2){
        JOptionPane.showMessageDialog(null, "正解です");
    }else{
        JOptionPane.showMessageDialog(null, "不正解です");
    }
}

void greeting() ②
    String name = JOptionPane.showInputDialog("名前を入力してください");
    JOptionPane.showMessageDialog(null, name+"さん。ようこそ。問題を始めます。");
}
```

- これだと、showQuizメソッド、greetingメソッドの順に呼び出され、結果が変わる。(実際にやってみよう。)

問題1
クラス名：Q1MethodCall

ダイアログに1～6まで順に表示されるように
空欄を埋めなさい。

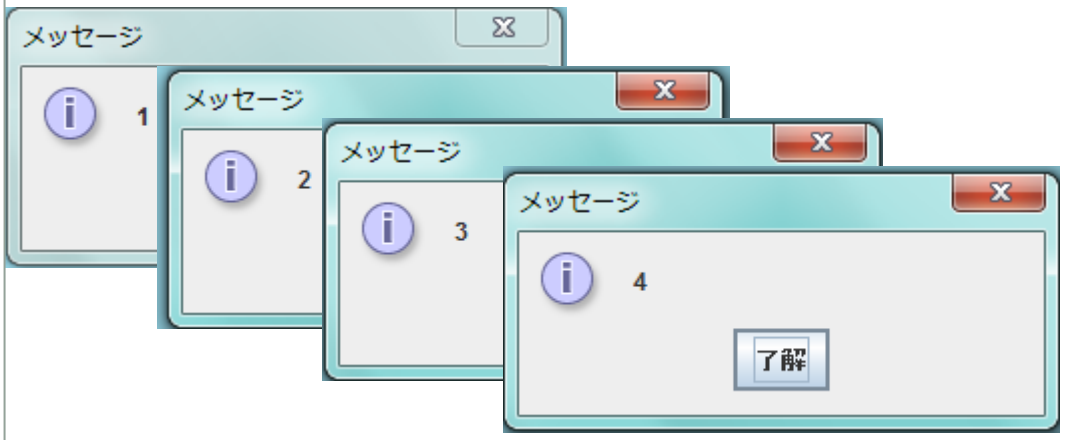
```
void start() {
```

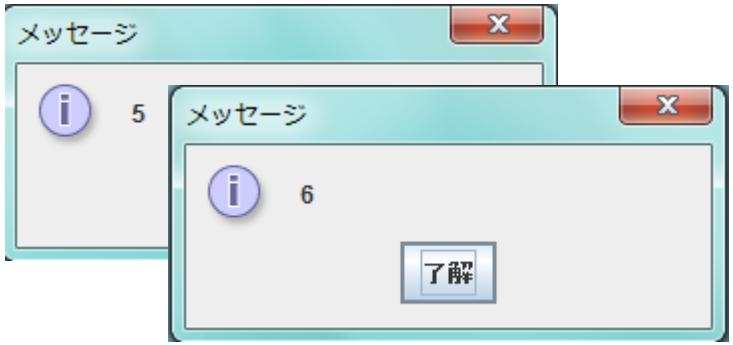
[　　]

```
}
```

```
void show34() {
    JOptionPane.showMessageDialog(null, "3");
    JOptionPane.showMessageDialog(null, "4");
}

void show12() {
    JOptionPane.showMessageDialog(null, "1");
    JOptionPane.showMessageDialog(null, "2");
}

void show6() {
    JOptionPane.showMessageDialog(null, "6");
}

void show5() {
    JOptionPane.showMessageDialog(null, "5");
}
```

実行結果





ヒント：　メソッドを呼び出すには
メソッド名();　と書く。

メソッドと引数(ひきすう)

次のプログラムを見てみよう・・・

