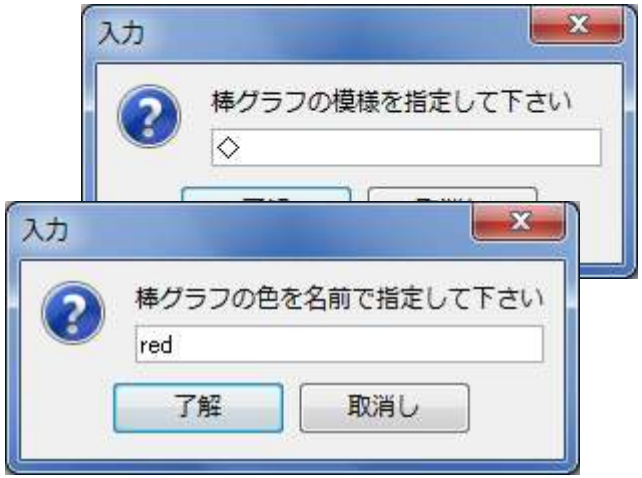
プログラミング入門2

第2回　メソッドの利用

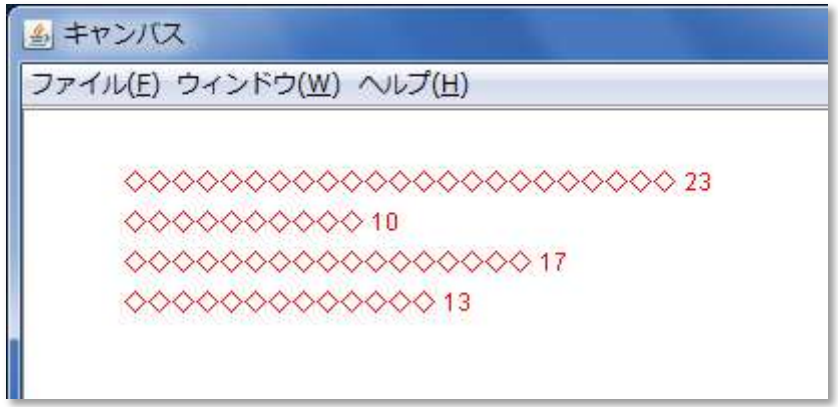
本日の主な目的(題材)

- 棒グラフの模様と、色名を指定して、色つきの棒グラフを作成する(問題32)
  - 模様を好きに選べる。
  - 色の名前で表示色を指定できる。

入力



出力



int[] a = { 23, 10, 17, 13 };

今日の講義のテーマ（メソッドの様々な使い方）

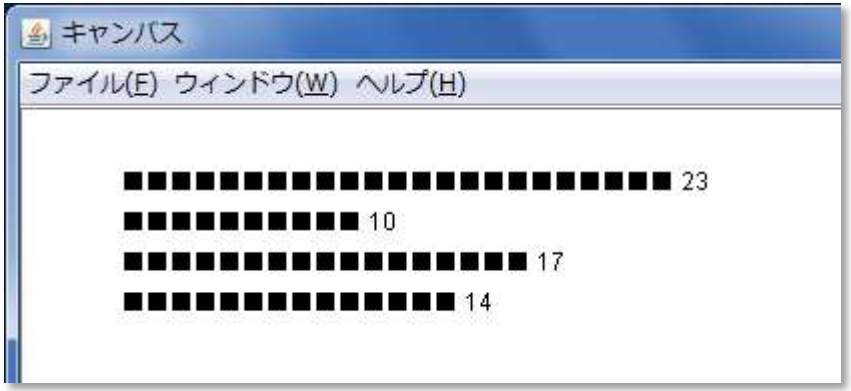
- メソッドの作成例(1)
  パラメータ化による機能の一般化
- メソッドの作成例(2)
  対応表による単純なデータの変換
- メソッドの作成例(3)
  演算によるデータの変換
  少し複雑な処理を用いたデータの変換
  再帰メソッド(1)

メソッドを効果的に使う方法やパターンは様々であるが、本日の講義では、いくつか代表的なものをとりあげる。Java 1の「メソッド」の講義の内容も含まれるので、そちらも参照すると理解がより深まる。

□復習　前回例題34

問題：正の整数を引数nをとり、n個だけ「■」を横に並べた文字列を返すメソッドを作成せよ。また、実行例を参考にその挙動を確かめよ。

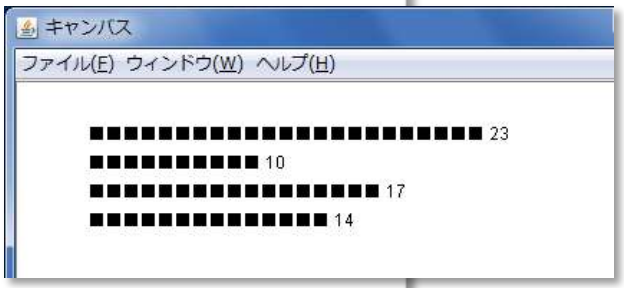
返り値の型	メソッド名(引数)	機能
String	makeBar(int n)	■をn個横に並べた文字列を返す。



（クラス名： Ex34MakeBar）

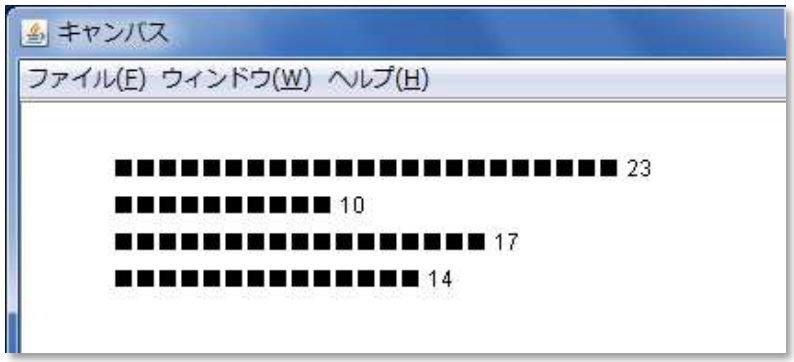
前回
例題34

```
1  package j2.lesson01;
2
3  import gpjava.Canvas;
4
5  public class Ex34MakeBar {
6      public static void main(String[] args) {
7          new Ex34MakeBar().start();
8      }
9
10     void start() {
11         int a1 = 23;
12         int a2 = 10;
13         int a3 = 17;
14         int a4 = 14;
15
16         Canvas.show();
17         String result;
18         result = makeBar(a1) + " " + a1;
19         Canvas.drawString(50, 40, result);
20         result = makeBar(a2) + " " + a2;
21         Canvas.drawString(50, 60, result);
22         result = makeBar(a3) + " " + a3;
23         Canvas.drawString(50, 80, result);
24         result = makeBar(a4) + " " + a4;
25         Canvas.drawString(50, 100, result);
26     }
27
28     String makeBar(int n) {
29         String result = "";
30         String tile ="■";
31
32         for(int i = 0; i < n; i++) {
33             result = result + tile;
34         }
35
36         return result;
37     }
38 }
```

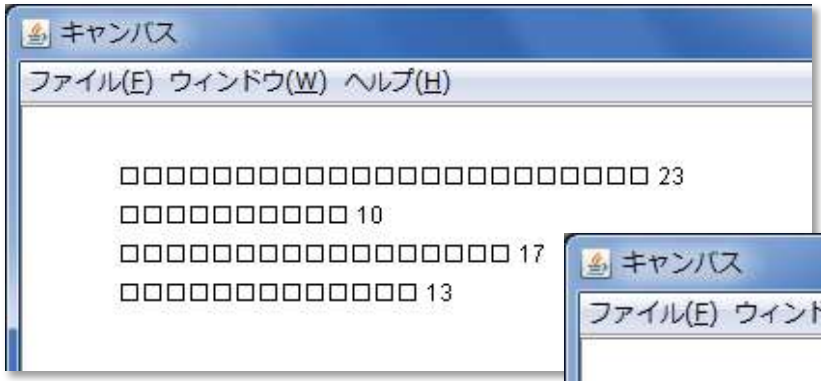


makeBar(int n)
を計4回呼び出して
いる。

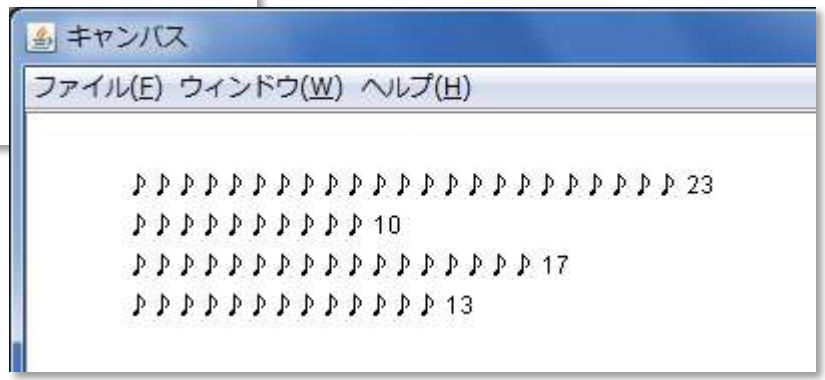
前回例題34のmakeBar(int n)を拡張することを考える。■以外の好きな模様も表示できるようにしたい。



