プログラミング入門2

第1回　プログラミングの復習

ログオン(システムに入る)

- 授業開始前にログオンする
  - Webブラウザを起動して以下のURLを開いてみよう
  - http://java2011.cis.k.hosei.ac.jp/
- 以下の情報の入力が必要なので確認
  - ユーザID
  - パスワード
    - 他人に教えるのはユーザの義務違反
- 周りを見回して困っている人がいたら助けてやってください。

今日の講義のテーマ

□導入

講義の進め方、評価方法

□復習

プログラミング言語の基本機能の復習

標準的な講義の進め方

□説明

新たな言語機能の説明

プログラム構築のためのアイディアの説明

□例題にチャレンジ

学んだ事項を実際に試してみる。

□問題（課題）にチャレンジ

学んだ事項を利用、応用して、自力でプログラムを完成させる。

評価

□中間・期末試験（全クラス共通）の達成度

□詳細は、各教員の方針に従う。

参考図書

- Java言語プログラミングレッスン（下）
  - 結城 浩
  - ソフトバンククリエイティブ
- やさしいJava (第4版)
  - 高橋 麻奈
  - ソフトバンクパブリッシング
- Head First Java-頭とからだで覚えるJavaの基本 (第２版)
  - キャシー シエラ, バート ベイツ
  - オライリージャパン

自分に合った参考書を探してみてください

Teaching Assistant について

- 授業には教員の他に学生によるTeaching Assistant (TA) が参加
- 主に以下を担当:
  - 演習の補助
  - 質問の受け付け
  - 課題のチェック
- 学生アシスタント（学部生）
  - GBCにも勤務しています

授業で困ったことがあれば
遠慮なくTAに声をかけて下さい。
GBCも利用してください。

今日の講義のテーマ

□導入

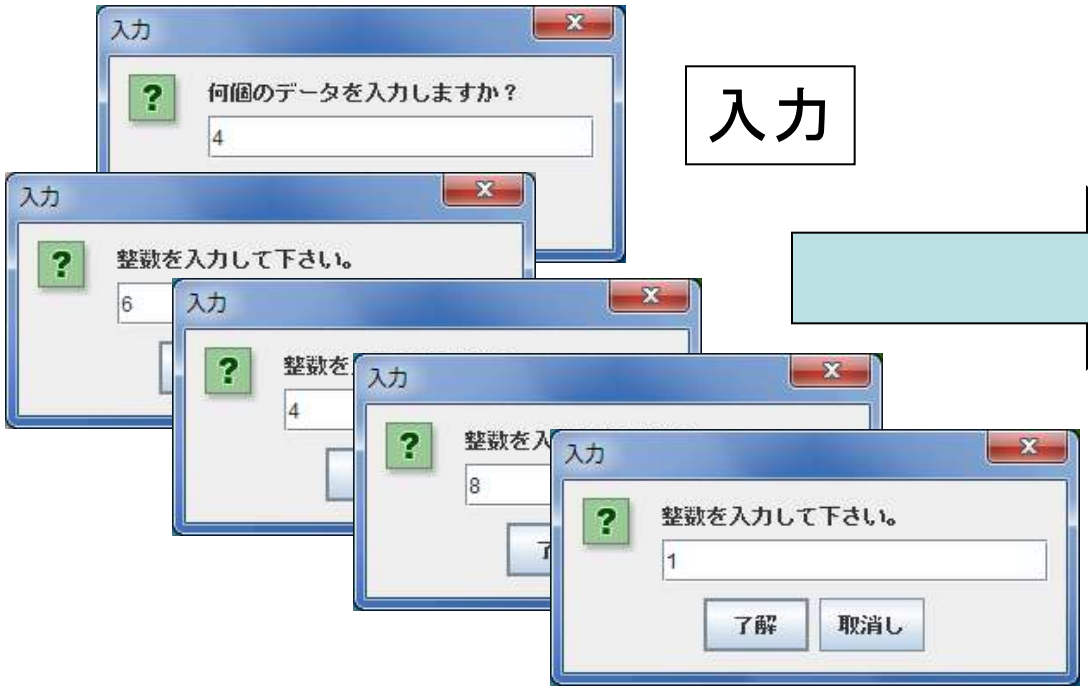
講義の進め方、評価方法

□復習

プログラミング言語の基本機能の復習

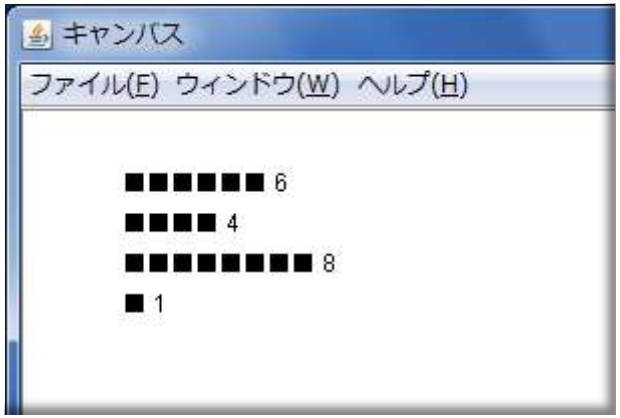
本日の主な目的(題材)

- n個の数値をユーザから得て、棒グラフとして表示せよ。(問題21，問題42)



入力

出力



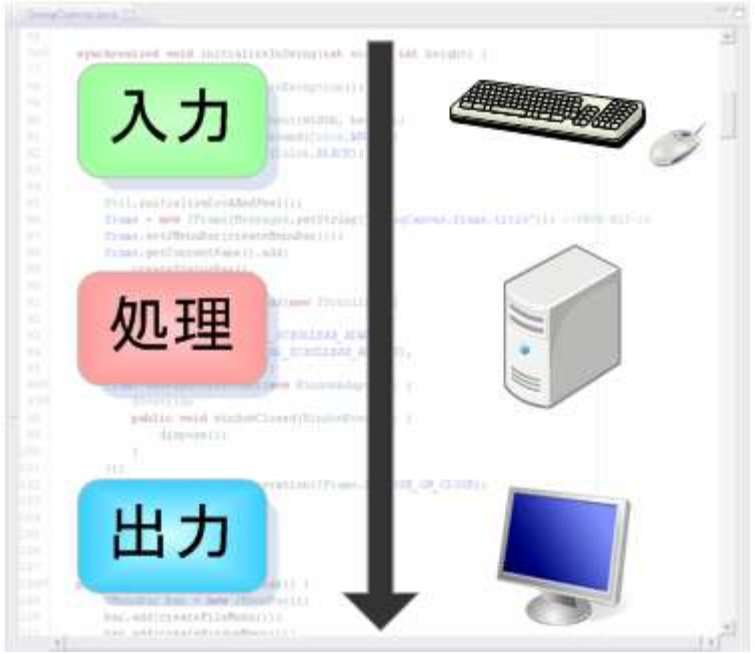
今回は、基本的に自主学習。

プログラムの構造

- このプログラムは次のような内容
  - 入力ダイアログを表示して名前を入力
  - **入力した内容の先頭に「名前は」を追加する処理**
  - 完成した文字列をメッセージダイアログで出力

入力、処理、出力
という流れ

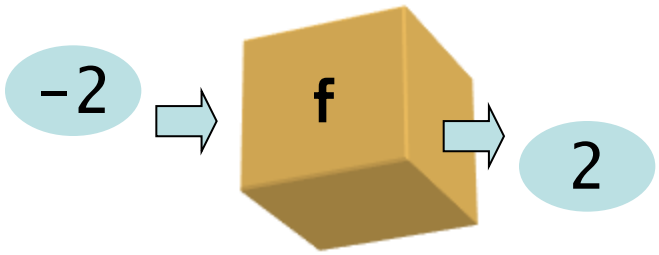
初回の授業なので
簡単な処理のみ



「入力→処理→出力」の流れの例

- （例）ある数値の絶対値を求めるプログラム
  - 知りたい数値をユーザに**入力**してもらう。
  - 入力値に応じて、答えを求める**処理を**する。
  - 結果を**出力**する。