```
void start() {
    String input = JOptionPane.showInputDialog("文字列を入力");
    showMessage();
}

void showMessage() {
    JOptionPane.showMessageDialog(null, input);
}
```

- これだとエラーが起こり、実行できません。
- 変数は宣言したメソッド内でしか使えないからです。

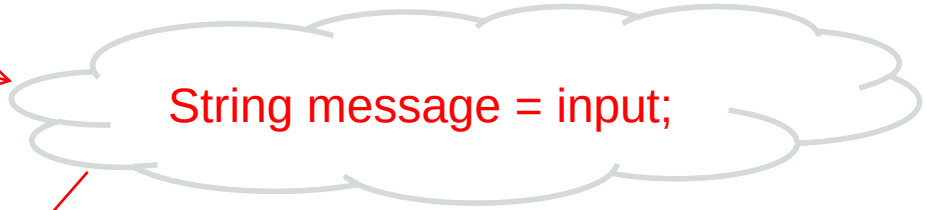
```
void start() {
    String input = JOptionPane.showInputDialog("文字列を入力");
    showMessage();
}

void showMessage() {
    JOptionPane.showMessageDialog(null, input);
}
```

変数inputが使える範囲

- では、どうすればinputを使えるのか・・・

メソッドと引数(ひきすう)

```
void start() {
    String input = JOptionPane.showInputDialog("文字列を入力");
    showMessage(input);
}

void showMessage(String message) {
    JOptionPane.showMessageDialog(null, message);
}
```



- startメソッドからshowMessageメソッドを起動する際に、「showMessage(input);」のように、messageに記憶させる値をshowMessage(input)のカッコ内のinputを指定し、messageにinputを代入する。

メソッドと引数(ひきすう)

```
void start() {
    String input = JOptionPane.showInputDialog("文字列を入力");
    showMessage(input);
}

void showMessage(String message) {
    JOptionPane.showMessageDialog(null, message);
}
```

このinputを**実引数**という

このStringを**仮引数の型**という

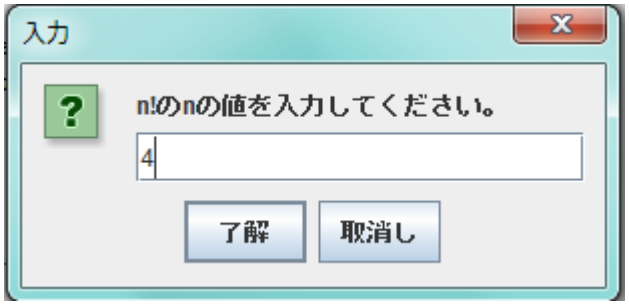
このmessageを**仮引数**という

問題2
クラス名：Q2ShowFactorial

- 次の空欄を埋めて、入力した値の階上(ex.n!=n×(n-1) ×・・・×2×1)を、表示するプログラムを作りなさい。

なおint型の値の範囲は2147483647~ -2147483648なので、nが13以上だと、正しく計算できませんが、
そのことについては、考慮しなくていいです。(0!=1,1!=1,2!=2,3!=6,4!=24になっていればok)

```
void start() {
    int number = Integer.parseInt(JOptionPane.showInputDialog("n!のnの値を入力してください。"));
    showFactorial (_____);
}

void showMessage(String message) {
    JOptionPane.showMessageDialog(null, message);
}

void showFactorial (int n) {
    [                                ]
    showMessage(n + "!=" + Factorial + "です");
}
```



ヒント

- 1+2+3+･･･+nの和を求める場合は、

```
void showsum(int n) {
        int sum = 0;
        for(int i = 1; i <= n; i++){
              sum = sum + i;
        }
        showMessage("1+2+3+･･･+n=" + sum + "です");
}
```

これをn×(n-1) ×･･･×2×1の場合を考える。

sum=n*(n+1)/2でも求まるが、問題のヒントにはならない。

複数の引数をとるメソッド

- 次のプログラムを見てみよう・・・
- クラス名：Ex13QuizMethod2