だけではなく、



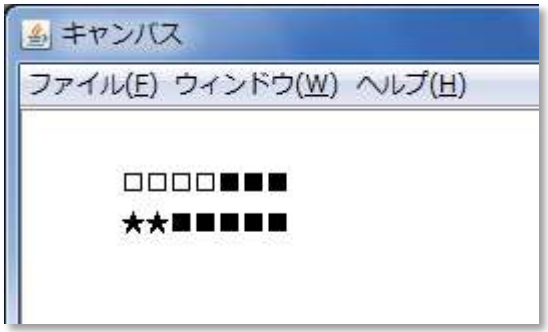
とか、



■例題11

問題：下記の二つのmakeBarを作成せよ。最初のメソッドは、任意の文字列を棒グラフの模様として指定できる。メソッドの多重定義(overloading)を用いよ。

返り値	メソッド名(引数)	機能
String	makeBar(int n, String tile)	tileで指定した文字列を横にn個並べたものを返す。
String	makeBar(int n)	■を横にn個並べたものを返す。(すなわち makeBar(n, “■”))

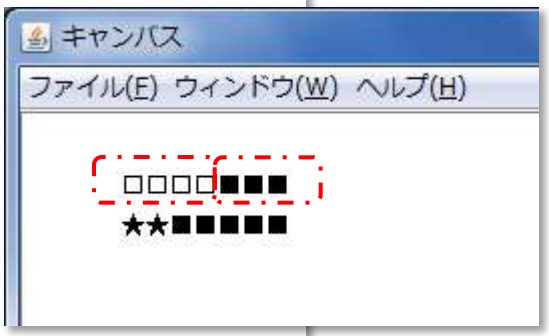


(クラス名：Ex11MakeBar)

例題11

```
1  package j2.lesson02;
2
3  import gpjava.Canvas;
4
5  public class Ex11MakeBar {
6      public static void main(String[] args) {
7          new Ex11MakeBar().start();
8      }
9
10     void start() {
11         Canvas.show();
12         String result = "";
13         result = makeBar(4, "□");
14         result += makeBar(3);
15         Canvas.drawString(50, 40, result);
16
17         result = "";
18         result = makeBar(2, "★");
19         result += makeBar(5);
20         Canvas.drawString(50, 60, result);
21     }
22
23     String makeBar(int n, String tile) {
24         String result = "";
25         for (int i = 0; i < n; i++) {
26             result += tile;
27         }
28         return result;
29     }
30
31     String makeBar(int n) {
32         return makeBar(n, "■");
33     }
34 }
```

両者makeBarだが
引数の部分が異なるので
別個のメソッドとして定義できる(overloading)



前回
例題34

両者の違いに注目

```
27
28  String makeBar(int n) {
29      String result = "";
30      String tile ="■";
31
32      for(int i = 0; i < n; i++) {
33          result = result + tile;
34      }
35
36      return result;
37  }
38 }
```

String tile
が、パラメータ（引数）
の部分に移った。
→■以外の好きな模様もメソッド
の外から与えることができる。
（内部で使用している変数を
パラメータ化した）

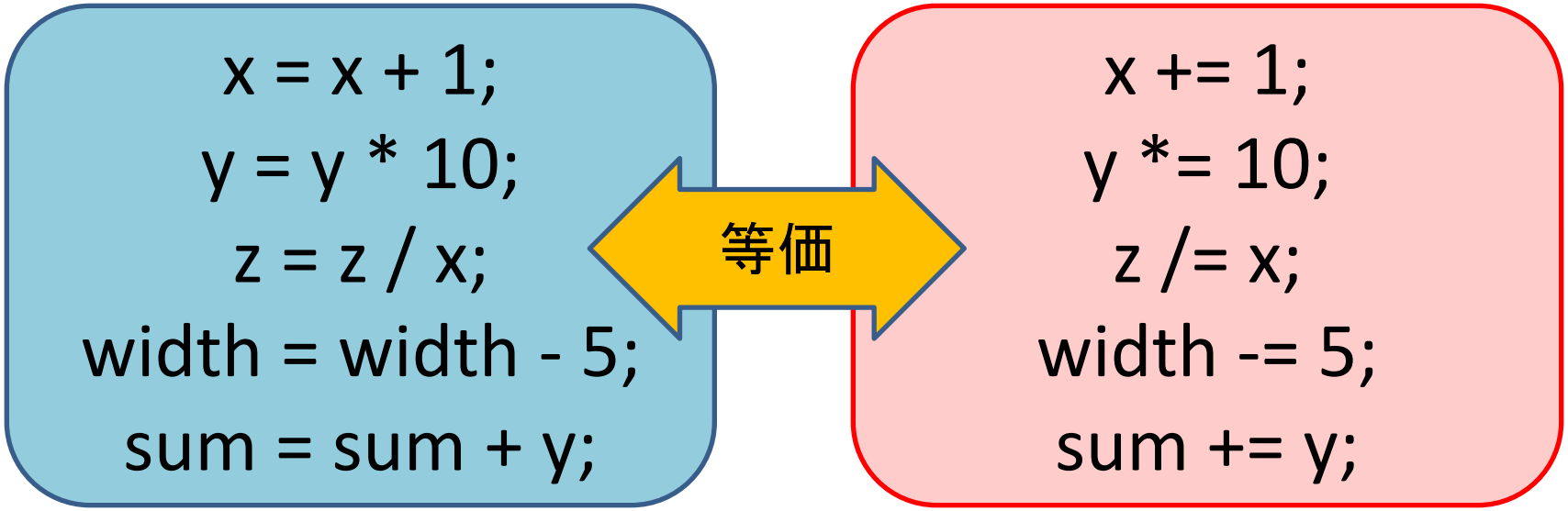
x = x + 1 のように、
変数に演算を加えた結果を
同じ変数に代入する時、
x += 1
のような形の複合代入演算子
を利用できる

例題11

```
22
23  String makeBar(int n, String tile) {
24      String result = "";
25      for (int i = 0; i < n; i++) {
26          result += tile;
27      }
28      return result;
29  }
30
```

複合代入演算子について

- 加減乗除演算子、論理演算子に対して、複合代入演算子が利用できる。文字列の連結に使われる + 演算子にも利用できる。



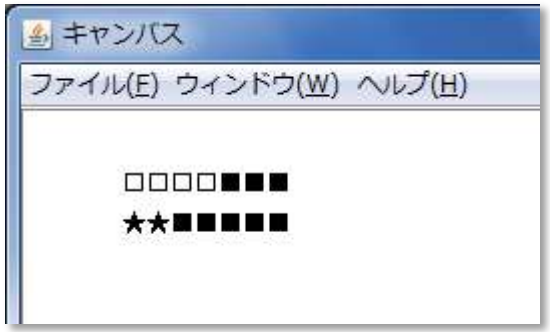
特に、x += 1, y -= 1 という1の増減については、x++, y-- と表現できる

このメソッドを使う場合は、tileに表示させたい模様を指定する。

返り値	メソッド名(引数)	機能
String	makeBar(int n, String tile)	tileで指定した文字列を横にn個並べたものを返す。
String	makeBar(int n)	■を横にn個並べたものを返す。(すなわち makeBar(n, “■”))

こちらは表示させる模様を指定しなくてよい。指定しない場合は「■」を模様とする。
（デフォルトの模様が「■」である。）

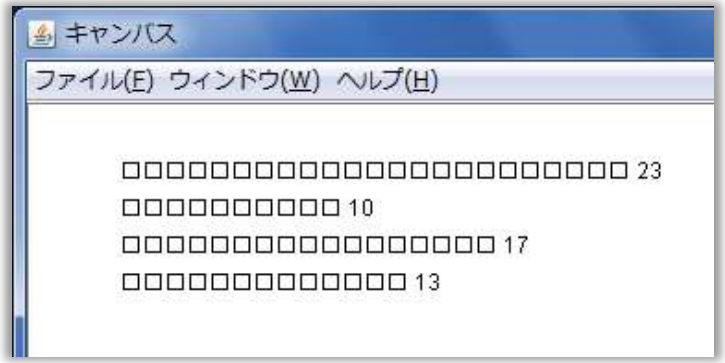
```
12          String result = "";
13          result = makeBar(4, "□");
14          result += makeBar(3);
15          Canvas.drawString(50, 40, result);
16
17          result = "";
18          result = makeBar(2, "★");
19          result += makeBar(5);
20          Canvas.drawString(50, 60, result);
```



■例題12

問題: (例題11の応用) 例題11のmakeBar(int n, String tile)を用いることで、模様を指定して、棒グラフの表示をするプログラムを作成せよ。

返り値	メソッド名(引数)	機能
String	*makeBar(int n, String tile)*	*tileで指定した文字列を横にn個並べたものを返す。(前の物と同じ)*
void	showGraph(String tile, int[] anArray)	模様がtileである棒グラフをダイアログで表示する。棒グラフの各値はanArrayに対応する。



(クラス名：Ex12MakeGraph)

例題12

```
1  package j2.lesson02;
2
3  import gpjava.Canvas;
4  import javax.swing.JOptionPane;
5
6  public class Ex12ShowGraph {
7      public static void main(String[] args) {
8          new Ex12ShowGraph().start();
9      }
10
11     void start() {
12         String message = "棒グラフの模様を指定して下さい";
13         String tile = JOptionPane.showInputDialog(message);
14         int[] a = { 23, 10, 17, 13 };
15         showGraph(tile, a);
16     }
17
18     String makeBar(int n, String tile) {
19         String result = "";
20         for (int i = 0; i < n; i++) {
21             result += tile;
22         }
23         return result;
24     }
25
26     // このプログラムでは用いない
27     String makeBar(int n) {
28         return makeBar(n, "■");
29     }
30
31     void showGraph(String tile, int[] anArray) {
32         Canvas.show();
33         String result = "";
34         for (int i = 0; i < anArray.length; i++) {
35             result = makeBar(anArray[i], tile) + " " + anArray[i];
36             Canvas.drawString(50, 40 + i * 20, result);
37         }
38     }
39 }
```