$$f(x) = |x|$$



x ←知りたい絶対値は？

入力

xの値が、
…
-2 ならば答えは2
-1ならば答えは1
0 ならば答えは0
1 ならば答えは1
2 ならば答えは2
…

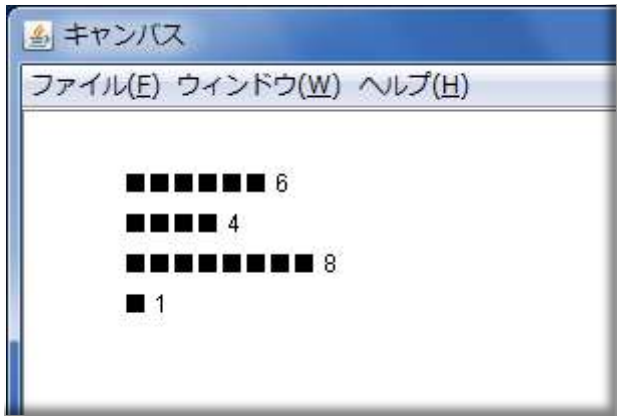
処理f

答えを表示する

出力

「入力→処理→出力」

入力　処理　出力

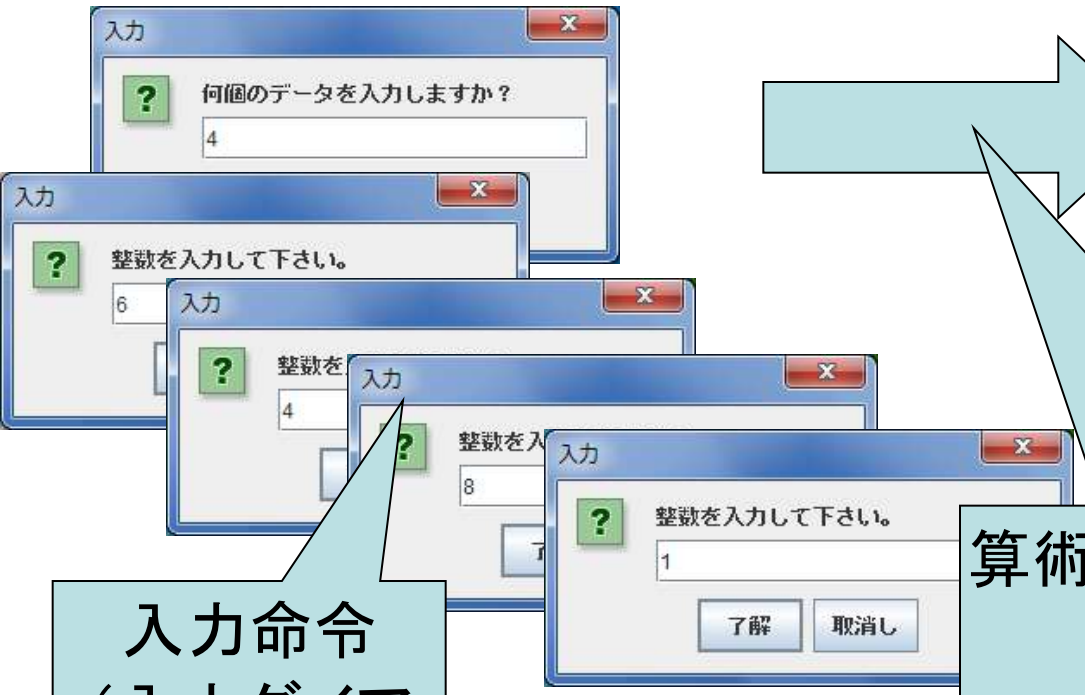


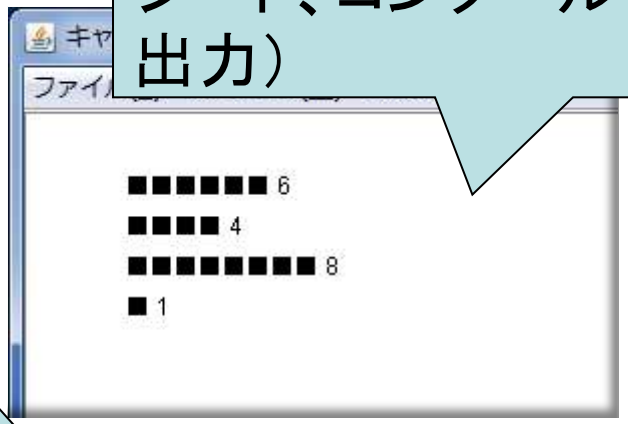
この場合の処理とは、
「入力として得たデータ（数値列）を棒グラフの形となるよう、文字列を構成する。」という**処理**

必要となる言語の機構（一部）

入力　処理　出力

出力命令
（ダイアログ、キャンバス、スプレッドシート、コンソール出力）





入力命令
（入力ダイアログなど）

算術式、論理式、文字列処理、
変数機構、型、
条件分岐、繰り返し、
メソッド、配列、etc

復習事項

- 例題11 ～
  - 入力、出力命令、変数、繰り返し(for文)
- 例題21 ～
  - Spreadsheet を使う(入門1の分科教材の利用)
  - 繰り返し(for文) , if文, 論理演算
- 例題31 ～
  - Canvas を使う(入門1の分科教材の利用)
  - メソッド（関数、手続き呼び出し）機構
- 例題41 ～
  - １次元配列、およびまとめ

例題集

本日の例題と問題

- Ex11, Ex12, Q11
- Ex21, Ex22 , Q21
- Ex31, Ex32, Q31 , Ex33, Ex34, Q32
- Ex41, Ex42, Q41, Ex43, Ex44, Q42

(Ex:例題, Q: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。

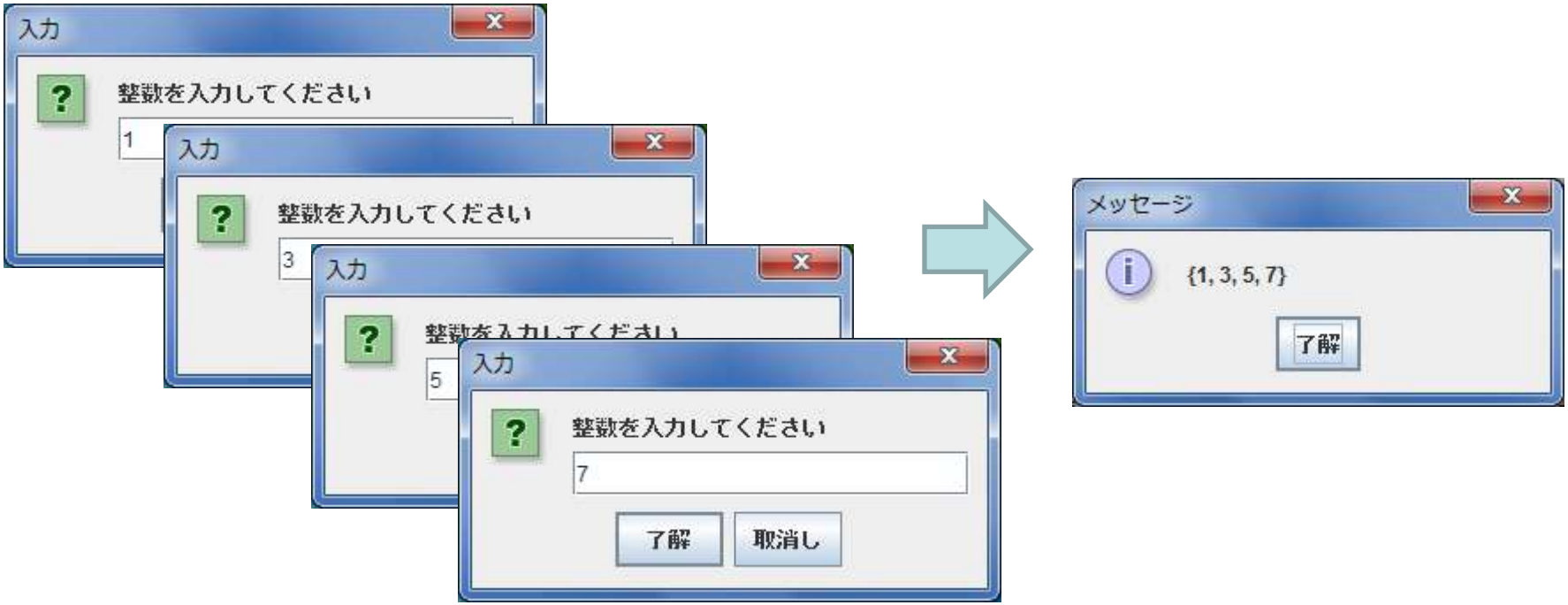
パッケージ「j2.lesson01」を作成する。

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

※プロジェクトは「java2」を使う
存在しない場合は上記URLに作り方が載っている

■例題11

問題：入力ダイアログを用いてユーザから4つの数値を得て、これらの数値を実行例を参考に ダイアログに表示せよ。



（クラス名：Ex11InputFour）

例題11

```
1  package j2.lesson01;
2
3  import javax.swing.JOptionPane;
4
5  public class Ex11InputFour {
6      public static void main(String[] args) {
7          new Ex11InputFour().start();
8      }
9
10     void start() {
11         String inputA0 = JOptionPane.showInputDialog("整数を入力してください");
12         int a0 = Integer.parseInt(inputA0);
13         String inputA1 = JOptionPane.showInputDialog("整数を入力してください");
14         int a1 = Integer.parseInt(inputA1);
15         String inputA2 = JOptionPane.showInputDialog("整数を入力してください");
16         int a2 = Integer.parseInt(inputA2);
17         String inputA3 = JOptionPane.showInputDialog("整数を入力してください");
18         int a3 = Integer.parseInt(inputA3);
19
20         String result = "{" + a0 + ", " + a1 + ", " + a2 + ", " + a3 + "}";
21         JOptionPane.showMessageDialog(null, result);
22     }
23 }
```

変数

- 変数は記憶できる値の種類ごとに宣言
  - 文字列: String
  - 整数: int
  - 実数: double
- 後から変数名を指定して値を利用できる

変数の宣言

```
String <変数名> = <文字列>;
int <変数名> = <整数>;
double <変数名> = <実数>;
```

数値への変換

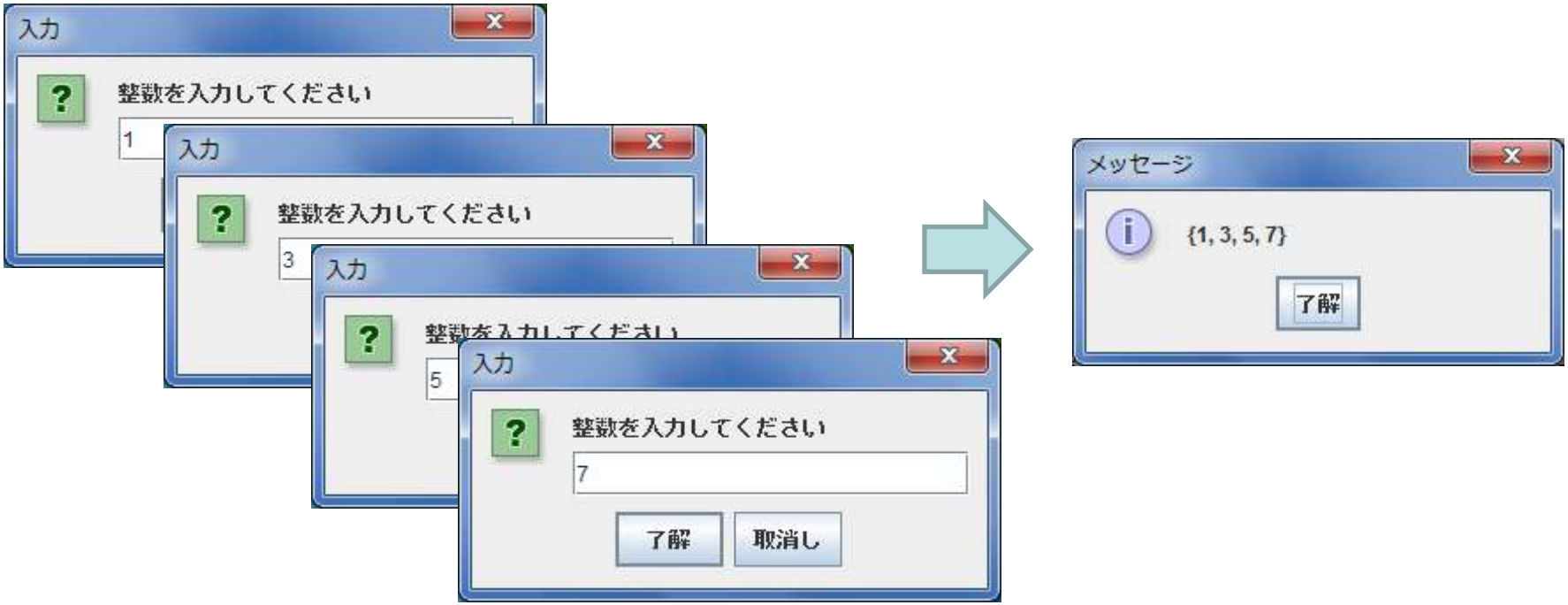
- 文字列と数値は別物
  - 数値に変換する命令をプログラムに書く
  - 下記はリテラルの代わりに使用できる