```
void start(){
    showQuiz("10+20=?",30);
    showQuiz("リンゴの色は？(1)青(2)赤",2);
    showQuiz("うるう年の1年の日数は?",366);
}

void showQuiz(String message,int correct){
    String input = JOptionPane.showInputDialog(message);
    int answer =Integer.parseInt(input);
    if(answer==correct){
        JOptionPane.showMessageDialog(null, "正解です");

    }else {
        JOptionPane.showMessageDialog(null,"不正解です" +"正解は"+correct+"です");
    }
}
```

- 仮引数を2つ以上宣言する場合には、上記のようにそれぞれの仮引数を，(カンマ)で区切ります。同様に、このメソッドを起動するには2つの実引数が必要になりますが、こちらも2つの実引数をカンマで区切って指定します。

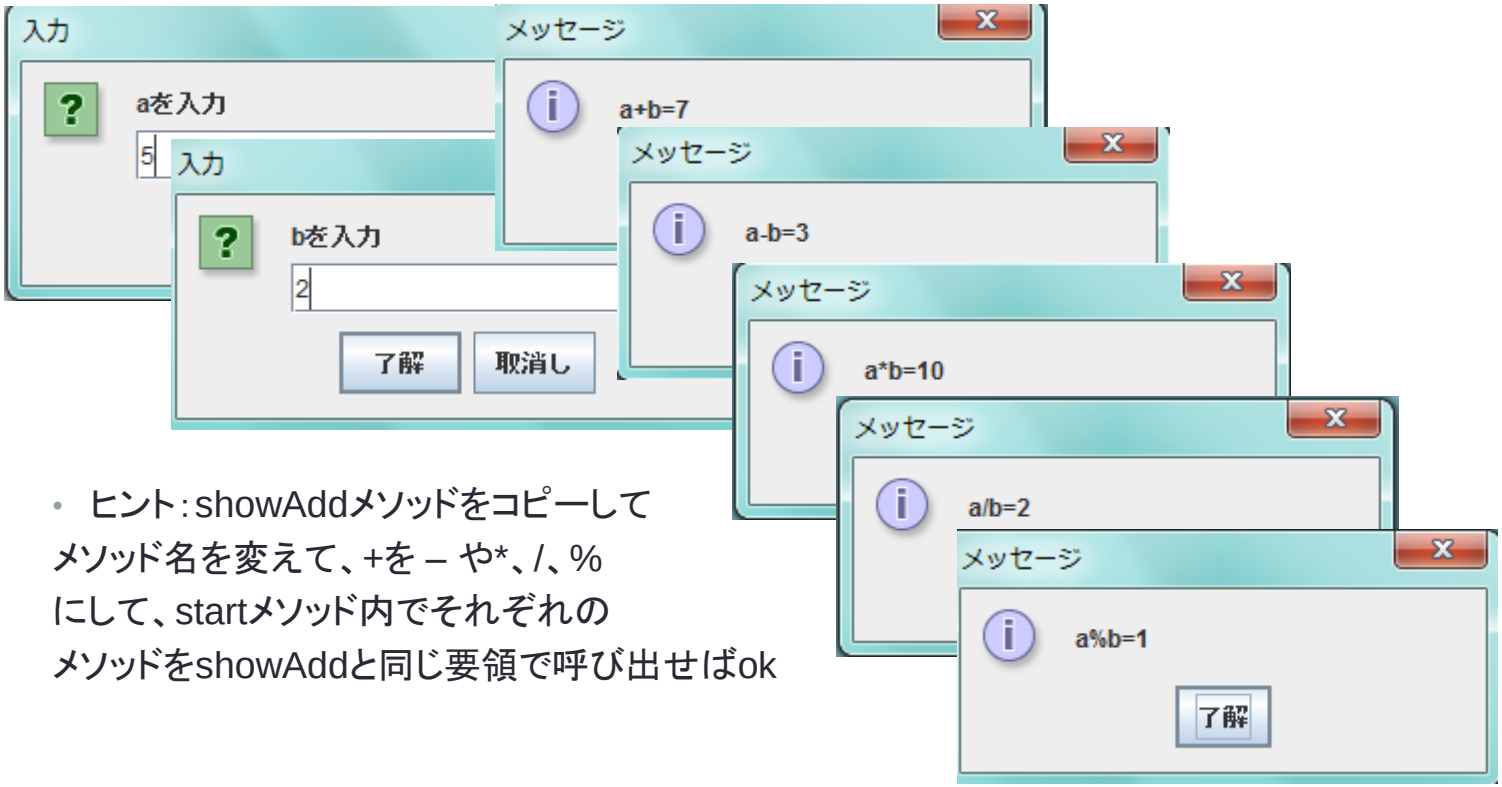
複数の引数をとるメソッド
クラス名:Ex14Add

