資料編

図 1-1　レイアウト


Javaプログラミング講座
HEADER
index
chapter1 初めてのJavaプログラミング
chapter2 Javaプログラミングの基礎
chapter3 配列、for文、カプセル化、if文
chapter4 問題：BMI値を計算するプログラム
chapter5 JavaでGUIアプリケーションを作る
chapter6 画面のレイアウトとイベント処理
chapter7 BMI計算アプリを作ろう
chapter8 問題：数当てゲームを作ろう
MENU
メニューエリア
メインエリア
FOOTER
Copyright (C) 2012 Saki Fujio All Rights Reserved.
Javaプログラミング講座ついて
Java開発環境の準備
参考リンク
参考文献
style.css
#HEADER {
width: 302px;
position: fixed;
left: 20px;
top: 45px;
text-align: right;
}
#PAN {display: none;}
#MENU {
width: 300px;
position: fixed;
left: 20px;
top: 150px;
background-color: #FFFFFF;
border: 1px solid #999999;
}
#KIZI {
line-height: 1.5;
height: 100%;
margin-left: 350px;
padding-top: 1px;
max-width: 60em;
background-color: #FFFFFF;
border: solid #999999;
border-width: 0 1px 0 1px;
}
#FOOTER {
text-align: center;
font-size: 85%;
width: 300px;
position: fixed;
left: 20px;
bottom: 10px;
}
〈DIV〉クラスの
ID#HEADER
ID#MENU
ID#FOOTER に
「position: fixed;」
を指定することで
絶対位置に配置

図 1-2　IE6 用疑似フレーム

```
/* ── IE6用(擬似フレーム) ── */
* html,* html body {
	overflow-y: hidden;
	display: block;
	height: 100%;
}
* html div#HEADER,* html div#MENU,* html div#FOOTER{
	position: absolute;
}
* html div#KIZI {
	overflow:auto;
}
```

図 2　BMI 値を計算するプログラム

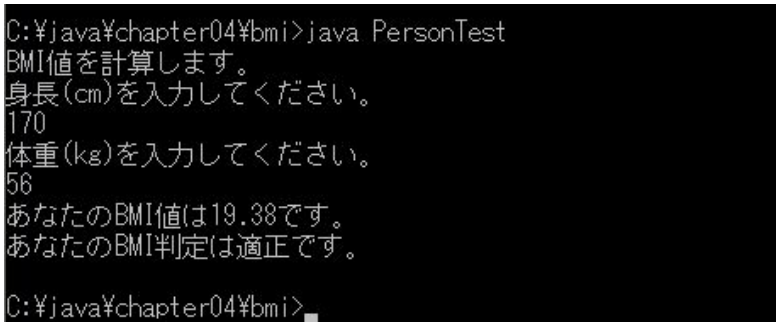
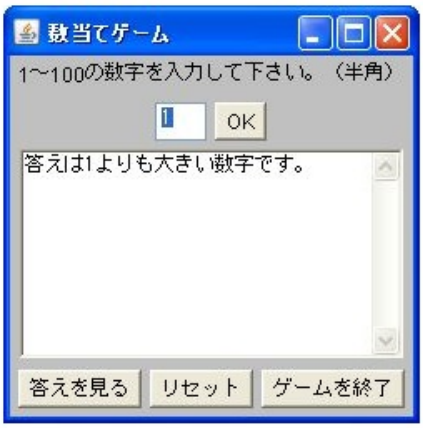