数値への変換

```
Integer.parseInt(<文字列>)
Double.parseDouble(<文字列>)
```

■例題12

問題：入力ダイアログを用いてユーザから4つの数値を得て、これらの数値を実行例を参考に ダイアログに表示せよ。for文による繰り返しを利用せよ。



（クラス名：Ex12InputFour2）

例題12

```
1  package j2.lesson01;
2
3  import javax.swing.JOptionPane;
4
5  public class Ex12InputFour {
6      public static void main(String[] args) {
7          new Ex12InputFour().start();
8      }
9
10     void start() {
11         String message = "整数を入力してください";
12         String inputIntNum;
13         int intNum;
14         String result = "";
15         for(int i = 0; i < 4; i++) {
16             inputIntNum = JOptionPane.showInputDialog(message);
17             intNum = Integer.parseInt(inputIntNum);
18             result = result + intNum + " ";
19         }
20
21         JOptionPane.showMessageDialog(null, result);
22     }
23 }
```

繰り返しの命令

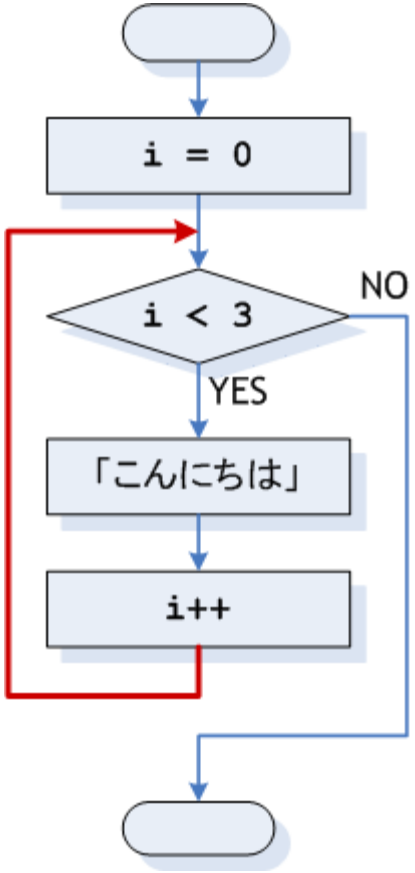
- forから始まる命令を書いている
  - 今回はダイアログが３回表示される

```
for (int i = 0; i < 3; i++) {
   JOptionPane.showMessageDialog(null,
        "こんにちは！");
}
```

これを「for文」と呼ぶ

for文の構造

- 初期化: int **i** = 0
  - 最初に１度だけ変数**i**を宣言
- 条件: **i** < 3
  - **i** < 3の場合は繰り返し
  - それ以外では繰り返しを終了
- 本体: {…}
  - 繰り返しごとに処理されるブロック
- 更新: **i**++
  - 繰り返しごとに**i**に１を加算

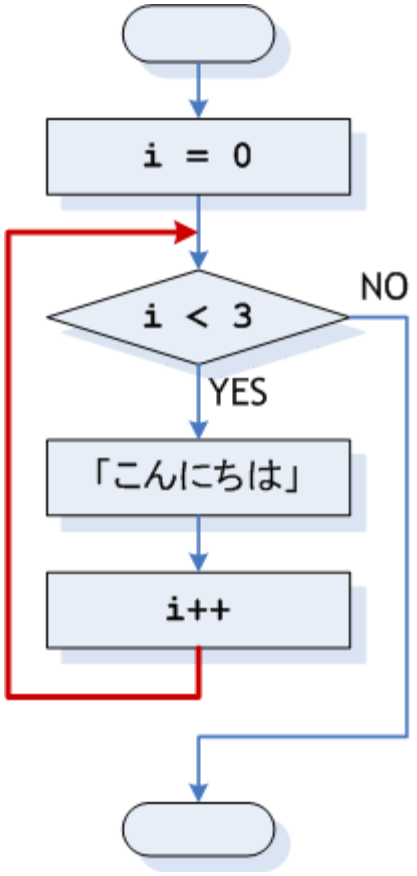



```
for (int i = 0; i < 3; i++) { ... }
```

for文の流れ

- 最初にiに0を記憶させる
- ブロックを処理するごとにiを1増やす
  - i = **0**, 1, 2, **3, 4, …**
- i < 3になったら繰り返しをやめる
  - i = **0**, 1, 2

```
for (int i = 0; i < 3; i++) { ... }
```



今回はブロックを**3**回処理
(i = 0, 1, 2 のとき)

■例題21

問題：実行例のようにn個の数値をユーザから得て、スプレッドシートに表示せよ。

（クラス名： Ex21InputN）

教材フレームワークの導入

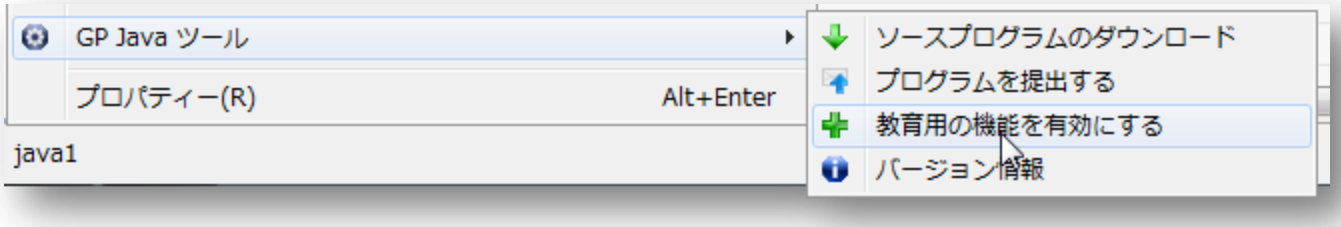
- 新しいプロジェクト java2 に対して、教材フレームワークを導入する
  - プログラミング入門1の資料を参考に、教材フレームワークを導入する。
    - http://java2010.cis.k.hosei.ac.jp/appendix/install-framework/
  - 既に、Eclipse に教材フレームワークを導入している環境では、上記のページの最後にある「教材フレームワークの確認」の手順だけをプロジェクト「java2」に対して行う。

教材フレームワークの導入 (10)

- ここまですべて終わったらEclipseを起動
  - プロジェクトを右クリックしたメニューに「GP Javaツール」という項目が増える

分化教材の有効化

- 「GP Javaツール」メニューから「教育用の機能を有効にする」を選択



ここまでで分化教材を利用するための全ての手順は完了

例題21(例題12と比較してみよ)

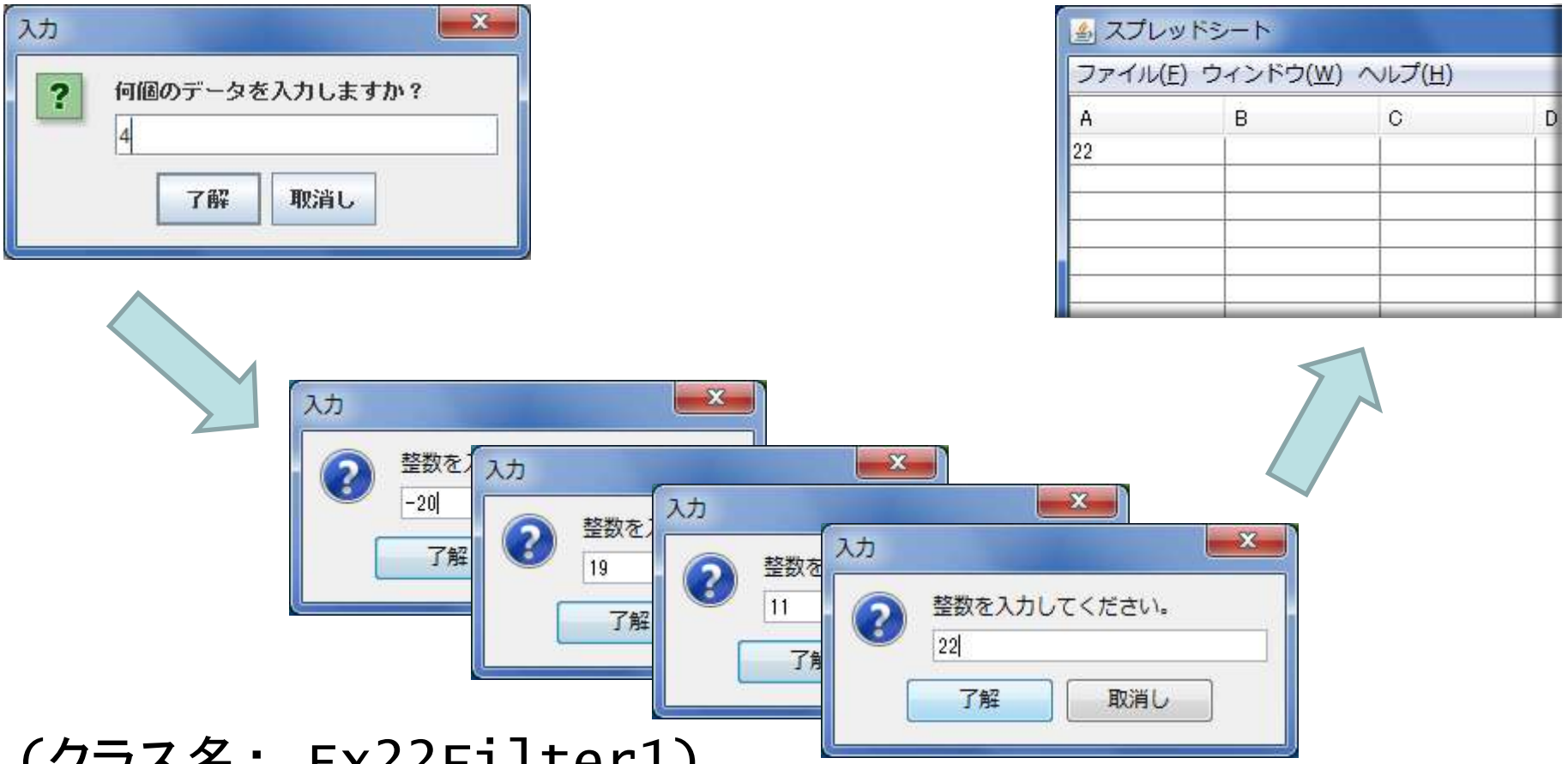
```
1  package j2.lesson01;
2
3  import gpjava.Spreadsheet;
4  import javax.swing.JOptionPane;
5
6  public class Ex21InputN {
7      public static void main(String[] args) {
8          new Ex21InputN().start();
9      }
10
11     void start() {
12         String inputNumData = JOptionPane.showInputDialog("何個のデータを入力しますか？");
13         int numData = Integer.parseInt(inputNumData);
14
15         String message = "整数を入力してください。";
16         String inputIntNum;
17         int intNum;
18
19         Spreadsheet.show();
20         for(int i = 0; i < numData; i++) {
21             inputIntNum = JOptionPane.showInputDialog(message);
22             intNum = Integer.parseInt(inputIntNum);
23             Spreadsheet.setInt(i, 0, intNum);
24         }
25     }
26 }
```