```
void start() {
        String input1=JOptionPane.showInputDialog(“aを入力”);
        String input2=JOptionPane.showInputDialog(“bを入力”);
        Int a = Integer.parseInt(input1);
        int b = Integer.parseInt(input2);
        showadd(a,b);
}

void showAdd(int a,int b){
        showMessage("a+b="+(a+b));
}

void showMessage(String message) {
        JOptionPane.showMessageDialog(null, message);
}
```

このような使い方もできる。
ちなみに、上のように、実引数と仮引数は同じ名前でもok。

問題3
クラス名：Q3Calculation

- 前のページのEx14Addを参考に、足し算、引き算、掛け算、
  割り算(aをbで割るものとする)、余剰、を求めるメソッドを作りなさい。



- ヒント：showAddメソッドをコピーして
メソッド名を変えて、+を – や*、/、%
にして、startメソッド内でそれぞれの
メソッドをshowAddと同じ要領で呼び出せばok

関数

- 今回は結果を起動元に返すメソッドについて紹介します。
- メソッドはvoid型だけではない。
- int,double,string型などがあり、これらのメソッドは値を返す、関数のようなプログラムを作ることができる。これらのメソッドの呼びだし部分が、変数にもなる。
- 値を返すためにメソッドの最後に
  return **返す値**を付けないといけない。

```
void start() {
        //変数名  =  メソッドの名前(実引数, ･･･);  =  (returnで)返す値
    型  変数名  =  メソッドの名前(実引数, ･･･);
}

型  メソッドの名前(仮引数の型  引数, ･･･) {
      命令
return  返す値;
}
```

関数
クラス名：Ex21Function(実際に書いてください)

- 例えば(クラス名：Ex21Function)、f(x)=3x+2なら

```
void start() {
    String input1 = JOptionPane.showInputDialog("xを入力");
    int x = Integer.parseInt(input1);
    int result = f(x);
    showMessage("3x+2="+result);
}

int f(int x) {
    return 3*x+2;
}

void showMessage(String message) {
    JOptionPane.showMessageDialog(null, message);
}
```

となる。

関数

```
void start() {
    String input1 = JOptionPane.showInputDialog("xを入力");
    int x = Integer.parseInt(input1);
    int result = f(x);
    showMessage("3x+2="+result);
}

int f(int x) {
    return 3*x+2;
}

void showMessage(String message) {
    JOptionPane.showMessageDialog(null, message);
}
```

仮引数のxに
実引数のxを代入

これがf(x)メソッド

3*x+2の値を返し
f(x)= 3*x+2となる。

- x=3なら return 3*3+2 = 11 = f(x) = result ということ。
- 関数のメソッドは、return ~ で、**値を返さないといけない**。

様々な型の関数
クラス名：Function2

今まで、int型しか使わなかったが、double,String型も使うことができる。
とりあえず、実行してみてください。

```
void start() {
    showMessage(getBracket("こんにちは"));
    showMessage("10+20="+add(10,20));
    showMessage("半径rの円の面積は"+circleArea(3.9)+"です。");