図 3-1　数当てゲーム

図 3-2　数当てゲーム（入力時）



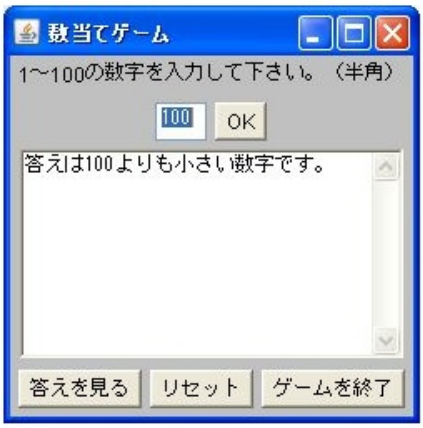


図４　トップページ


Javaプログラミング講座
index
chapter1 初めてのJavaプログラミング
chapter2 Javaプログラミングの基礎
chapter3 配列、for文、カプセル化、if文
chapter4 問題：BMI値を計算するプログラム
chapter5 JavaでGUIアプリケーションを作る
chapter6 画面のレイアウトとイベント処理
chapter7 BMI計算アプリを作ろう
chapter8 問題：数当てゲームを作ろう
Copyright (C) 2012 Saki Fujio All Rights Reserved.
Javaプログラミング講座ついて
この講座では、Javaプログラミングの基礎から、AWTを使用したGUIアプリケーション作成までを学ぶ事が出来ます。
これからJavaを始める人の為に、コマンドプロンプトの設定やプログラムの実行の仕方を始め、クラスやメソッド、配列なども、サンプルプログラムを交えながら画像つきで詳しく解説しています。
この講座は、PythonやPHPなど他のプログラミング言語を学んだことがある人が、次のステップとしてJavaの基礎を学習することを想定して作られています。
初心者にも分かりやすく説明しているつもりですが、全くのプログラミング初心者ではところどころ難しいこともあるかもしれません。
そういった方は、Javaの参考書か、Javaの入門サイトなどと合わせて読みながら勉強して下さい。
TOP
Java開発環境の準備
Javaでアプリケーションを開発するには、JDK というJava言語でプログラミングを行う際に必要な最低限のソフトウェアのセットをインストールする必要があります。これはJavaの開発元であるSun Microsystems社が開発、配布しています。
この講座では、JDKのインストール方法を紹介していないので、以下のサイトを参考にして事前に開発環境を整えてからchapterの内容に進んで下さい。また、chapter1でもこれと同じ注意書きをしています。
＊JavaDrive - JDK 6のインストールと環境設定
MacOSXの場合は、Java開発環境が元々備わっているので、この作業は必要ありません。
TOP
参考リンク
JavaTM 2 Platform Standard Edition 5.0 API 仕様
Java初心者入門講座
JavaDrive
Java AWT 入門
とほほのJava入門
TOP
参考文献
堀江 幸生(2007) 『世界でいちばん簡単なJavaのe本―Javaの基本と考え方がわかる本』 秀和システム
鳥海 不二夫(2007) 『よくわかるJava』 秀和システム
掌田 津耶乃(2010) 『Eclipse 3.6ではじめるJavaプログラミング入門―Eclipse 3.6 Helios対応』 秀和システム
TOP

図５　chapter1 初めての Java プログラミング

図 6　chapter2 Java プログラミングの基礎

Javaプログラミング講座

index
chapter1 初めてのJavaプログラミング
chapter2 Javaプログラミングの基礎
- ソースコードの骨格
- クラス
- mainメソッド
- Javaプログラミングの基礎
- フィールドとコンストラクタ
- クラスのインスタンス化
- メソッド
- まとめと次回予告

chapter3 配列、for文、カプセル化、if文
chapter4 問題：BMI値を計算するプログラム
chapter5 JavaでGUIアプリケーションを作る
chapter6 画面のレイアウトとイベント処理
chapter7 BMI計算アプリを作ろう
chapter8 問題：数当てゲームを作ろう

Copyright (C) 2012 Saki Fujio All Rights Reserved.

chapter2. プログラミングの基礎

ソースコードの骨格

Sample.javaの解説

はじめに前回書いた「Sample.java」のソースコードについて解説しながら前回の復習をしていきましょう。

Sampleプログラムは、「こんにちは」 という文字列をディスプレイなどの出力装置に出力するだけのとても簡単なプログラムです。
「""」ダブルクォーテーションで囲んだ文字列を変えると、出力される文字列が変わるので試してみて下さい。

Sample.java

```
class Sample {
  public static void main(String[] args) {
    System.out.println("こんにちは");
  }
}
```

このソースコードの骨格を整理すると、大体以下のようになります。

クラス名.java

```
class クラス名{
  mainメソッド( 引数 ) {
    …実行する処理…
  }
}
```

第１章でも書いた通り、ソースコードのファイル名は「クラス名.java」という規則になっています。
前回は言いませんでしたが、一般的にクラス名は単語の先頭を大文字にします。
クラス名を複数の単語で構成する場合は、各単語の先頭だけ大文字にします。　例：SampleTest
そしてこのソースコードをコンパイルすると、「クラス名.class」というクラスファイルが作成されます。
実行するときにはこのクラスファイルを実行するんでしたね。
これらの作業はこれから何度も行うことになるので、しっかり復習しておいて下さい。