Spreadsheetの使い方の概要

- package の次の行に、import 文を記入
  - import gpjava.Spreadsheet;
- 実行の最初に、Spreadsheet.show() を実行
  - 横に5列以上の表示を行う時には、show() の引数として、show(行数, 列数)を指定する。
- String, int, double の変数を表示させる
  - Spreadsheet.setString(行, 列, String変数);
  - Spreadsheet.setInt(行, 列, int変数);
  - Spreadsheet.setDouble(行, 列, double変数);
- 画面表示の消去(初期化)
  - Spreadsheet.clear();

■例題22

問題：実行例のようにn個の数値をユーザから得て、20以上の30以下となるもののみを表示せよ。



(クラス名： Ex22Filter1)

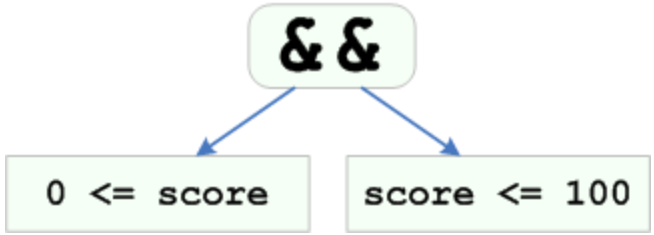
例題22（例題21との違いに注意）

```
1  package j2.lesson01;
2
3  import gpjava.Spreadsheet;
4  import javax.swing.JOptionPane;
5
6  public class Ex22Filter1 {
7
8      public static void main(String[] args) {
9          new Ex22Filter1().start();
10     }
11
12     void start() {
13         String inputNumData = JOptionPane.showInputDialog("何個のデータを入力しますか？");
14         int numData = Integer.parseInt(inputNumData);
15
16         String message = "整数を入力してください。";
17         String inputIntNum;
18         int intNum;
19         int index = 0;
20
21         Spreadsheet.show();
22         for(int i = 0; i < numData; i++) {
23             inputIntNum = JOptionPane.showInputDialog(message);
24             intNum = Integer.parseInt(inputIntNum);
25             if(intNum >=20 && intNum <=30) {
26                 Spreadsheet.setInt(index, 0, intNum);
27                 index++;
28             }
29         }
30     }
31 }
32
```

条件を満たす入力（intNum）のときだけ結果を表示する。

AかつB

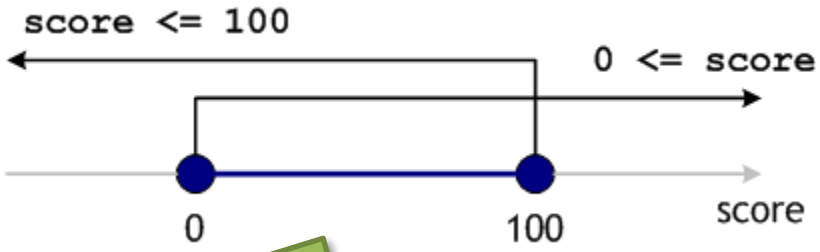
- &&は「AかつB」という接続詞のようなもの
  - 「scoreが0以上」かつ「scoreが100以下」




```
if (0 <= score && score <= 100) {
  JOptionPane.showMessageDialog(null, "正常な得点");
}
else {
  JOptionPane.showMessageDialog(null, "異常な得点");
}
```

数直線: AかつB

- 数直線にすると読みやすい

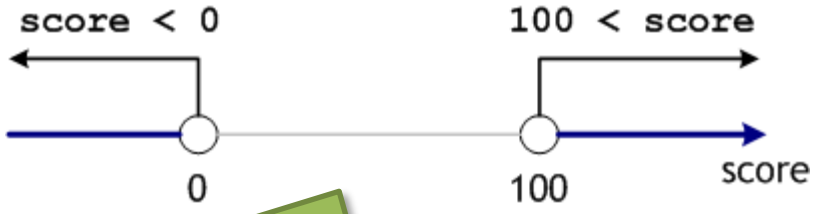


「0 <= score」と「score <= 100」が
重なり合うところだけが有効

「0 <= score <= 100」
とは書けない

AまたはB

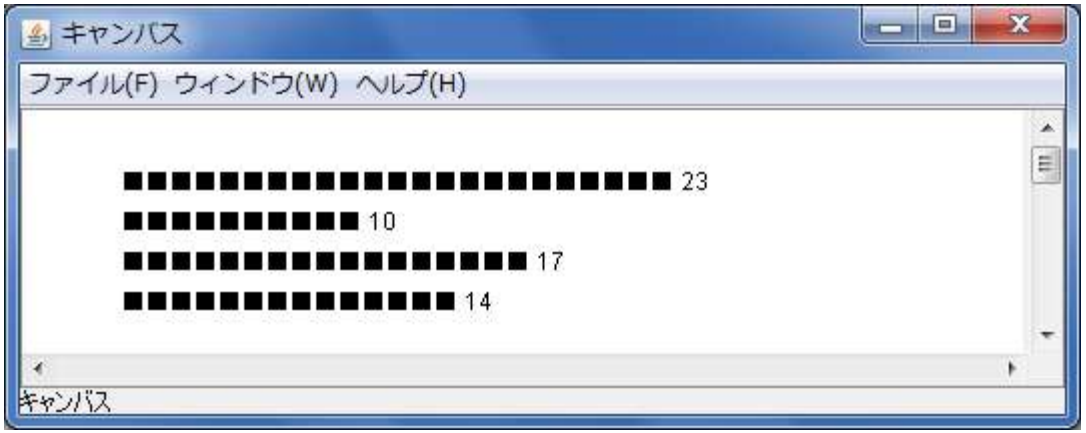
- 「 A || B 」で「AまたはB」となる
  - 「score < 0 || 100 < score」



「0 <= score」と「score <= 100」のどちらかがあれば有効

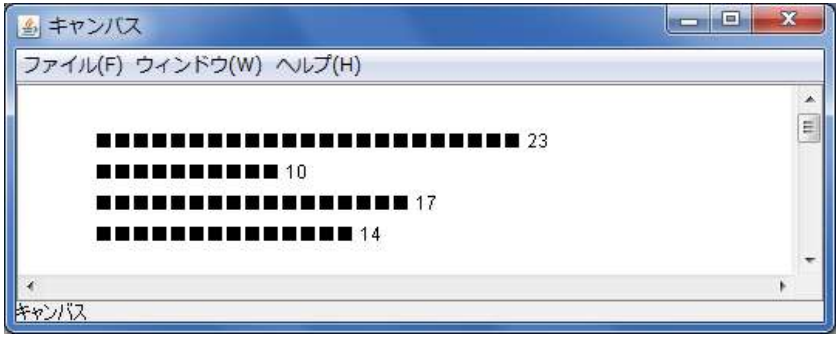
■例題31

問題：実行例のように棒グラフを表示するプログラムを作成せよ.