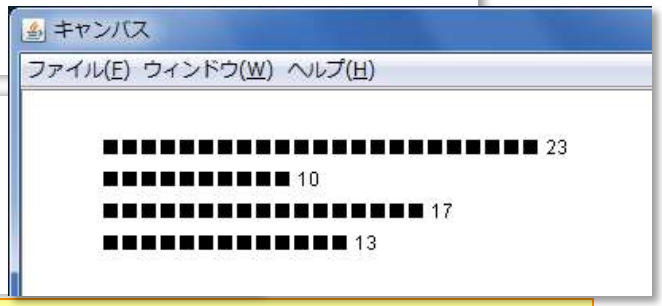
（前の例題と同じ）

模様を自由に指定できる。

前回
例題44

```
26    void showGraph(int[] anArray) {
27        Canvas.show();
28        String result;
29
30        for(int i = 0; i < anArray.length; i++) {
31            result = makeBar(anArray[i]) + " " + anArray[i];
32            Canvas.drawString(50, 40 + i * 20, result);
33        }
34    }
```

```
10    void start() {
11        int[] a = {23, 10, 17, 13};
12        showGraph(a);
13    }
```

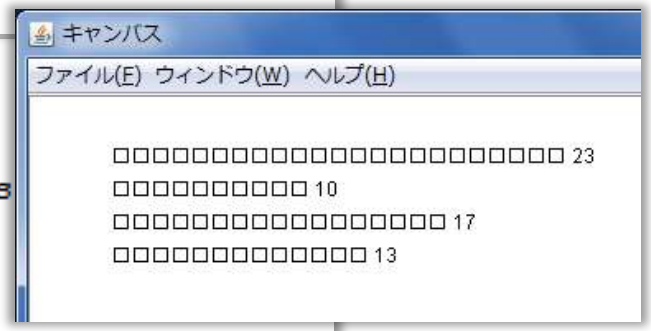


両者の違いに注目。前回は「■」のみでしか棒グラフを作れないが、
今回は、模様を自由に指定することができる。
→例題12は前回のものと比べ、**より多くの場合に対処可能**な（すなわち、**より一般化された**）プログラムといえる。

例題12

```
31    void showGraph(String tile, int[] anArray) {
32        Canvas.show();
33        String result = "";
34        for (int i = 0; i < anArray.length; i++) {
35            result = makeBar(anArray[i], tile) + " " + anArray[i];
36            Canvas.drawString(50, 40 + i * 20, result);
37        }
```

```
10
11    void start() {
12        String message = "棒グラフの模様を指定して下さい";
13        String tile = JOptionPane.showInputDialog(mes
14        int[] a = { 23, 10, 17, 13 };
15        showGraph(tile, a);
16    }
```



ここまでのまとめ

- プログラムを色々な場合に対応できるように考えることは重要。その一つのやり方が、パラメータ化によるメソッドの一般化（抽象化）。
  - ただし、一般化（抽象化）しすぎると何をやっているのかが分かりにくくなることがある。具体的なケースが想像できるようなメソッドの記述になるよう設計するのがよい。

今日の講義のテーマ

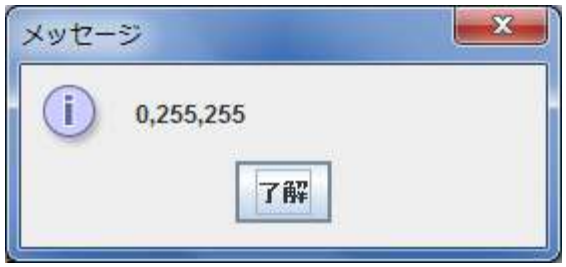
- メソッドの作成例(1)
  - パラメータ化による機能の一般化
- メソッドの作成例(2)
  - 対応表による単純なデータの変換
- メソッドの作成例(3)
  - 演算によるデータの変換
  - 少し複雑な処理を用いたデータの変換
  - 再帰メソッド(1)

■例題31

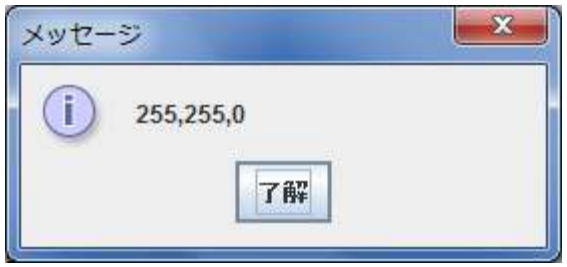
問題:色名を対応するRGB値の数字文字列に変換するメソッドを作成せよ。

返り値	メソッド名(引数)	機能
String	getColorByName (String colorName)	基本の8色の色名に対応するRGBの数字列を返す。 例:”black” -> “0,0,0”

“cyan”に対応する色番号



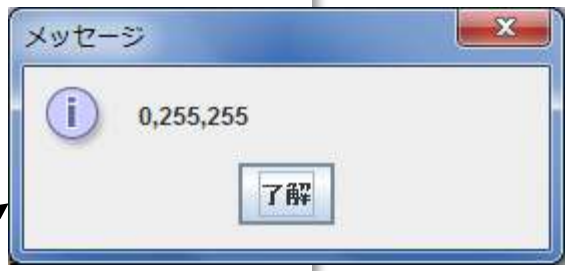
“yellow”に対応する色番号

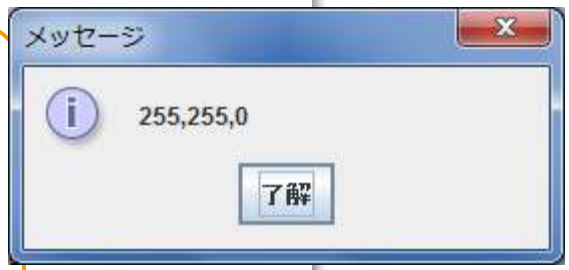


(クラス名： Ex31GetColorByName)

例題31

```
1  package j2.lesson02;
2
3  import javax.swing.JOptionPane;
4
5  public class Ex31GetColorByName {
6      public static void main(String[] args) {
7          new Ex31GetColorByName().start();
8      }
9
10     void start() {
11         JOptionPane.showMessageDialog(null, getColorByName("cyan"));
12         JOptionPane.showMessageDialog(null, getColorByName("yellow"));
13     }
14
15     String getColorByName(String colorName) {
16         String black   = "0,0,0";
17         String blue    = "0,0,255";
18         String green   = "0,255,0";
19         String red     = "255,0,0";
20         String cyan    = "0,255,255";
21         String magenta = "255,0,255";
22         String yellow  = "255,255,0";
23         String white   = "255,255,255";
24
25         if ("black".equals(colorName)) {
26             return black;
27         } else if ("blue".equals(colorName)) {
28             return blue;
29         } else if ("red".equals(colorName)) {
30             return red;
31         } else if ("green".equals(colorName)) {
32             return green;
33         } else if ("magenta".equals(colorName)) {
34             return magenta;
35         } else if ("yellow".equals(colorName)) {
36             return yellow;
37         } else if ("cyan".equals(colorName)) {
38             return cyan;
39         } else if ("white".equals(colorName)) {
40             return white;
41         } else { // 登録されていない色の場合は白を返す。
42             return white;
43         }
44     }
45 }
```





例題31補足：文字列同士の比較

```
if ("black".equals(colorName)) {
    return black;
} else if ("blue".equals(colorName)) {
    return blue;
} else if ("red".equals(colorName)) {
```

この場合、

```
if (colorName.equals("black") {
   return black;
} …
```

という書き方も可能。

文字列s1と文字列s2の内容が
等しいかどうかを確かめる時には、
　s1.equals(s2)
を用いる。等しいときにはtrue、
そうでないときは、falseを返す。
s1==s2では、うまくいかない。
(理由は、他の機会に）

```
15    String getColorByName(String colorName) {
16        String black   = "0,0,0";
17        String blue    = "0,0,255";
18        String green   = "0,255,0";
19        String red     = "255,0,0";
20        String cyan    = "0,255,255";
21        String magenta = "255,0,255";
22        String yellow  = "255,255,0";
23        String white   = "255,255,255";
24
25        if ("black".equals(colorName)) {
26            return black;
27        } else if ("blue".equals(colorName)) {
28            return blue;
29        } else if ("red".equals(colorName)) {
30            return red;
31        } else if ("green".equals(colorName)) {
32            return green;
33        } else if ("magenta".equals(colorName)) {
34            return magenta;
35        } else if ("yellow".equals(colorName)) {
36            return yellow;
37        } else if ("cyan".equals(colorName)) {
38            return cyan;
39        } else if ("white".equals(colorName)) {
40            return white;
41        } else { // 登録されていない色の場合は白を返す。
42            return white;
43        }
```

下記のような変換表から、色名に対応する色番号を求める。
（表を「探索する」(searching)という）

色名	番号
black	0,0,0
blue	0,0,255
green	0,255,0
red	255,0,0
cyan	0,255,255
magenta	255,0,255
yellow	255,255,0
white	255,255,255

色名を番号に変換
(色番号を色の名前から得る)

cyan → getColorByName → 0,255,255

今日の講義のテーマ

□メソッドの作成例(1)
　パラメータ化による機能の一般化
□メソッドの作成例(2)
　対応表による単純なデータの変換
　メソッドの作成例(3)
　演算によるデータの変換
　少し複雑な処理を用いたデータの変換
　再帰メソッド(1)

■例題41

問題:絶対値を求めるメソッドおよび2乗を求めるメソッドを作成せよ。

返り値	メソッド名(引数)	機能
int	absolute(int n)	引数の絶対値を返す
int	square(int n)	引数の2乗を返す

(クラス名： Ex41AbsoluteSquare)