このソースコードの骨格にあるのは、 **クラス** と **mainメソッド** と呼ばれるものです。
それぞれについて、以下で詳しく説明します。

TOP

クラス

先ほどのソースコードのように、Javaのプログラムは **クラス** 内に記述することになっています。

クラスについて

クラスとは何か、と書き始めると長くなるのでここでは簡単に説明します。

Javaでは何か処理をさせようとする時、まずその処理を行う"モノ"を作り、処理を行わせます。
目的の処理を行わせるには、その"モノ"がどんな機能を持っていてどんなデータを扱うのかを正しく設計する必要があります。
その作り出す"モノ"の「設計図」が **クラス** です。
つまり目的に応じたクラスを設計し、クラスから"モノ"を作り、その"モノ"に処理を実行させるというのがプログラムの流れとなるわけです。

例として計算機という"モノ"を作り出すことを考えてみます。
計算機に必要なものは加算や減算といったデータを処理する「機能」と、計算対象や計算結果などの「データ」に分けることができます。
Javaではこの「機能」を **メソッド** と呼び「データ」を **フィールド** と呼びます。
Sampleプログラムでも、クラスの中にメソッドを書き、そのメソッドに実行させたい処理を記述しています。
今回は単純なプログラムなのでフィールドの定義はありませんが、大体クラスは以下のような形になると思ってください。

クラスの定義

```
修飾子 class クラス名 {

  フィールド1;
  フィールド2;
  ...

  メソッド1 {
    メソッドの定義;
  }

  メソッド2 {
   メソッドの定義;
  }

  ...

 }
```

修飾子の後に、 **「class クラス名」** でクラスを定義することができます。
修飾子についてはいずれ説明します。とりあえずは、こういうものが付くこともある程度に覚えておいて下さい。

クラス名の後に **「{」** を書き、フィールドやメソッドなどクラスの持つデータや機能を書いたら、最後にまた **「}」** で閉じます。
この { から } の間を、 **ブロック** といいます。