（クラス名： Ex31BarGraph）

例題31




```
1  package j2.lesson01;
2
3  import gpjava.Canvas;
4
5  public class Ex31BarGraph {
6      public static void main(String[] args) {
7          new Ex31BarGraph().start();
8      }
9
10     void start() {
11         Canvas.show();
12         String result;
13         result = "■■■■■■■■■■■■■■■■■■■■■■■ 23";
14         Canvas.drawString(50, 40, result);
15         result = "■■■■■■■■■■ 10";
16         Canvas.drawString(50, 60, result);
17         result = "■■■■■■■■■■■■■■■■■■■ 17";
18         Canvas.drawString(50, 80, result);
19         result = "■■■■■■■■■■■■■■ 14";
20         Canvas.drawString(50, 100, result);
21     }
22 }
```

Canvas の使い方の概要(1)

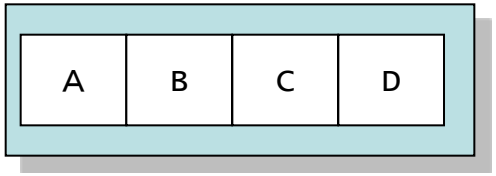
- package の次の行に、import 文を記入
  - import gpjava.Canvas;
- 実行の最初に、Canvas.show() を実行
- 表示用のメソッド群(上から、文字列、長方形、楕円)
  - Canvas.drawString(double x, double y, String message);
  - Canvas.drawRect(double x, double y, double width, double height);
  - Canvas.drawOval(double x, double y, double width, double height);
- 塗りつぶし用のメソッド群(上から、長方形、楕円)
  - Canvas.fillRect(double x, double y, double width, double height);
  - Canvas.fillOval(double x, double y, double width, double height);

Canvas の使い方の概要(2)

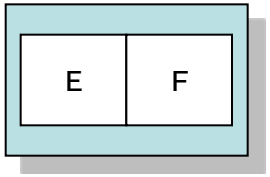
- 色(RGB)の指定
  - Canvas.setColor(int red, int green, int blue);
- 画面表示の消去(初期化)
  - Canvas.clear();
- 一定時間の休止
  - Canvas.waitForCountdown(int msec);
- マウスのクリックの認識
  - Canvas.waitForPoint(String message);
    - Canvas.getPointedX(); // X座標を返却
    - Canvas.getPointedY(); // Y座標を返却

文字列の詳細（1）

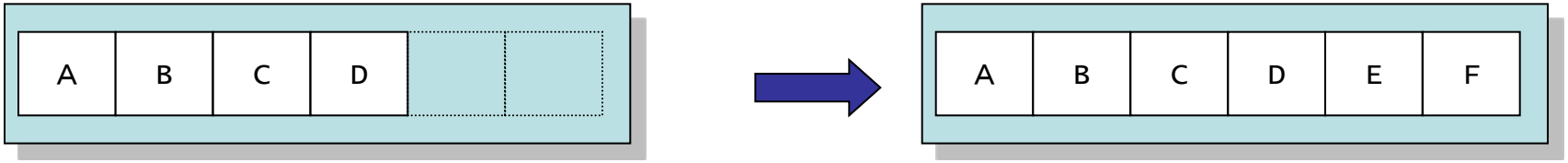
文字列“ABCD”



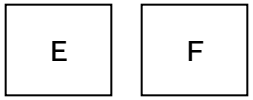
文字列“EF”



“ABCD” ＋ “EF”の処理（考え方）

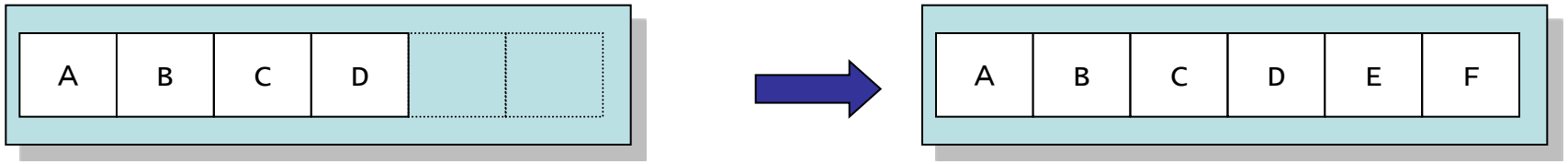


タイルを2枚分入れる場所を確保して
をはめる

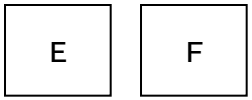


文字列の詳細(2)

“ABCD” + “EF”の処理(考え方)



タイルを2枚分入れる場所を確保して
をはめる

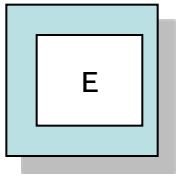


文字を定数として用いるときは
シングルクォートでその文字を囲む

```
char c1 = 'E';
char c2 = 'F';
```

文字列に含まれる、このような
タイルの一枚一枚を「文字」
(character)という。型名は「**char**」

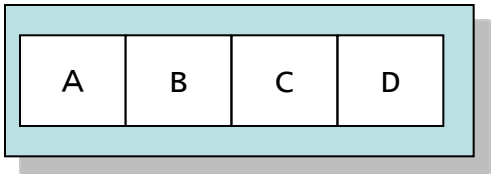
文字列は、タイルをはめる枠に
必要な分だけタイルをはめたものである。
右のものは長さ1の文字列(長さは次スライド)
で、文字「E」が含まれている。文字列と文字の
違いに注意が必要。

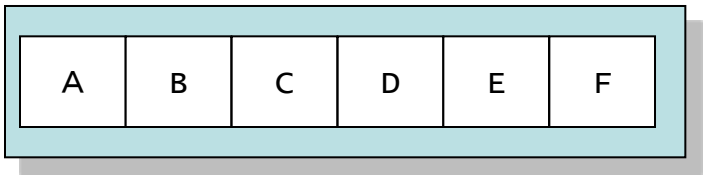



```
String s1 = "E";
```

文字列の長さ

文字列式.length() で文字列式の値（文字列）の長さが得られる



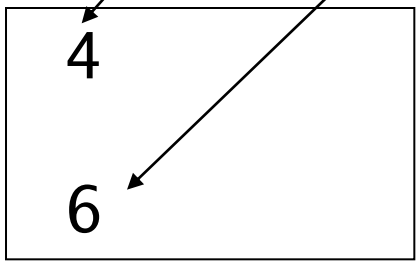


文字列”ABCD”の長さは4

文字列“ABCDEF”の長さは6

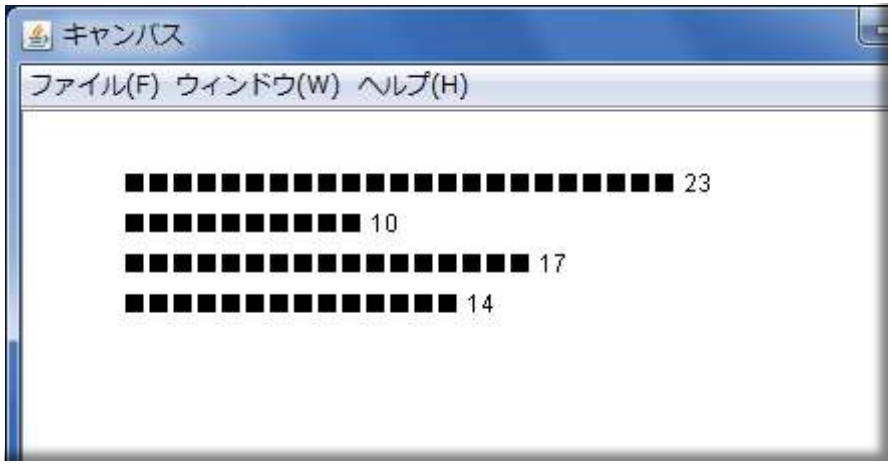
```
String s1 = “ABCD”;
String s2 = s1 + “EF”;

System.out.println(s1.length());
System.out.println(s2.length());
```



■例題32

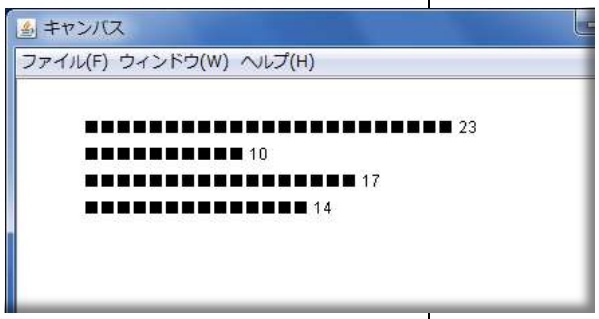
問題：実行例のように棒グラフを表示するプログラムを作成せよ。ただし、表示したい数は、変数a0, a1, a2, a3に格納されているものとする。



（クラス名： Ex32BarGraph2）

例題32

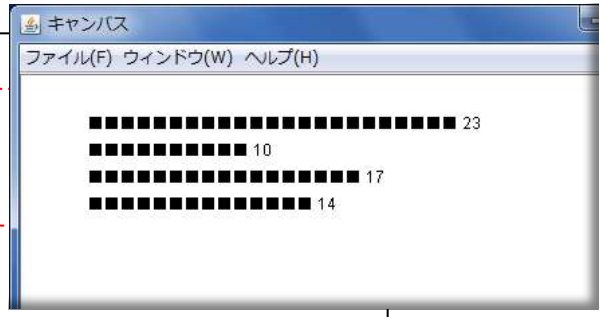
```
1  package j2.lesson01;
2
3  import gpjava.Canvas;
4
5  public class Ex32BarGraph2 {
6      public static void main(String[] args) {
7          new Ex32BarGraph2().start();
8      }
9
10     void start() {
11         int a0 = 23;
12         int a1 = 10;
13         int a2 = 17;
14         int a3 = 14;
15
16         Canvas.show();
17
18         String result = "";
19         for(int i = 0; i < a0; i++) {
20             result = result + "■";
21         }
22         result = result + " " + a0;
23         Canvas.drawString(50, 40, result);
24
```



変数a0で
指定された
回数(=23)
「■」を加えていく。

例題32（続き）

```
25          result = "";
26          for(int i = 0; i < a1; i++) {
27              result = result + "■";
28          }
29          result = result + " " + a1;
30          Canvas.drawString(50, 60, result);
31
32          result = "";
33          for(int i = 0; i < a2; i++) {
34              result = result + "■";
35          }
36          result = result + " " + a2;
37          Canvas.drawString(50, 80, result);
38
39          result = "";
40          for(int i = 0; i < a3; i++) {
41              result += "■";
42          }
43          result = result + " " + a3;
44          Canvas.drawString(50, 100, result);
45      }
46
47  }
```

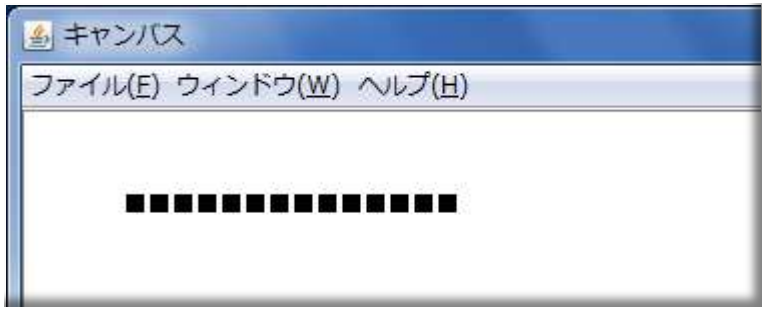


■例題33

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キャンバスに表示するメソッド 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
void	showBar(int n)	■をn個横に並べた文字列をキャンバスに表示する