例題41

```
1  package j2.lesson02;
2
3  import javax.swing.JOptionPane;
4
5  public class Ex41AbsoluteSquare {
6      public static void main(String[] args) {
7          new Ex41AbsoluteSquare().start();
8      }
9
10     void start() {
11         JOptionPane.showMessageDialog(null, "-10の絶対値は"+ absolute(-10));
12         JOptionPane.showMessageDialog(null, "-10の2乗は"+ square(-10));
13     }
14
15     int absolute(int n) {
16         if (n < 0) {
17             return -n;
18         } else {
19             return n;
20         }
21     }
22
23     int square(int n) {
24         return n * n;
25     }
26
27 }
```

return (n < 0 ? –n : n);
とも書ける。

条件演算子と呼び、
b ? x: y
と書くと、bが真ならx,
bが偽ならyになる

（数学的な）関数のメソッドでの表現

```
23      int square(int n) {
24          return n * n;
25      }
```

```
int square(int n) {
  if ( n == 0) {
      return 0;
  } else if  (n == 1) {
      return 1;
  } else if (n == 2) {
      return 4;
 } else if (n == 3) {
     return 9;
 } …
```

引数 x	返り値 square(x)
…	…
-2	4
-1	1
0	0
1	1
2	4
3	9
…	…

もちろん、このような書き方はしないが、
原理的にはこのようにも書ける。
(intで表せる数の種類は有限であるから
「if (n == ?)」による場合分けを書き尽くすことができる。）

両者とも、結果的には、
下記のような変換表（対応）から、
パラメータに対する関数の値を求める、
という機能を持つ。数学的な「関数」も
「対応」の１種（多対１対応）である。

■例題42

問題：１０進数を２進表現に変換するメソッドを作成せよ

返り値	メソッド名(引数)	機能
String	toBinary(int n)	引数の2進数表現を返す。

(クラス名：Ex42ToBinary)

例題42

```
1  package j2.lesson02;
2
3  import javax.swing.JOptionPane;
4
5  public class Ex42ToBinary {
6      public static void main(String[] args) {
7          new Ex42ToBinary().start();
8      }
9
10     void start() {
11           JOptionPane.showMessageDialog(null, "10進数の20を2進数に変換すると"+ toBinary(20));
12     }
13
14     String toBinary(int n){
15         String result = "";
16         do{
17             if(n%2 == 1)
18                 result = "1" + result;
19             else
20                 result = "0" + result;
21             n = n / 2;
22         }while(n != 0);
23
24         return result;
25     }
26 }
```

変換処理のメソッド

```
String toBinary(int n){
    String result = "";
    do{
        if(n%2 == 1)
            result = "1" + result;
        else
            result = "0" + result;
        n = n / 2;
    }while(n != 0);

    return result;
}
```

引数 n	返り値 toBinary(n)
0	“0”
1	“1”
2	“10”
3	“11”
4	“100”
…	….

```
String toBinary(int n) {
  if ( n == 0) {
      return “0”;
  } else if  (n == 1) {
      return “1”;
  } else if (n == 2) {
      return “10”;
  } else if (n == 3) {
      return “11”;
  } …
```

この場合も、両者とも、
上記のような変換表（対応）から、
パラメータに対する変換後の値を求める
という機能を持つ。

もちろん、このような書き方はしないが、
原理的にはこのようにも書ける。

今日の講義のテーマ

□メソッドの作成例(1)
　パラメータ化による機能の一般化
□メソッドの作成例(2)
　対応表による単純なデータの変換
□メソッドの作成例(3)
　演算によるデータの変換
　少し複雑な処理を用いたデータの変換
　再帰メソッド(1)

■例題44

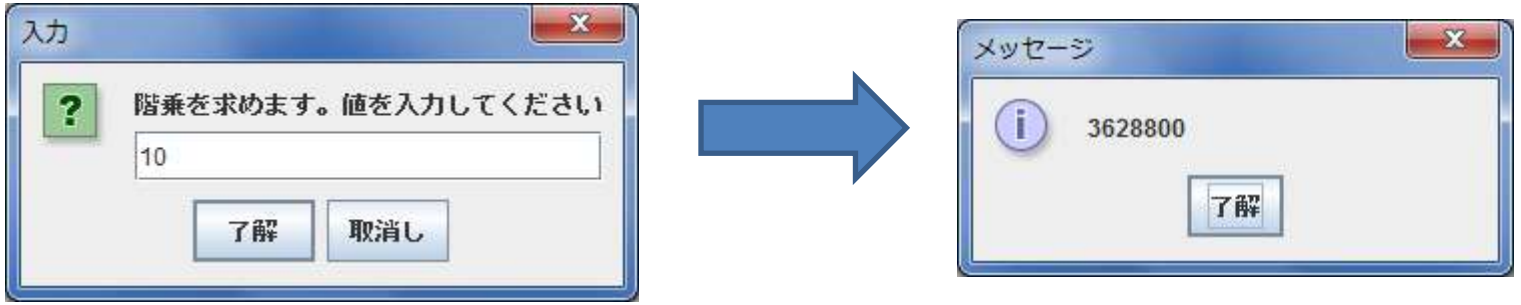
問題:階乗を求めるメソッドを作成せよ。nの階乗n!を求める階乗の式は、

n! = 1　　(n = 0の場合)

　= n × (n-1)!　(n >= 1の場合)

と表せる。

返り値	メソッド名(引数)	機能
int	factorial(int n)	引数の階乗の値を返す。



(クラス名 : Ex44Factorial)

階乗の計算(左のプログラムを参考に空欄を埋めよ)

```
int factorial(int n) {
  int result = 1;
  if ( n == 0) { return result; }
  result = 1 * result ;
  if (n == 1 ) { return result; }
  result = 2 * result;
  if (n == 2 ) { return result; }
  result = 3 * result;
  if (n == 3 ) { return result; }
  ….
  result = x  * result;
  if (n == x ) { return result; }
  ….
}
```

1

1*1

2*(1*1)

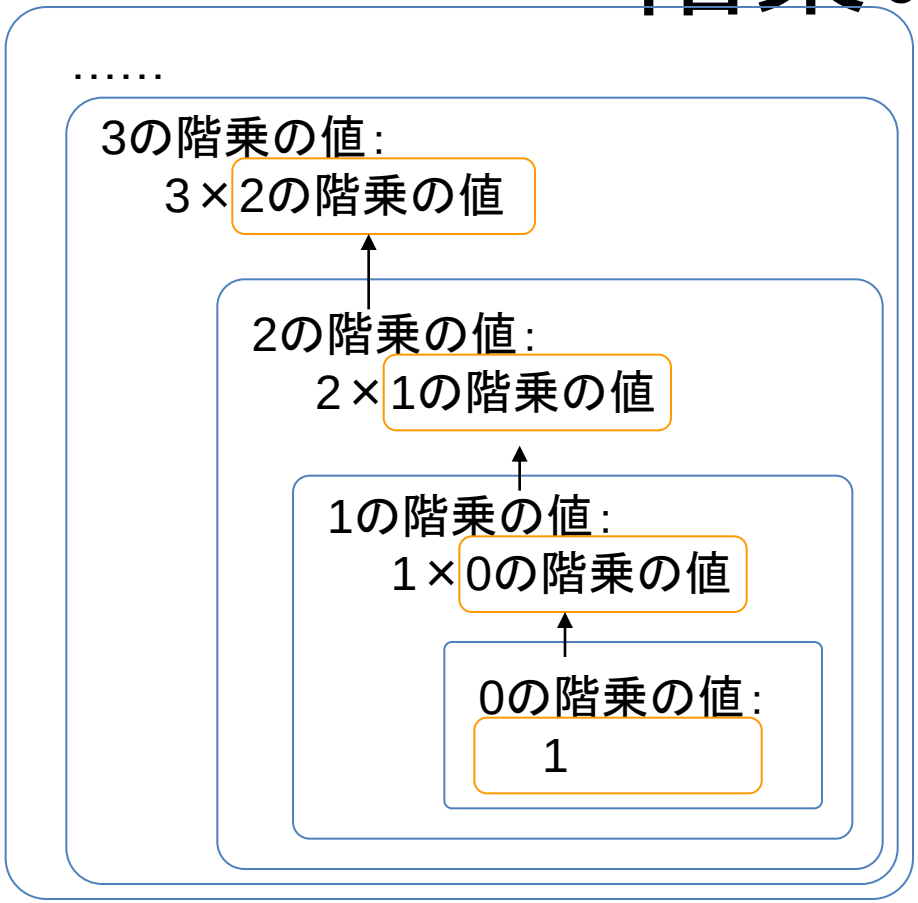
3*(2 * (1 * 1)

x*(x-1 * … (2 * (1 * 1)

もちろん、このような書き方はしない。

n	n!
0	1
1	1
2	2
3	6
4	24
5	120
6	
7	
8	
9	
10	3628800
…	…

階乗の計算



n	n!
0	1
1	1
2	2
3	6
4	24
5	120
6	
7	
8	
9	
10	3628800
…	…

上の図を数式で書くと

n! = 1　　(n = 0の場合)

　= n × (n-1)!　(n >= 1の場合)