}

String getBracket(String message){
    String bracketmessage = "「" + message + "」";
    return bracketmessage;
}

int add(int a, int b) {
    return a+b;
}

double circleArea(double r){
    return r*r*3.14;
}

void showMessage(String message) {
    JOptionPane.showMessageDialog(null, message);
}
```

ただし、void　は　値を返すことはできません。

次の例

- クラス名：Ex23Add

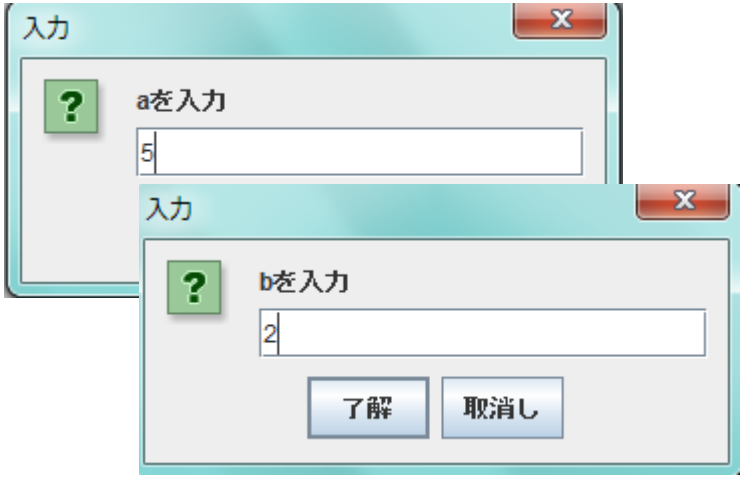
```
void start() {
        String input1 = JOptionPane.showInputDialog("aを入力");
        String input2 = JOptionPane.showInputDialog("bを入力");
        int a = Integer.parseInt(input1);
        int b = Integer.parseInt(input2);
        int result1 = add(a,b);
        showMessage("a+b="+result1);
}

int add(int a, int b) {
        Int result = a+b;
        return result;
}

void showMessage(String message) {
        JOptionPane.showMessageDialog(null, message);
}
```

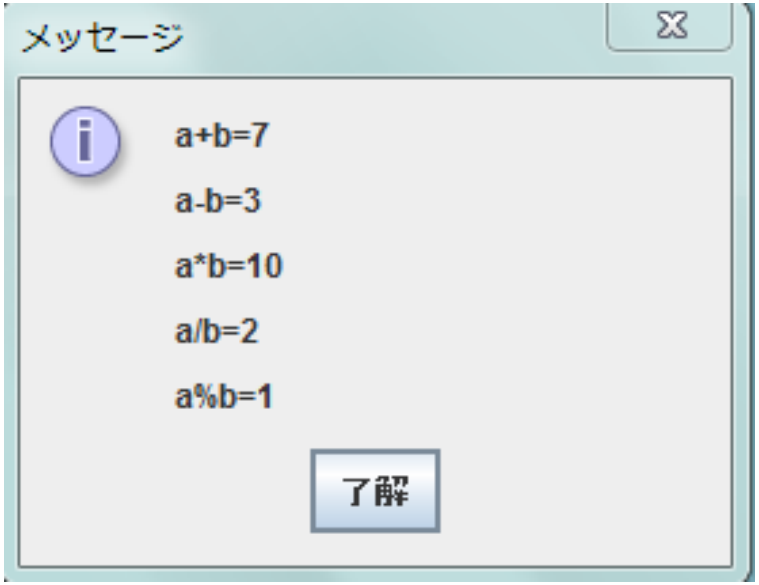
この部分は、return a+b;
でも大丈夫です。

こう書くと、前のEx14Addと同じ動作をする。

問題4
クラス名：Q4Calculation

- 前のページのEx23Addを参考に、足し算、引き算、掛け算、割り算(aをbで割るものとする)、余剰、を求めるint型のメソッドを作り、結果をまとめて表示しなさい。





Boolean

- 変数は、String、int、double、の他に　boolean　という型があります。

- booleanには、文字や数字は入りません。「成立する(真) 」ということをtrue で、「成立しない(偽) 」ということをfalse で表します。

- 実は、第二回のwhile文で出てきている、 true　もboolean型の一つです。

- さらにif 文のif(条件)の部分にも実は、if(条件== true) が省略されています。

Boolean
クラス名：Ex23boolean(実際に書いてください)