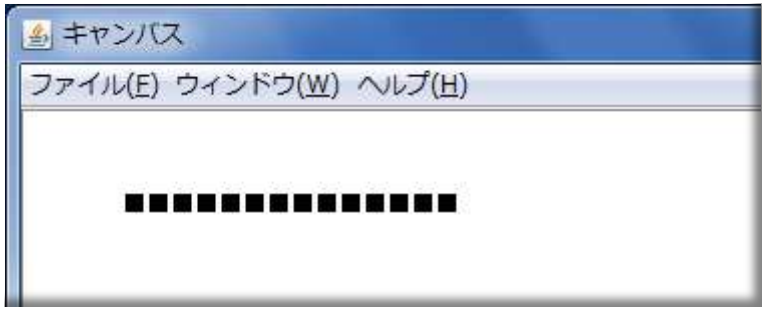
例：showBar(23)



（クラス名： Ex33ShowBar）

例題33

```
1  package j2.lesson01;
2
3  import gpjava.Canvas;
4
5  public class Ex33ShowBar {
6      public static void main(String[] args) {
7          new Ex33ShowBar().start();
8      }
9
10     void start() {
11         int a0 = 23;
12         int a1 = 10;
13         int a2 = 17;
14         int a3 = 14;
15
16         Canvas.show();
17
18         showBar(a0);
19         showBar(a1);
20         showBar(a2);
21         showBar(a3);
22     }
23
24     void showBar(int n) {
25         Canvas.clear();
26         String result = "";
27         String tile ="■";
28
29         for(int i = 0; i < n; i++) {
30             result = result + tile;
31         }
32         Canvas.drawString(50, 50, result);
33         Canvas.waitForCountdown(1000);
34     }
35 }
```



メソッドの宣言

- void … () {…} という部分が「メソッド」
  - start, greeting, showQuiz はメソッドの名前

```
void start() {
   …
}
```

startもメソッド

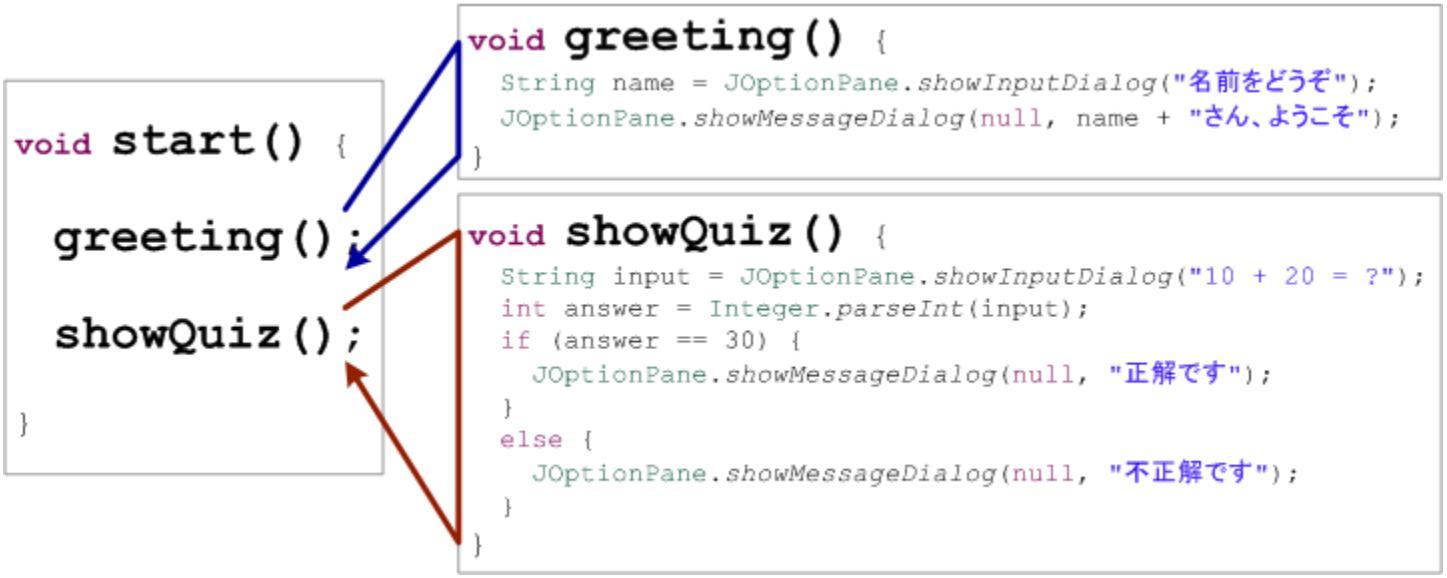
```
void greeting() {
   …
}
```

```
void showQuiz() {
   …
}
```

「メソッドの宣言」と呼ぶ

メソッドの起動

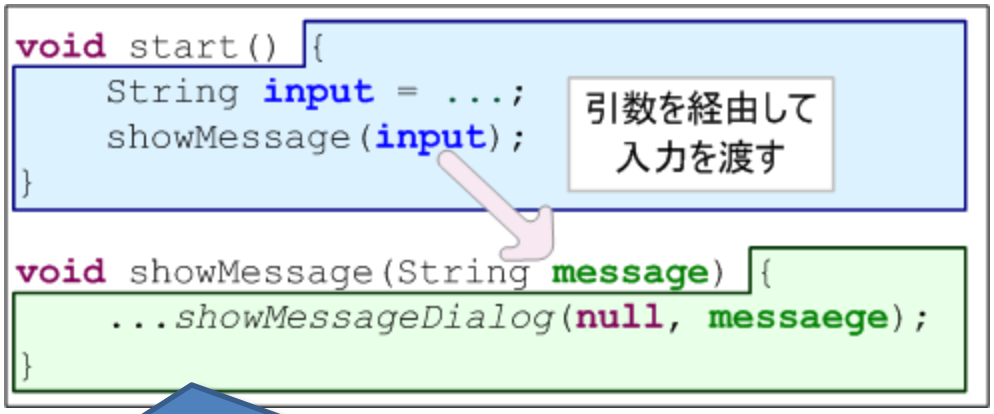
- greeting(); やshowQuiz(); という命令で宣言したメソッドの内容を処理
  - void greeting() {...} などの{...}を実行する



「メソッドの起動」と呼ぶ

メソッドの引数(ひきすう)

- 仮引数と実引数を組み合わせるとメソッドに値を渡せる
  - 実引数: startのinputの内容
  - 仮引数: showMessageのmessage



startで入力した値を
showMessageで使える

メソッドの宣言と起動

メソッドの宣言

```
void <メソッドの名前>(<仮引数宣言1>, …, <仮引数宣言N>) {
    <命令>
}
```

メソッドの起動

```
<メソッドの名前>(<実引数1>, …, <実引数N>);
```

仮引数の宣言

```
String <仮引数の名前>
int <仮引数の名前>
double <仮引数の名前>
```

メソッド起動時に
同じ位置の実引数が
仮引数に記憶される

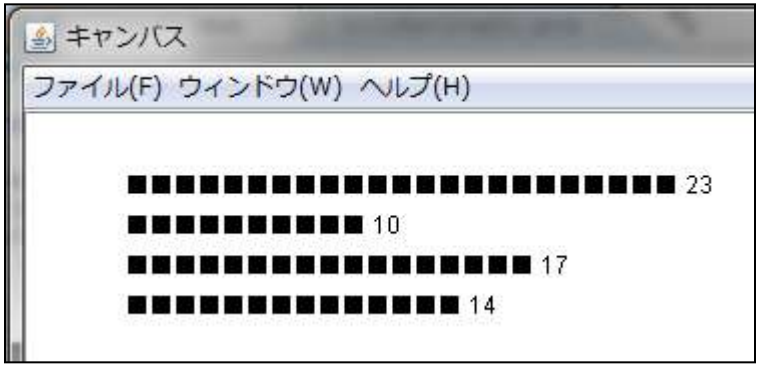
メソッドを利用することの利点

- 命令を束ねて使うことができる
  → プログラムを短くできる
- 束ねた部分の仕事(機能)に名前をつけられる
  →機能に名前をつけることで、人間が詳細(実装)を忘れることができる。(捨象、抽象化、abstraction)
  → プログラムが見やすくなる
  もし名前が適切に付けられていれば、名前から機能を推測することができる。中身の詳細を知らずに機能を利用できる。
- 一連の処理を一つにまとめることができる
  → プログラムに間違いがあっても修正しやすい

■例題34（重要）

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。

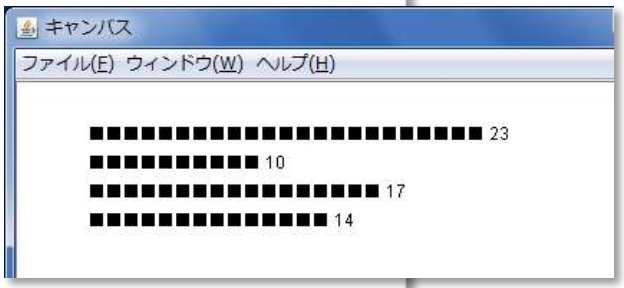
返り値の型	メソッド名(引数)	機能
String	makeBar(int n)	■をn個横に並べた文字列を返す。