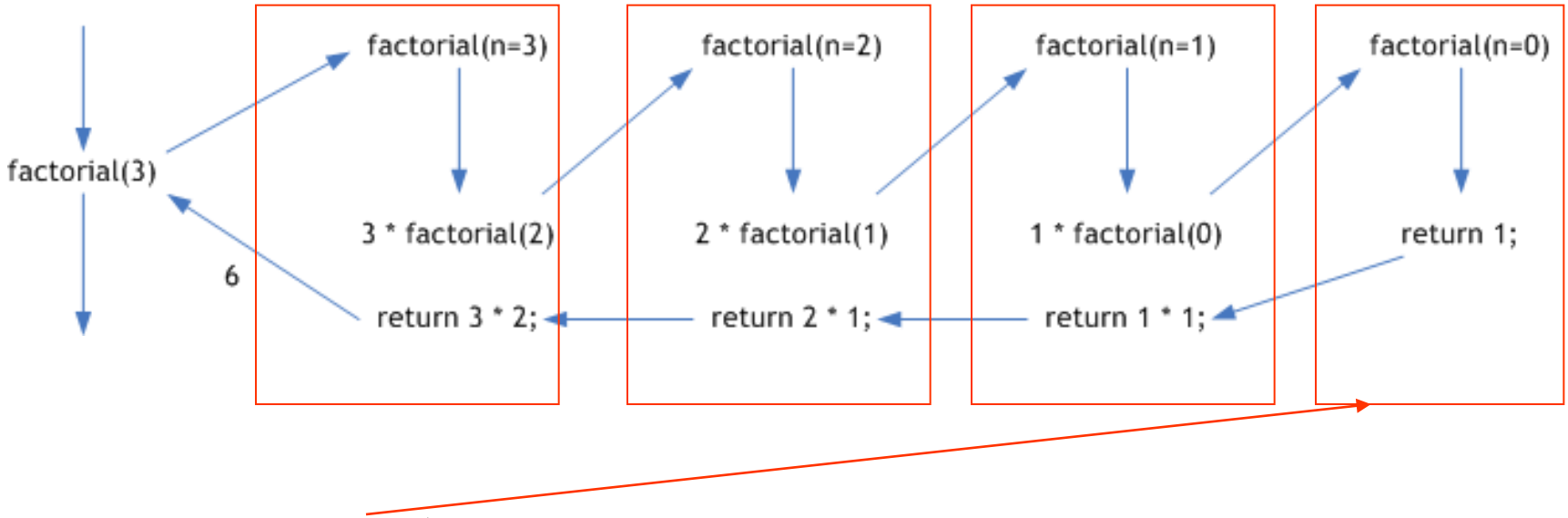
階乗の計算を再帰的(recursive)なメソッドで記述する。

考え方

```
factorial(N) = 1                          N = 0 のとき
factorial(N) = N * factorial(N-1)   それ以外のとき
```

```
int factorial(int n) {
  // factorial(N) = 1 (N = 0 のとき)
  if (n == 0) {
    return 1;
  }
  // factorial(n) = n * factorial(n-1) (それ以外のとき)
  else {
    return n * factorial(n - 1);
  }
}
```

factorial(3) として起動すると



factorial(0)が実行されているときは4つの実行中あるいは再帰呼び出しからの帰り待ちの状態のメソッド起動がある。

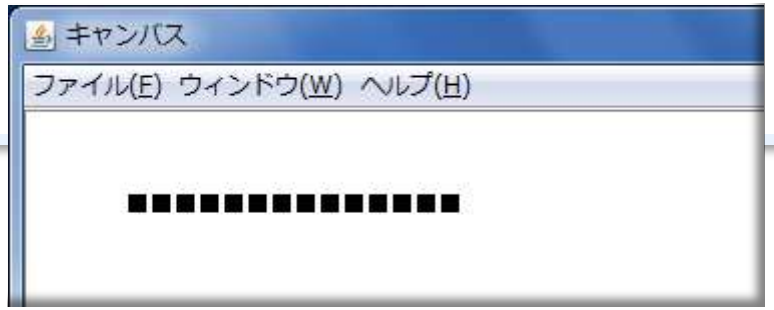
factorialメソッドの仮引数である自動変数nはfactorialの呼び出しの度に独立して自動的に作られ、消されるので、ソースプログラム上で同じ変数nであっても実行中の異なるメソッド起動では実体が異なる。

例題44

```
1  package j2.lesson02;
2
3  import javax.swing.JOptionPane;
4
5  public class Ex44Factorial {
6      public static void main(String[] args) {
7          new Ex44Factorial().start();
8      }
9
10     void start() {
11         String inputNum = JOptionPane.showInputDialog("階乗を求めます。値を入力してください");
12         int num = Integer.parseInt(inputNum);
13
14         int result = factorial(num);
15
16         JOptionPane.showMessageDialog(null, result);
17     }
18
19     int factorial(int n) {
20         if (n == 0) {
21             return 1;
22         } else {
23             return n * factorial(n - 1);
24         }
25     }
26 }
```

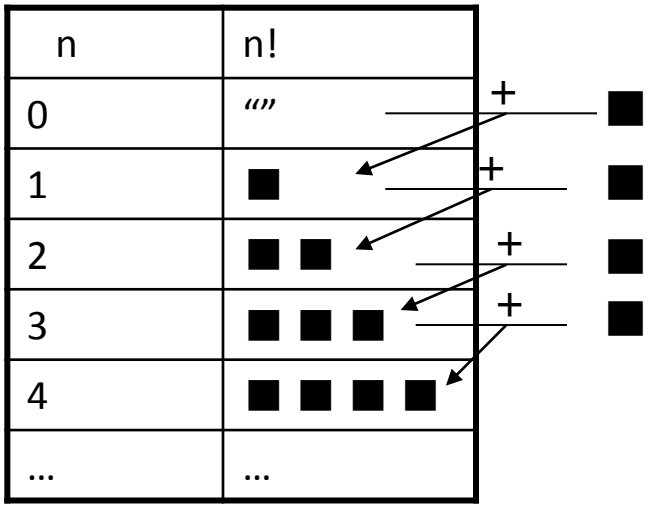
makeBar(int n)を再帰的なメソッドとして定義

```
1  package j2.lesson02;
2
3  import javax.swing.JOptionPane;
4
5  public class SampleMakeBarRec {
6      public static void main(String[] args) {
7          new SampleMakeBarRec().start();
8      }
9
10     void start() {
11         JOptionPane.showMessageDialog(null, makeBar(14));
12     }
13
14     String makeBar(int n) {
15         if (n <= 0) {
16             return "";
17         } else {
18             return "■" + makeBar(n - 1);
19         }
20     }
21 }
```



makeBar(int n)の再帰版

```
String makeBar(int n) {
    if (n <= 0) {
        return "";
    } else {
        return "■" + makeBar(n - 1);
    }
}
```



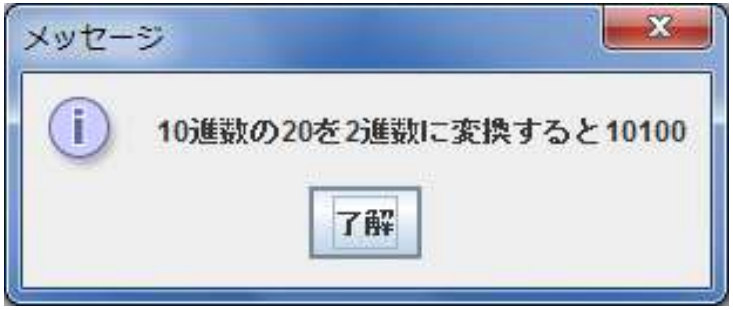
makeBar(n) = “” (n <= 0の場合)

= “■” + makeBar(n-1) (n >= 1の場合)

■例題45 (optional)

問題：10進数を2進数に変換するメソッドを作成せよ。（再帰を用いる）

返り値	メソッド名(引数)	機能
String	toBinary(int n)	引数の2進数表現を返す。



(クラス名 : Ex45ToBinary)

例題45

前のスライドのやり方で、引数と返り値の対応表を作成してみよ。
整数同士の割り算は「商」が値であることに注意。(3/2 = 1, 5/2=2, 4/2=2 ..)

```
1  package j2.lesson02;
2
3  import javax.swing.JOptionPane;
4
5  public class Ex45ToBinary {
6      public static void main(String[] args) {
7          new Ex45ToBinary().start();
8      }
9
10     void start() {
11         JOptionPane.showMessageDialog(null, "10進数の20を2進数に変換すると"
12                 + toBinary(20));
13     }
14
15     String toBinary(int n) {
16         if (n == 0)
17             return "";
18         if (n % 2 == 1)
19             return toBinary(n / 2) + "1";
20         else {
21             return toBinary(n / 2) + "0";
22         }
23     }
24 }
```

再帰的なメソッドの記述（補足）

- 本日の再帰的なメソッドの例は、普通、実行速度の面を考慮したときには、再帰を用いないメソッドで実現する。（forなどの繰り返しなどを用いる）。
  - 引数の数(n)が大きいと、大量のメソッド呼び出しが発生。一般にメソッド呼び出しは、実行コスト（メモリを消費したり、実行に時間がかかる）がかかるため、nが大きい場合、実行速度が遅くなる。
- ただし、再帰的なメソッドは、数学の帰納的な定義をほとんどそのままの形で記述できるため、短くかつ直感的であるというメリットがある。また、複雑な帰納的な関数は、再帰的なメソッドを用いないと記述することが難しい場合がある。

今日のまとめ

□メソッドの作成例(1)
　パラメータ化による機能の一般化
□メソッドの作成例(2)
　対応表による単純なデータの変換
□メソッドの作成例(3)
　演算によるデータの変換
　少し複雑な処理を用いたデータの変換
　再帰メソッド(1)

メソッドを効果的に使う方法やパターンは様々である。適切なメソッドを設計し、使いこなすことは実は、熟練したプログラマにとっても、難しいテーマであり、逆にそれが楽しみであることもある。