```
void start(){
    int a = 1;
    if(a >= 0 && a <= 100 == true){
        JOptionPane.showMessageDialog(null, "0~100です");
    }else{
        JOptionPane.showMessageDialog(null, "0~100ではありません。");
    }
}
```

- 普通ならif(a >= 0 && a <= 100)ですが、このように、書くこともできます。もちろん結果は “0～100”です と表示されます。
- つまり、普段書いているif文のif(条件)の部分は、if(条件== true) が省略されていたということです。
- true はbooleanという型 なので、次のように書くこともできます。もちろん結果は同じです。

```
void start(){
    int a = 1;
    boolean regular = true;
    if(a >= 0 && a <= 100 == regular){
        JOptionPane.showMessageDialog(null, "0~100です");
    }else{
        JOptionPane.showMessageDialog(null, "0~100ではありません。");
    }
}
```

Boolean
クラス名：Ex23boolean

- このように書くと、結果は変わってしまいます。“0~100ではありません。”と表示されます。

```
void start(){
    int a = 1;
    if(a >= 0 && a <= 100 == false){
        JOptionPane.showMessageDialog(null, "0~100です");
    }else{
        JOptionPane.showMessageDialog(null, "0~100ではありません。");
    }
}
```

- 次のように書いても上の結果と、変わりません。

```
void start(){
    int a = 1;
    boolean regular = false;
    if(a >= 0 && a <= 100 == regular){
        JOptionPane.showMessageDialog(null, "0~100です");
    }else{
        JOptionPane.showMessageDialog(null, "0~100ではありません。");
    }
}
```

Boolean型のメソッド
クラス名：Ex24booleanMethod(実際に書いてください)

- もちろん、メソッドを使って書くこともできます。

```
void start(){
    int a = 1;
    if(regular(a)){
        JOptionPane.showMessageDialog(null, "0~100です");
    }else{
        JOptionPane.showMessageDialog(null, "0~100ではありません。");
    }
}

boolean regular(int a){
    if(a >= 0 && a <= 100){
        return true;
    }else{
        return false;
    }
}
```

- aの値を定義し、regularメソッドに実引数aを入れて、aが0～100ならtrue、それ以外ならflase を　返します。
- If文の中身が　regular(a)　しか、書いていないことに違和感を感じる人は
- `if(regular(a) == true){`　と書く良いかもしれません。

Boolean型のメソッド
クラス名：Ex24booleanMethod

- ちなみに

```
void start(){
    int a = 1;
    if(!regular(a)){
        JOptionPane.showMessageDialog(null, "0~100ではありません。");
    }else{
        JOptionPane.showMessageDialog(null, "0~100です");
    }
}

boolean regular(int a){
    if(a >= 0 && a <= 100){
        return true;
    }else{
        return false;
    }
}
```

- regular(a)の前に！をつけると、regularをtrueならfalseにfalseならtrueにする。
- この場合、regularはtrueなので!regulerはfalseになる。
- あくまで、省略されているのは　if(!regular ==　true)

注意

```
void start(){
    int a = 1;
    if(regular(a)){
        JOptionPane.showMessageDialog(null, "0~100です");
    }else{
        JOptionPane.showMessageDialog(null, "0~100ではありません。");
    }
}

boolean regular(int a){
    if(a >= 0 && a <= 100){
        return true;
    }
}
```

- これだと、booleanメソッドのaが0~100以外のときに、何をreturnするか書いていないので、エラーになります。

問題5
クラス名：Q5QuizMethod

- 空欄を埋めて、quizを完成させなさい。