（クラス名： Ex34MakeBar）

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。

値を返すメソッドの宣言

- int add(…) {…} という形で宣言している
  - 前回は void add(…) {…} という形

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

「整数(int)を返す」という宣言

実数(double)や文字列(String)も指定できる

return文

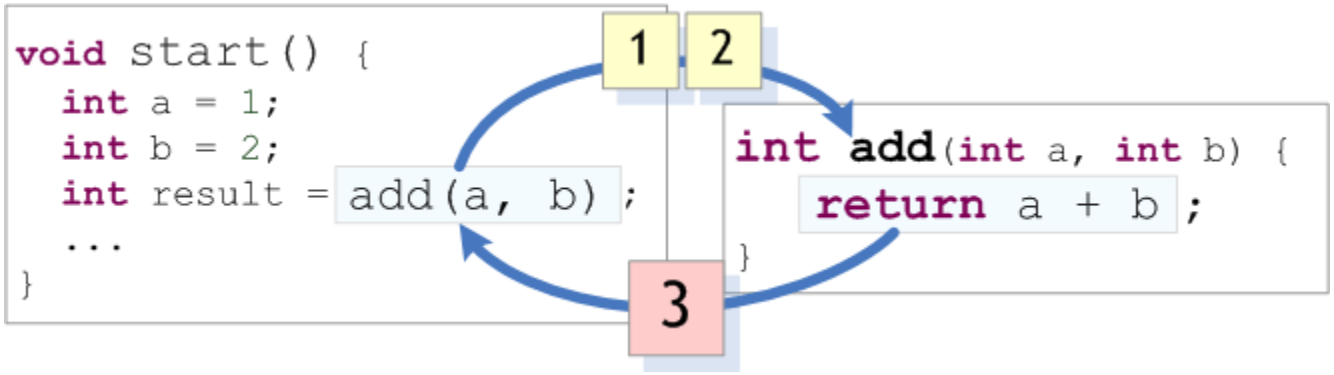
- addメソッドの中に**return**から始まる命令
  - 返す値を指定している

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

この命令を「return文」と呼ぶ

戻り値の受け取り

- メソッド起動をリテラルの代わりに書く
  - return文が返す「戻り値」を起動元で受け取れる



「add(1, 2) * 3」のように
ほかの式とも組み合わせられる

値を返すメソッド

メソッドの宣言

```
<戻り値の型> <メソッドの名前>(<仮引数の一覧>) {
    <命令>
    return <起動元に返す値>;
}
```

リテラルの代わりに書くと
返した値を受け取れる

論理値を返すメソッド

- 条件式として使うメソッドはboolean型を指定

メソッドの宣言

```
boolean <メソッドの名前>(<仮引数の一覧>) {
    <命令>
    return <trueまたはfalse>;
}
```

メソッドの起動

```
if (<メソッドの名前>(<実引数の一覧>)) {
    <メソッドがtrueを返す場合の命令>
}
if ( ! <メソッドの名前>(<実引数の一覧>)) {
    <メソッドがfalseを返す場合の命令>
}
```

■例題41

問題：実行例のようにn個の整数値をユーザから得て、
その平均値を求めるメソッドaverage 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
double	average( )	ダイアログで何回、整数値を入力するかユーザーに聞き、その回数だけ整数値を入力させ、その平均値を返す。

(クラス名：
Ex41Average)

例題41

```java
package j2.lesson01;

import gpjava.Spreadsheet;
import javax.swing.JOptionPane;

public class Ex41Average {
    public static void main(String[] args) {
        new Ex41Average().start();
    }

    void start() {
        double ave = average();
        Spreadsheet.show();

        Spreadsheet.setString(0, 0, "平均値");
        Spreadsheet.setDouble(0, 1, ave);
    }

    double average() {
        String inputNumData = JOptionPane.showInputDialog("何個のデータを入力しますか？");
        int numData = Integer.parseInt(inputNumData);

        String message = "整数を入力してください。";
        String inputIntNum;
        int intNum;
        int sum = 0;

        for(int i = 0; i < numData; i++) {
            inputIntNum = JOptionPane.showInputDialog(message);
            intNum = Integer.parseInt(inputIntNum);
            sum = sum + intNum;
        }

        return (double)sum / numData;
    }
}
```

例題41補足(明示的な型変換)

```
            sum += intNum;
        }
        return (double) sum / numData;
    }
```

- 6 / 4 → 1 整数型(int) 6を4で割ったときの商
- 6.0 / 4 → 1.5 実数型(double)

-sum/numData→ 整数型(int型).
  -sumをnumDataで割ったときの商
  -すなわち小数部は切り捨てられる

-(double)sum / numData →
  -sumをdouble型の値に変換し、
  -int型のnumDataで割る。(結果はdouble型)

■例題42

問題：次のメソッドaverage, max, contentStringを作成し、その挙動を確かめよ。

返り値の型	メソッド名(引数)	機能
double	average(int [ ] array)	引数の配列の全要素の平均値を返す。
int	max(int [ ] array)	引数の配列の全要素の中から一番大きい値を返す。
String	contentString(int[] values	引数の配列を適当な文字列表現にして返す。

（クラス名：Ex42AverageMax）

例題42

```
1  package j2.lesson01;
2
3  import gpjava.Spreadsheet;
4
5  public class Ex42AverageMax {
6      public static void main(String[] args) {
7          new Ex42AverageMax().start();
8      }
9
10     void start() {
11         Spreadsheet.show();
12         int[] a = {23, 10, 17, 13};
13         String content = contentString(a);
14         Spreadsheet.setString(0, 0, "データ");
15         Spreadsheet.setString(0, 1, content);
16         Spreadsheet.setString(1, 0, "平均値");
17         Spreadsheet.setDouble(1, 1, average(a));
18         Spreadsheet.setString(2, 0, "最大値");
19         Spreadsheet.setInt(2, 1, max(a));
20     }
21
```

例題42(続き)

```
22    double average(int[] array) {
23        int sum = 0;
24
25        for(int i = 0; i < array.length; i++) {
26            sum = sum + array[i];
27        }
28        return (double)sum / array.length;
29    }
30
31    int max(int[] array) {
32        int currentMax = array[0];
33
34        for(int i = 1; i < array.length; i++) {
35            if(currentMax < array[i]) {
36                currentMax = array[i];
37            }
38        }
39
40        return currentMax;
41    }
42
43    String contentString(int[] values) {
44        String result = "{";
45
46        for(int i = 0; i < values.length; i++) {
47            result = result + values[i] + " ";
48        }
49
50        result = result + "}";
51        return result;
52    }
53 }
```

例題42(説明)

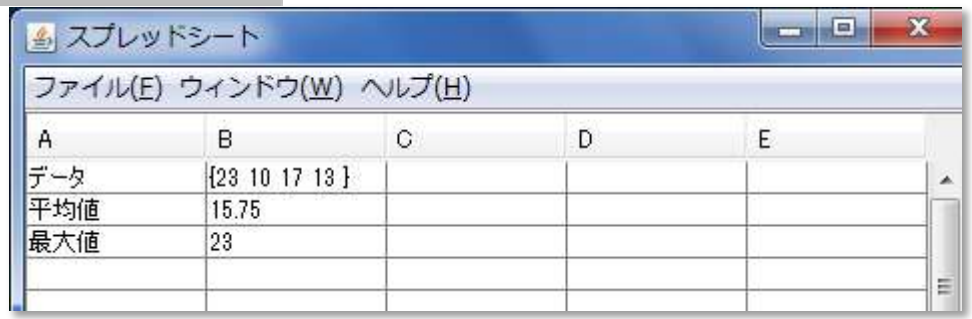
```
5  public class Ex42AverageMax {
6      public static void main(String[] args) {
7          new Ex42AverageMax().start();
8      }
9
10     void start() {
11         Spreadsheet.show();
12         int[] a = {23, 10, 17, 13};
13         String content = contentString(a);
14         Spreadsheet.setString(0, 0, "データ");
15         Spreadsheet.setString(0, 1, content);
16         Spreadsheet.setString(1, 0, "平均値");
17         Spreadsheet.setDouble(1, 1, average(a));
18         Spreadsheet.setString(2, 0, "最大値");
19         Spreadsheet.setInt(2, 1, max(a));
```

配列の生成

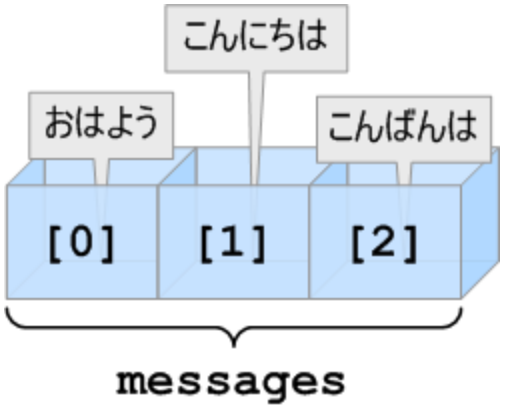
配列の内容(contents)を文字列として取得。この場合「{23 10 17 13 }」という文字列

配列の要素の平均をとる

配列の要素の最大値を選ぶ



配列の生成



配列の生成

```
int[] <変数の名前> = new int[<配列の長さ>];
double[] <変数の名前> = new double[<配列の長さ>];
String[] <変数の名前> = new String[<配列の長さ>];
```

末尾に[]が付いた型を「配列型」と呼ぶ

「配列の長さ」は配列に記憶させられる要素の個数

配列要素の使用

- 配列要素の位置をインデックスで指定する

配列要素への代入

```
<変数の名前>[<インデックス>] = <代入する値>;
```

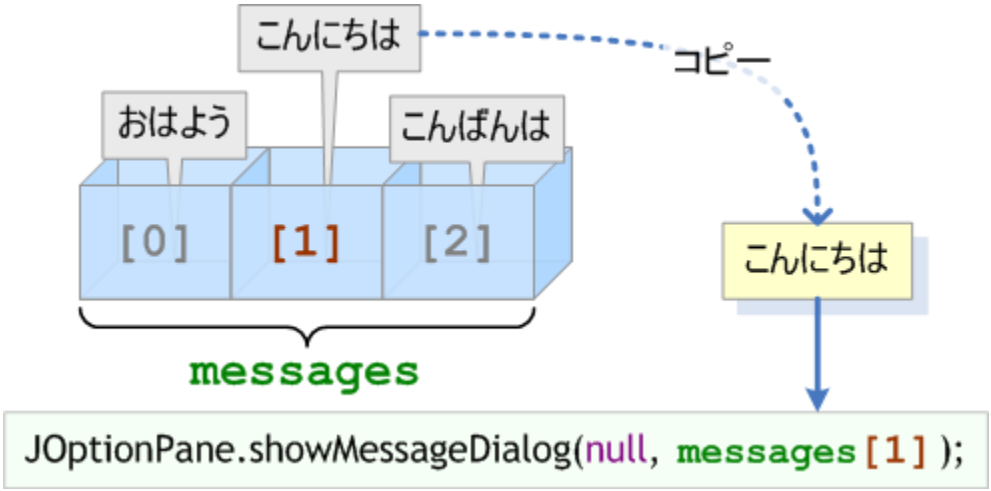
配列要素の参照

```
<変数の名前>[<インデックス>]
```

インデックスは0から始まり
(配列の長さ-1)まで

配列要素の参照

- 「messages[1] 」のような書き方で配列から値をコピーして使える
  - 指定したインデックスの位置の要素を使う



複数の値の中から1つを選んで
取り出すことができる

配列とメソッド

- 配列を受け取るメソッド
  - メソッドの仮引数の型に配列型を指定
  - メソッドの実引数に配列をそのまま渡す
- 配列を返すメソッド
  - メソッドの戻り値型に配列型を指定
  - return文で配列をそのまま返す