本日の例題と問題

- □メソッドの作成例(1)
  - Ex11, Ex12, (Q11*) , Ex13, Q12, (Q13)
  - Ex21, Ex22, Ex23, Q21, Q22
- □メソッドの作成例(2)
  - Ex31, Q31, Q32
- □メソッドの作成例(3)
  - Ex41, Ex42, Q41, Ex43, Q42
  - Ex44, Q43 , (Ex45*), (Q44*)

(Ex:例題, Q:問題, *は少し手間のかかる問題)

各自に適した順番で解けばよいが、上記の順番が自然な流れとなるよう構成されている。ただし、括弧で囲まれた問題は、後に解いても良い。（難しくはない。）

例題集

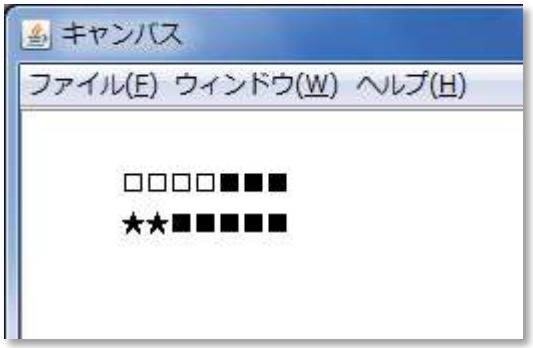
パッケージ「j2.lesson02」を作成する。

パッケージやクラスの作成,実行の仕方の説明は省略する。
作り方を忘れた場合は過去のスライドや
http://java2010.cis.k.hosei.ac.jp/01/material-01/
を参考にせよ

■例題11

問題：下記の二つのmakeBarを作成せよ。最初のメソッドは、任意の文字列を棒グラフの模様として指定できる。メソッドのオーバーロードを用いよ。

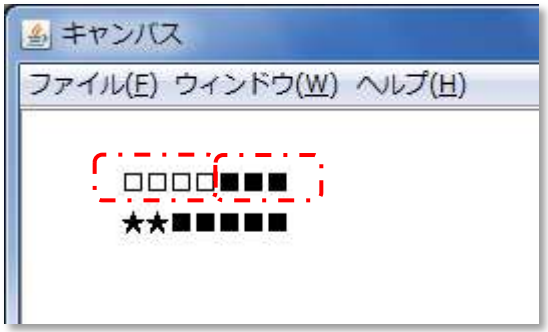
返り値	メソッド名(引数)	機能
String	makeBar(int n, String tile)	tileで指定した文字列を横にn個並べたものを返す。
String	makeBar(int n)	■を横にn個並べたものを返す。 (すなわち makeBar(n, “■”))



(クラス名：Ex11MakeBar)

例題11

```
1  package j2.lesson02;
2
3  import gpjava.Canvas;
4
5  public class Ex11MakeBar {
6      public static void main(String[] args) {
7          new Ex11MakeBar().start();
8      }
9
10     void start() {
11         Canvas.show();
12         String result = "";
13         result = makeBar(4, "□");
14         result += makeBar(3);
15         Canvas.drawString(50, 40, result);
16
17         result = "";
18         result = makeBar(2, "★");
19         result += makeBar(5);
20         Canvas.drawString(50, 60, result);
21     }
22
23     String makeBar(int n, String tile) {
24         String result = "";
25         for (int i = 0; i < n; i++) {
26             result += tile;
27         }
28         return result;
29     }
30
31     String makeBar(int n) {
32         return makeBar(n, "■");
33     }
34 }
```



両者makeBarだが
引数の部分が異なるので
別個のメソッドとして定義
できる。

前回
例題34

両者の違いに注目

```
27
28  String makeBar(int n) {
29      String result = "";
30      String tile ="■";
31
32      for(int i = 0; i < n; i++) {
33          result = result + tile;
34      }
35
    return result;
```

String tile
が、パラメータ（引数）
の部分に移った。
→■以外の好きな模様もメソッド
の外から与えることができる

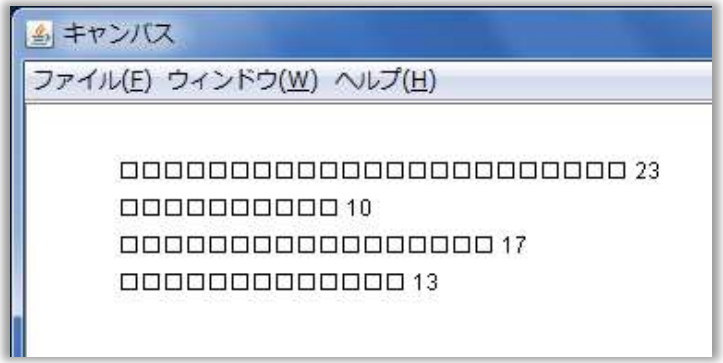
例題11

```
22
23      String makeBar(int n, String tile) {
24          String result = "";
25          for (int i = 0; i < n; i++) {
26              result += tile;
27          }
28          return result;
29      }
30
```

■例題12

問題: (例題11の応用) 例題11のmakeBar(int n, String tile)を用いることで、模様を指定して、棒グラフを表示するプログラムを作成せよ。

返り値	メソッド名(引数)	機能
String	*makeBar(int n, String tile)*	*tileで指定した文字列を横にn個並べたものを返す。(前の物と同じ)*
void	showGraph(String tile, int[] anArray)	模様がtileである棒グラフをダイアログで表示する。棒グラフの各値はanArrayに対応する。



(クラス名：Ex12ShowGraph)

例題12

```
1  package j2.lesson02;
2
3  import gpjava.Canvas;
4  import javax.swing.JOptionPane;
5
6  public class Ex12ShowGraph {
7      public static void main(String[] args) {
8          new Ex12ShowGraph().start();
9      }
10
11     void start() {
12         String message = "棒グラフの模様を指定して下さい";
13         String tile = JOptionPane.showInputDialog(message);
14         int[] a = { 23, 10, 17, 13 };
15         showGraph(tile, a);
16     }
17
18     String makeBar(int n, String tile) {
19         String result = "";
20         for (int i = 0; i < n; i++) {
21             result += tile;
22         }
23         return result;
24     }
25
26     // このプログラムでは用いない
27     String makeBar(int n) {
28         return makeBar(n, "■");
29     }
30
31     void showGraph(String tile, int[] anArray) {
32         Canvas.show();
33         String result = "";
34         for (int i = 0; i < anArray.length; i++) {
35             result = makeBar(anArray[i], tile) + " " + anArray[i];
36             Canvas.drawString(50, 40 + i * 20, result);
37         }
38     }
39 }
```

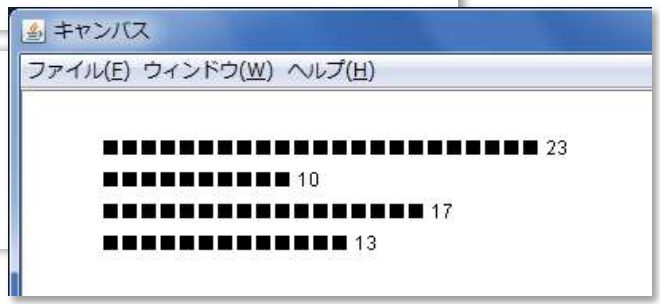
（前の例題と同じ）

模様を自由に
指定できる。

前回
例題44

```
26    void showGraph(int[] anArray) {
27        Canvas.show();
28        String result;
29
30        for(int i = 0; i < anArray.length; i++) {
31            result = makeBar(anArray[i]) + " " + anArray[i];
32            Canvas.drawString(50, 40 + i * 20, result);
33        }
34    }
```

```
10    void start() {
11        int[] a = {23, 10, 17, 13};
12        showGraph(a);
13    }
```

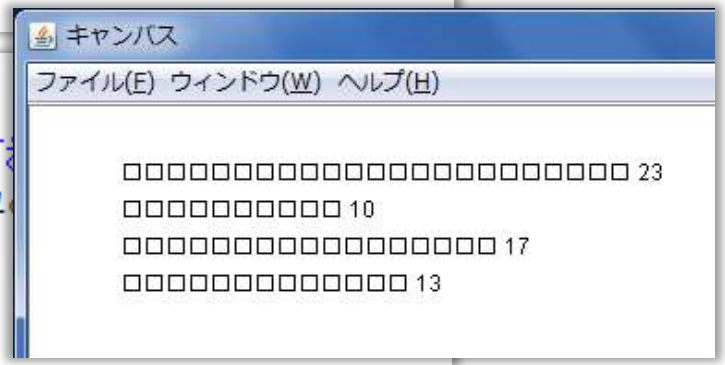


両者の違いに注目

例題12

```
31    void showGraph(String tile, int[] anArray) {
32        Canvas.show();
33        String result = "";
34        for (int i = 0; i < anArray.length; i++) {
35            result = makeBar(anArray[i], tile) + " " + anArray[i];
36            Canvas.drawString(50, 40 + i * 20, result);
37        }
```

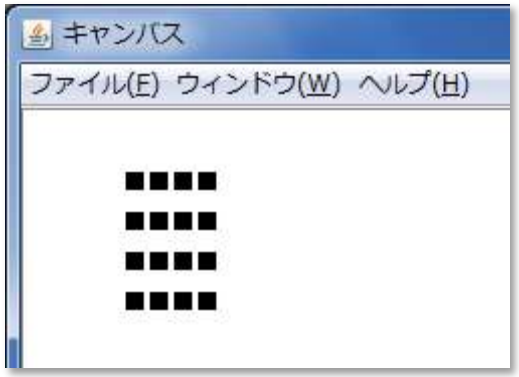
```
10
11    void start() {
12        String message = "棒グラフの模様を指定して下
13        String tile = JOptionPane.showInputDial
14        int[] a = { 23, 10, 17, 13 };
15        showGraph(tile, a);
16    }
```