```
void start(){
        showQuiz("10+20=?",30);
        showQuiz("リンゴの色は？¥n(1)青(2)赤",2);
        showQuiz("うるう年の1年の日数は?",366);
}

void showQuiz(String message,int correct){
        int answer = getInput( _______ );
        if  (isInvalid( ______ ,  _______ )){
                JOptionPane.showMessageDialog(null, "正解です");
        }else{
        JOptionPane.showMessageDialog(null,"不正解です" +"正解は"+correct+"です");
        }
}
```

.
.
.

続き

```
int getInput(String message){
    String input = JOptionPane.showInputDialog( _______ );
    int answer=Integer.parseInt(input);
    return   _____;
}

boolean isInvalid(int answer,int correct){
    if( _____   ==   _____   ){
        return   ____;
    }
    else{
        return   ____;
    }
}
```

発展問題6 (問題5まで終わって暇な人用。かなり難しいです。)
クラス名：Q6TakeStone

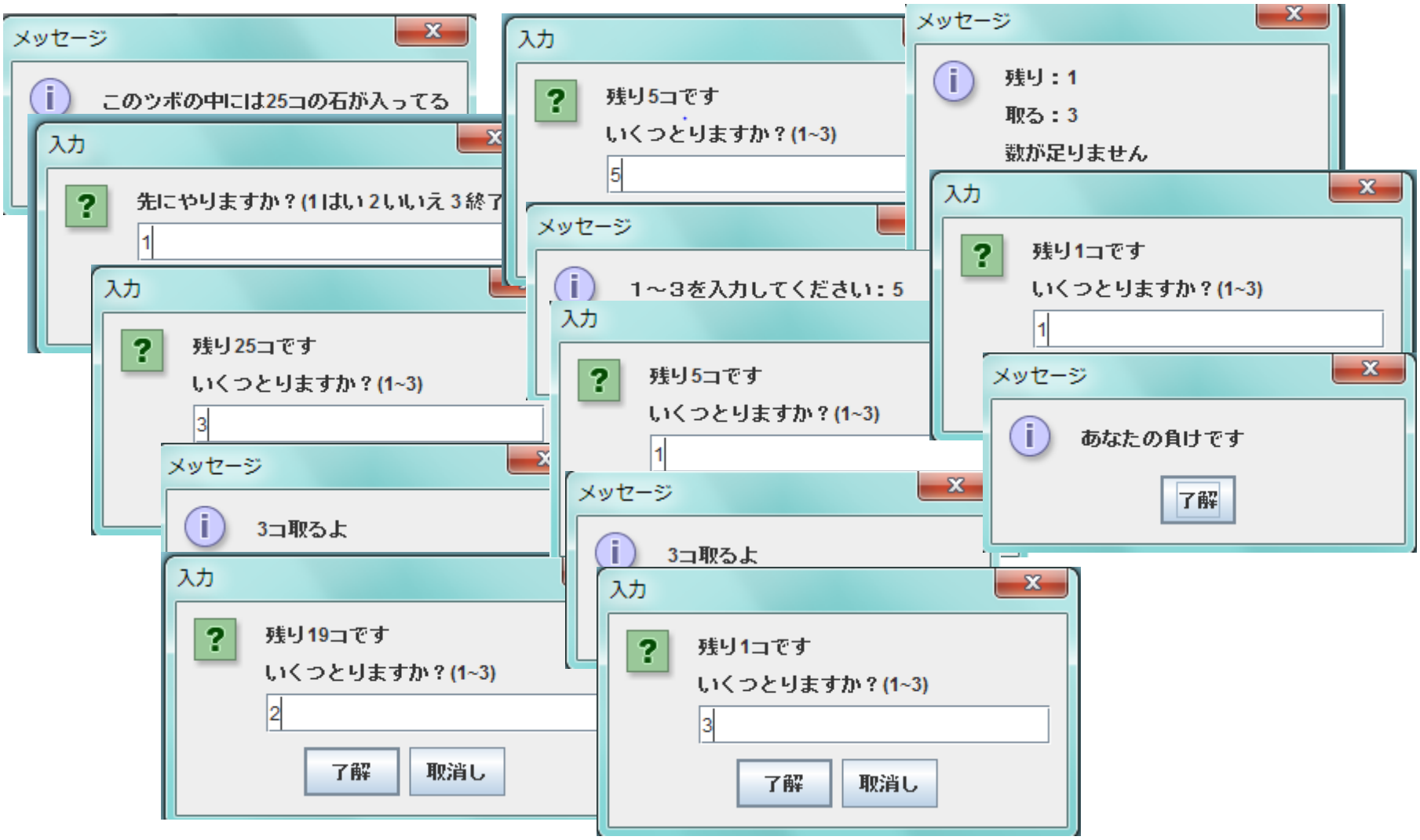
- 石取りゲーム
- 石取りゲームとは一山の石(数は20~40ぐらい)から順に石を取り・・・

  最後に取った人が負けになるというものです。

  一回に取れる石の数は最大3コで、パスはできない。

自分は取る個数を選択し、コンピュータはランダムで選択する石取りゲームを実装しなさい。

次のページに実行例


メッセージ
このツボの中には25コの石が入ってる
入力
先にやりますか？(1はい 2いいえ 3終了
1
入力
残り25コです
いくつとりますか？(1~3)
3
メッセージ
3コ取るよ
入力
残り19コです