「配列をそのまま」の際には
インデックスを指定しない

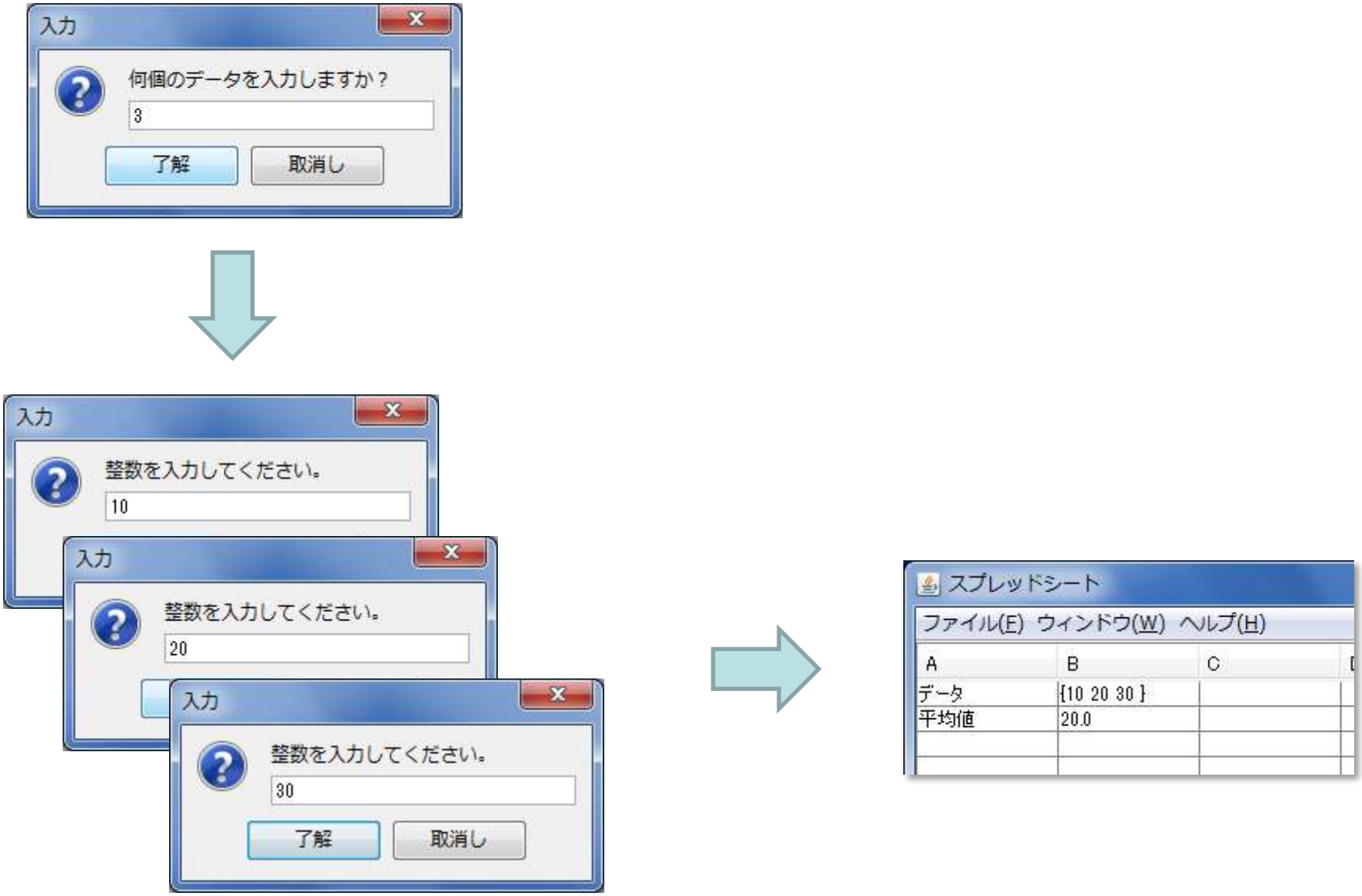
■例題43

問題：実行例のようにn個の数値をユーザから得て、int型配列として数値列を返すメソッド inputIntegersを作成せよ。Ex41で作成したメソッドを用いて、入力された数値列に 対する平均値を求め表示せよ。

返り値の型	メソッド名(引数)	機能
double	*average(int [ ] array)*	*（例題42と同じ。．．引数の配列の全要素の平均値を返す。）*
String	*contentString(int[ ] values*	*（例題42と同じ。．．．引数の配列を適当な文字列表現にして返す。）*
int [ ]	inputIntegers	int型の配列を作成し、返す。 ユーザーは配列の大きさと、要素をダイアログで入力する。

（クラス名： Ex43InputIntegers）

実行例


入力
何個のデータを入力しますか？
3
了解
取消し
入力
整数を入力してください。
10
入力
整数を入力してください。
20
入力
整数を入力してください。
30
了解
取消し
スプレッドシート
ファイル(F) ウィンドウ(W) ヘルプ(H)
A
B
C
データ
{10 20 30 }
平均値
20.0

例題43

```
 1  package j2.lesson01;
 2
 3  import gpjava.Spreadsheet;
 4  import javax.swing.JOptionPane;
 5
 6  public class Ex43InputIntegers {
 7
 8      public static void main(String[] args) {
 9          new Ex43InputIntegers().start();
10      }
11
12      void start() {
13          Spreadsheet.show();
14          int[] a = inputIntegers();
15          String content = contentString(a);
16          Spreadsheet.setString(0, 0, "データ");
17          Spreadsheet.setString(0, 1, content);
18          Spreadsheet.setString(1, 0, "平均値");
19          Spreadsheet.setDouble(1, 1, average(a));
20      }
21
```

例題43（続き）

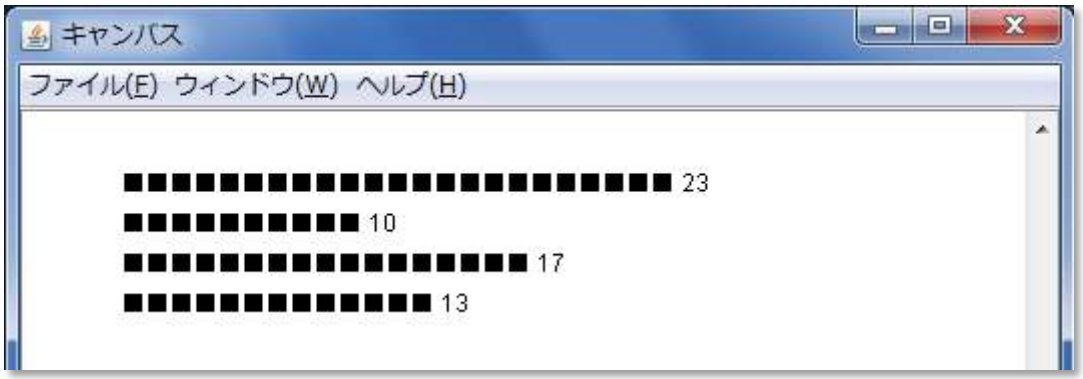
```
22  int[] inputIntegers() {
23      String inputNumData = JOptionPane.showInputDialog("何個のデータを入力しますか？");
24      int numData = Integer.parseInt(inputNumData);
25      int[] result = new int[numData];
26
27      String message = "整数を入力してください。";
28      String inputIntNum;
29      int intNum;
30
31      for(int i = 0; i < numData; i++) {
32          inputIntNum = JOptionPane.showInputDialog(message);
33          intNum = Integer.parseInt(inputIntNum);
34          result[i] = intNum;
35      }
36
37      return result;
38  }
39
40  double average(int[] array) {
41      int sum = 0;
42
43      for(int i = 0; i < array.length; i++) {
44          sum = sum + array[i];
45      }
46      return (double)sum / array.length;
47  }
48
49  String contentString(int[] values) {
50      String result = "{";
51
52      for(int i = 0; i < values.length; i++) {
53          result = result + values[i] + " ";
54      }
55
56      result = result + "}";
57      return result;
58
59  }
60  }
```

（前の例題と同じ）

■例題44

問題：ある数値列がint型配列データに格納されているものとする。この数値列を棒グラフとして表示するメソッドmakeGraphを作成せよ。

返り値の型	メソッド名(引数)	機能
String	*makeBar(int n)*	*（前と同じ．．．■をn個横に並べた文字列を返す。）*
void	showGraph(int [ ]anArray)	引数の配列を基に、棒グラフをダイアログで表示する。



（クラス名： Ex44MakeGraph）

例題44

（前の例題と同じ）

```
1  package j2.lesson01;
2
3  import gpjava.Canvas;
4
5  public class Ex44MakeGraph {
6      public static void main(String[] args) {
7          new Ex44MakeGraph().start();
8
9
10     void start() {
11         int[] a = {23, 10, 17, 13};
12         showGraph(a);
13     }
14
15     String makeBar(int n) {
16         String result = "";
17         String tile ="■";
18
19         for(int i = 0; i < n; i++) {
20             result = result + tile;
21         }
22
23         return result;
24     }
25
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     }
35 }
```