■例題13 (問題12への準備)

問題：縦と横に同じ数だけ、つまりn×nだけ「■」を並べて正方形の形となる文字列を画面に表示するメソッドshowSquareを作成せよ。

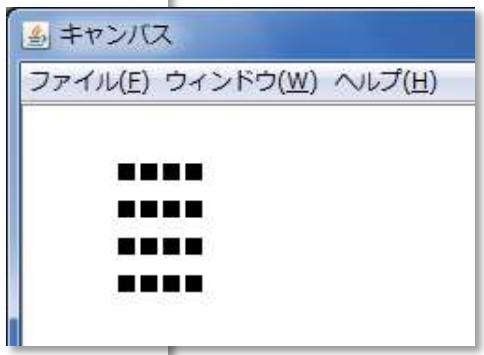
返り値	メソッド名(引数)	機能
void	showSquare(int n)	■を縦と横にn個並べた文字列を返す。



(クラス名： Ex13ShowSquare)

例題13

```
1  package j2.lesson02;
2
3  import gpjava.Canvas;
4
5  public class Ex13ShowSquare {
6      public static void main(String[] args) {
7          new Ex13ShowSquare().start();
8      }
9
10     void start() {
11         Canvas.show();
12         showSquare(2);
13         Canvas.waitForCountdown(1000);
14         showSquare(4);
15     }
16
17     void showSquare(int n) {
18         Canvas.clear();
19         String result = "";
20         for (int i = 0; i < n; i++) {
21             result = "";
22             for (int j = 0; j < n; j++) {
23                 result += "■";
24             }
25             Canvas.drawString(50, 40 + i * 20, result);
26         }
27     }
28 }
```



■例題21

問題：色付きの文字をキャンパスに表示せよ。

返り値	メソッド名(引数)	機能
void	setMagenta()	表示色をマゼンタにする
void	setYellow()	表示色を黄色にする

(クラス名：Ex21ColorMessage)

例題21

```
1  package j2.lesson02;
2
3  import gpjava.Canvas;
4
5  public class Ex21ColorMessage {
6      public static void main(String[] args) {
7          new Ex21ColorMessage().start();
8      }
9
10     void start() {
11         Canvas.show();
12
13         String contents = "★★★★";
14
15         setMagenta();
16         Canvas.drawString(50, 40, contents);
17         setYellow();
18         Canvas.drawString(50, 60, contents);
19     }
20
21     void setMagenta() {
22         Canvas.setColor(255, 0, 255);
23     }
24
25     void setYellow() {
26         Canvas.setColor(255, 255, 0);
27     }
28 }
```

■例題22

問題：色付きのコンテンツを(x,y)座標から表示するメソッドcolorContentsを作成せよ。

返り値	メソッド名(引数)	機能
void	colorContents(String contents, int x, int y, int r, int g, int b)	文字列 contents を (x, y)座標を右上として、指定した RGB色(r, g, b)で表示する。

(クラス名：Ex22ColorMessage)

例題22


```
1  package j2.lesson02;
2
3  import gpjava.Canvas;
4
5  public class Ex22ColorMessage {
6      public static void main(String[] args) {
7          new Ex22ColorMessage().start();
8      }
9
10     void start() {
11         Canvas.show();
12         colorContents("■■■■ ★★★★", 50, 40, 255, 0, 255);
13         colorContents("■■■■ ★★★★", 50, 60, 255, 255, 0);
14
15         colorContents("赤です■", 50, 80, 255, 0, 0);
16         colorContents("緑です■", 50, 100, 0, 255, 0);
17         colorContents("青です■", 50, 120, 0, 0, 255);
18     }
19
20     void colorContents(String contents, int x, int y, int r, int g, int b) {
21         Canvas.setColor(r, g, b);
22         Canvas.drawString(x, y, contents);
23     }
24 }
```

■例題23

問題：例題11のmakeBarを用いて、棒の色も指定できるように拡張したshowColorBarを作成し、挙動を確かめよ。

（前の例題と同じ）

返り値	メソッド名(引数)	機能
void	*colorContents(* *String contents,* *int x, int y,* *int r, int g, int b)*	*文字列 contents を (x, y)座標を右上として、指定した RGB 色(r, g, b) で表示する。*
String	*makeBar(int n,* *String tile)*	*tile で指定した文字列を横にn列並べたものを返す。*
void	showColorBar(int n, String tile, int x, int y, int r, int g, int b)	tileで指定した文字列を(x, y)座標を起点に横にn列て表示する。さらに引数r, g, bでRGB色を指定できるようにする。

(クラス名：Ex23MakeColorBar)

実行例

```
10    void start() {
11        Canvas.show();
12
13        showColorBar(4, "■", 50, 40, 255, 0, 255);
14        showColorBar(6, "★", 50, 60, 255, 255, 0);
15    }
```

例題23

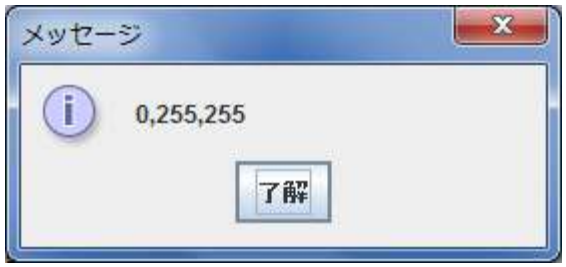
```
1  package j2.lesson02;
2
3  import gpjava.Canvas;
4
5  public class Ex23ShowColorBar {
6      public static void main(String[] args) {
7          new Ex23ShowColorBar().start();
8      }
9
10     void start() {
11         Canvas.show();
12
13         showColorBar(4, "■", 50, 40, 255, 0, 255);
14         showColorBar(6, "★", 50, 60, 255, 255, 0);
15     }
16
17     void colorContents(String contents, int x, int y, int r, int g, int b) {
18         Canvas.setColor(r, g, b);
19         Canvas.drawString(x, y, contents);
20     }
21
22     void showColorBar(int n, String tile, int x, int y, int r, int g, int b) {
23         colorContents(makeBar(n, tile), x, y, r, g, b);
24     }
25
26     String makeBar(int n, String tile) {
27         String result = "";
28         for (int i = 0; i < n; i++) {
29             result += tile;
30         }
31         return result;
32     }
33 }
```

■例題31

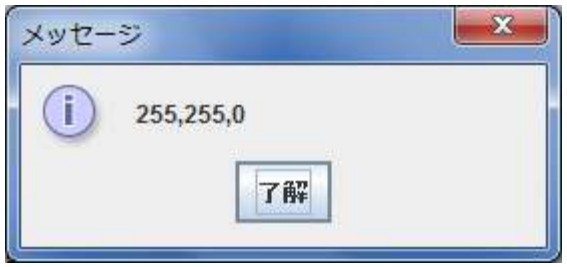
問題:色名を対応するRGB値の数字文字列に変換するメソッドを作成せよ。

返り値	メソッド名(引数)	機能
String	getColorByName (String colorName)	基本の8色の色名に対応するRGBの数字文字列を返す。 例:”black” -> “0,0,0”

“cyan”に対応する色番号



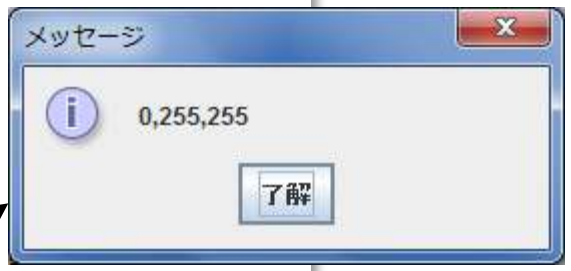
“yellow”に対応する色番号

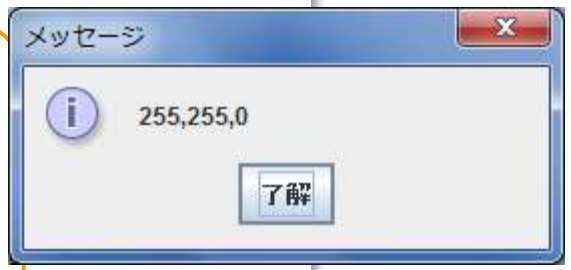


(クラス名： Ex31GetColorByName)

例題31

```
1  package j2.lesson02;
2
3  import javax.swing.JOptionPane;
4
5  public class Ex31GetColorByName {
6      public static void main(String[] args) {
7          new Ex31GetColorByName().start();
8      }
9
10     void start() {
11         JOptionPane.showMessageDialog(null, getColorByName("cyan"));
12         JOptionPane.showMessageDialog(null, getColorByName("yellow"));
13     }
14
15     String getColorByName(String colorName) {
16         String black   = "0,0,0";
17         String blue    = "0,0,255";
18         String green   = "0,255,0";
19         String red     = "255,0,0";
20         String cyan    = "0,255,255";
21         String magenta = "255,0,255";
22         String yellow  = "255,255,0";
23         String white   = "255,255,255";
24
25         if ("black".equals(colorName)) {
26             return black;
27         } else if ("blue".equals(colorName)) {
28             return blue;
29         } else if ("red".equals(colorName)) {
30             return red;
31         } else if ("green".equals(colorName)) {
32             return green;
33         } else if ("magenta".equals(colorName)) {
34             return magenta;
35         } else if ("yellow".equals(colorName)) {
36             return yellow;
37         } else if ("cyan".equals(colorName)) {
38             return cyan;
39         } else if ("white".equals(colorName)) {
40             return white;
41         } else { // 登録されていない色の場合は白を返す。
42             return white;
43         }
44     }
45 }
```