いくつとりますか？(1~3)
2
了解
取消し
入力
残り5コです
いくつとりますか？(1~3)
5
メッセージ
1～3を入力してください：5
入力
残り5コです
いくつとりますか？(1~3)
1
メッセージ
3コ取るよ
入力
残り1コです
いくつとりますか？(1~3)
3
了解
取消し
メッセージ
残り：1
取る：3
数が足りません
入力
残り1コです
いくつとりますか？(1~3)
1
メッセージ
あなたの負けです
了解

ヒント

- startメソッド
- 自分が石を取る個数(値を返す)int型のtakeメソッド(引数　石の数)
- 相手が石を取る個数(値を返す)int型のrandomtakeメソッド(引数　石の数)
- 質問の答えの数値を返すint型のgetInputメソッド(引数　石の数と、質問文であるstring message)

自分なりに、工夫してメソッドを作ってもいいと思う。

ヒント

void start(){
まず、石の数を乱数で20から40コの間で定義する。
石の数を表示
自分の答えの変数にgetinputを使って先攻か後攻かを決める
条件分岐で先攻なら、
石の数から自分の取る石の数を引く
そして、while文で相手と交互に石を取り、引いて0になった人が負け
}
int randomtake(int stone) {
1コから3コ取る
石の数より取る数が多いなら
石の数だけとるようにして
引く数を石の数に 「〜コ取るよ」 と発言
}

int take(int stone) {
getinput使って石を取る。残りの石の数も表示すべき。
}

int getInput(int stone,String message){
　ここで、先攻後攻や石の数を聞く。while文で、1から3なら値を返し、
　石の数より答えが多い場合や1から3以外の値が入力されたら、聞き返すようにした方がいい。
}

発展問題7(発展問題6まで終わってさらに暇な人用)

クラス名 : Q8TakeStone2

発展問題6のrandomtakeメソッドを書き換えて、強くしてください。

ヒント :

randomメソッドを書きかえる。
残り5コ状態で相手の番にすれば、勝てる。
残り5コ状態で相手の番にするには、
残り9コ状態で相手の番にする、
残り9コ状態で相手の番にするには・・・
つまり、stone をstone/4あまり1にするように、取ればいい。

次回について

- おつかれさまでした。
- 次回は、第一回から第三回までの復習です。
- 次は**5/10(木)**ですよ！