例題31補足：文字列同士の比較

```
if ("black".equals(colorName)) {
    return black;
} else if ("blue".equals(colorName)) {
    return blue;
} else if ("red".equals(colorName)) {
```

この場合、

```
if (colorName.equals("black") {
   return black;
} …
```

という書き方も可能。

文字列s1と文字列s2の内容が
等しいかどうかを確かめる時には、
　s1.equals(s2)
を用いる。等しいときにはtrue、
そうでないときは、falseを返す。
s1==s2では、うまくいかない。
(理由は、他の機会に)

```
15    String getColorByName(String colorName) {
16        String black   = "0,0,0";
17        String blue    = "0,0,255";
18        String green   = "0,255,0";
19        String red     = "255,0,0";
20        String cyan    = "0,255,255";
21        String magenta = "255,0,255";
22        String yellow  = "255,255,0";
23        String white   = "255,255,255";
24
25        if ("black".equals(colorName)) {
26            return black;
27        } else if ("blue".equals(colorName)) {
28            return blue;
29        } else if ("red".equals(colorName)) {
30            return red;
31        } else if ("green".equals(colorName)) {
32            return green;
33        } else if ("magenta".equals(colorName)) {
34            return magenta;
35        } else if ("yellow".equals(colorName)) {
36            return yellow;
37        } else if ("cyan".equals(colorName)) {
38            return cyan;
39        } else if ("white".equals(colorName)) {
40            return white;
41        } else { // 登録されていない色の場合は白を返す。
42            return white;
43        }
```

色名を番号に変換
(色番号を色の名前から得る)

下記のような変換表から、色名に対応する色番号を探している。

色名	番号
black	0,0,0
blue	0,0,255
green	0,255,0
red	255,0,0
cyan	0,255,255
magenta	255,0,255
yellow	255,255,0
white	255,255,255

■例題41

問題：絶対値を求めるメソッドおよび2乗を求めるメソッドを作成せよ。

返り値	メソッド名(引数)	機能
int	absolute(int n)	引数の絶対値を返す
int	square(int n)	引数の2乗を返す

(クラス名： Ex41AbsoluteSquare)

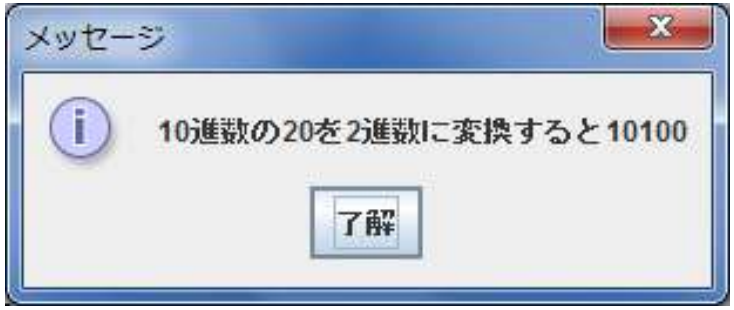
例題41

```
1  package j2.lesson02;
2
3  import javax.swing.JOptionPane;
4
5  public class Ex41AbsoluteSquare {
6      public static void main(String[] args) {
7          new Ex41AbsoluteSquare().start();
8      }
9
10     void start() {
11         JOptionPane.showMessageDialog(null, "-10の絶対値は"+ absolute(-10));
12         JOptionPane.showMessageDialog(null, "-10の2乗は"+ square(-10));
13     }
14
15     int absolute(int n) {
16         if (n < 0) {
17             return -n;
18         } else {
19             return n;
20         }
21     }
22
23     int square(int n) {
24         return n * n;
25     }
26
27 }
```

■例題42

問題：１０進数を２進表現に変換するメソッドを作成せよ

返り値	メソッド名(引数)	機能
String	toBinary(int n)	引数の2進数表現を返す。



(クラス名：Ex42ToBinary)

例題42

```java
package j2.lesson02;

import javax.swing.JOptionPane;

public class Ex42ToBinary {
    public static void main(String[] args) {
        new Ex42ToBinary().start();
    }

    void start() {
          JOptionPane.showMessageDialog(null, "10進数の20を2進数に変換すると"+ toBinary(20));
    }

    String toBinary(int n){
        String result = "";
        do{
            if(n%2 == 1)
                result = "1" + result;
            else
                result = "0" + result;
            n = n / 2;
        }while(n != 0);

        return result;
    }
}
```

■例題43 (問題42への準備)

問題: RGBについて以下のようなメソッドを作成せよ。

返り値	メソッド名(引数)	機能
int	getIntFromRGB(int r, int g, int b)	RGBの各値(10進数)から次のような整数値を求め、返す。 例 : Magenta (red=255, green=0, blue=255) = 255 * 65536 + 0 * 256 + 255 * 1

(クラス名 : Ex43GetIntFromRGB)

例題43

```
1  package j2.lesson02;
2
3  import javax.swing.JOptionPane;
4
5  public class Ex43GetIntFromRGB {
6      public static void main(String[] args) {
7          new Ex43GetIntFromRGB().start();
8      }
9
10     void start() {
11         JOptionPane.showMessageDialog(null, "Magenta(255,0,255) : "
12                 + getIntFromRGB(255, 0, 255));
13     }
14
15     int getIntFromRGB(int r, int g, int b) {
16         return r * 65536 + g * 256 + b * 1;
17     }
18
19 }
```

■例題44

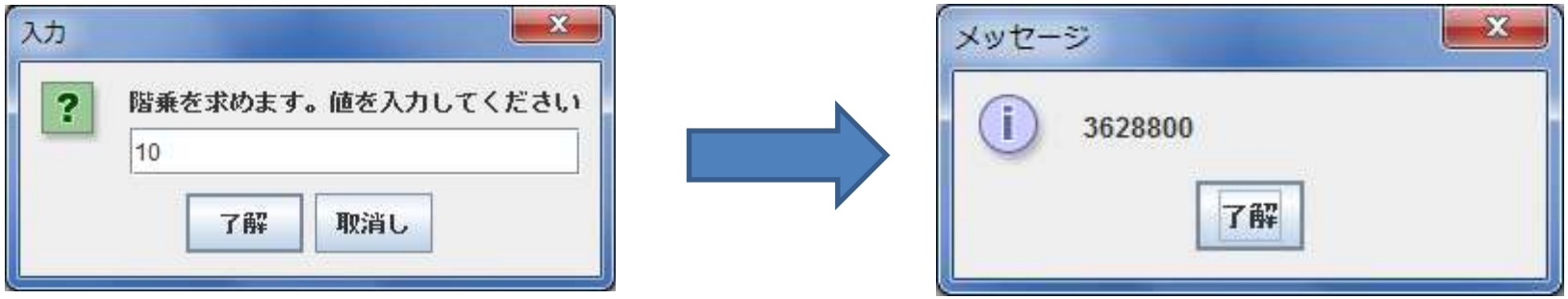
問題:階乗を求めるメソッドを作成せよ。nの階乗n!を求める階乗の式は、

n! = 1　　(n = 0の場合)

　= n × (n-1)!　(n >= 1の場合)

と表せる。

返り値	メソッド名(引数)	機能
int	factorial(int n)	引数の階乗の値を返す。



(クラス名 : Ex44Factorial)

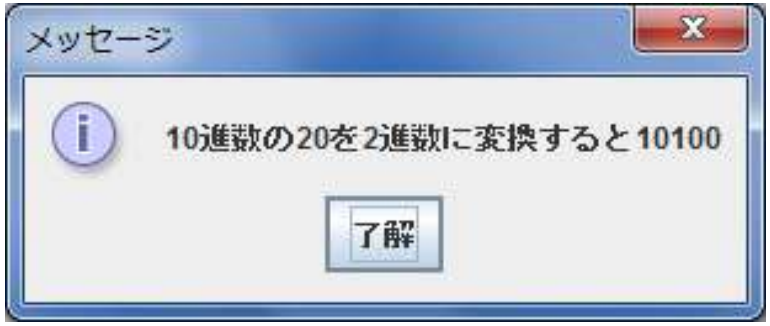
例題44

```
1  package j2.lesson02;
2
3  import javax.swing.JOptionPane;
4
5  public class Ex44Factorial {
6      public static void main(String[] args) {
7          new Ex44Factorial().start();
8      }
9
10     void start() {
11         String inputNum = JOptionPane.showInputDialog("階乗を求めます。値を入力してください");
12         int num = Integer.parseInt(inputNum);
13
14         int result = factorial(num);
15
16         JOptionPane.showMessageDialog(null, result);
17     }
18
19     int factorial(int n) {
20         if (n == 0) {
21             return 1;
22         } else {
23             return n * factorial(n - 1);
24         }
25     }
26 }
```

■例題45

問題：10進数を2進数に変換するメソッドを作成せよ。（再帰を用いる）

返り値	メソッド名(引数)	機能
String	toBinary(int n)	引数の2進数表現を返す。



(クラス名：Ex45ToBinary)

例題45

```
1  package j2.lesson02;
2
3  import javax.swing.JOptionPane;
4
5  public class Ex45ToBinary {
6      public static void main(String[] args) {
7          new Ex45ToBinary().start();
8      }
9
10     void start() {
11         JOptionPane.showMessageDialog(null, "10進数の20を2進数に変換すると"
12                 + toBinary(20));
13     }
14
15     String toBinary(int n) {
16         if (n == 0)
17             return "";
18         if (n % 2 == 1)
19             return toBinary(n / 2) + "1";
20         else {
21             return toBinary(n / 2) + "0";
22         }
23     }
24 }
```