今の段階では詳しく理解できなくてもかまいません。とりあえず以上の点を押さえておいてください。

TOP

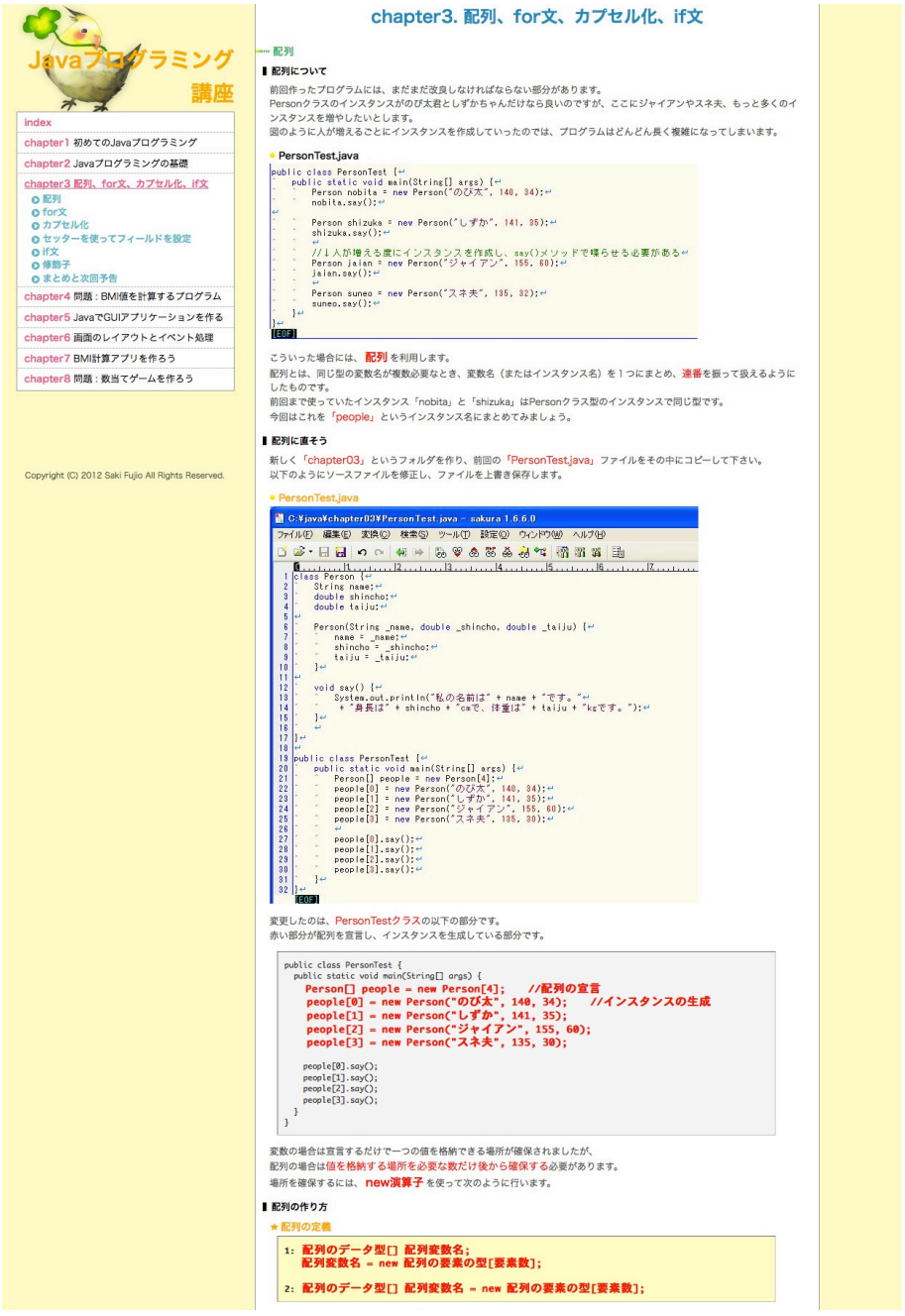
mainメソッド

Javaアプリケーションでは **main()メソッド** を定義する必要があります。

mainメソッドについて

「java クラス名」で実行した場合、引数に指定したクラスの中に記述された **mainメソッド** が呼び出されることになって

図 7　chapter3 配列、for 文、カプセル化、if 文



Javaプログラミング講座

index
chapter1 初めてのJavaプログラミング
chapter2 Javaプログラミングの基礎
chapter3 配列、for文、カプセル化、if文
- 配列
- for文
- カプセル化
- セッターを使ってフィールドを設定
- if文
- 修飾子
- まとめと次回予告

chapter4 問題：BMI値を計算するプログラム
chapter5 JavaでGUIアプリケーションを作る
chapter6 画面のレイアウトとイベント処理
chapter7 BMI計算アプリを作ろう
chapter8 問題：数当てゲームを作ろう

chapter3. 配列、for文、カプセル化、if文

配列

配列について

前回作ったプログラムには、まだまだ改良しなければならない部分があります。
Personクラスのインスタンスがのび太君としずかちゃんだけなら良いのですが、ここにジャイアンやスネ夫、もっと多くのインスタンスを増やしたいとします。
図のように人が増えるごとにインスタンスを作成していったのでは、プログラムはどんどん長く複雑になってしまいます。

PersonTest.java

```
public class PersonTest {
    public static void main(String[] args) {
        Person nobita = new Person("のび太", 140, 34);
        nobita.say();

        Person shizuka = new Person("しずか", 141, 35);
        shizuka.say();

        //1人が増える度にインスタンスを作成し、say()メソッドで喋らせる必要がある
        Person jaian = new Person("ジャイアン", 155, 60);
        jaian.say();

        Person suneo = new Person("スネ夫", 135, 32);
        suneo.say();
    }
}
[EOF]
```

こういった場合には、**配列**を利用します。
配列とは、同じ型の変数名が複数必要なとき、変数名（またはインスタンス名）を1つにまとめ、連番を振って扱えるようにしたものです。
前回まで使っていたインスタンス「nobita」と「shizuka」はPersonクラス型のインスタンスで同じ型です。
今回はこれを「people」というインスタンス名にまとめてみましょう。

配列に直そう

新しく「chapter03」というフォルダを作り、前回の「PersonTest.java」ファイルをその中にコピーして下さい。
以下のようにソースファイルを修正し、ファイルを上書き保存します。

PersonTest.java

```
C:¥java¥chapter03¥PersonTest.java - sakura 1.6.6.0
ファイル(F) 編集(E) 変換(C) 検索(S) ツール(T) 設定(O) ウィンドウ(W) ヘルプ(H)
 1 class Person {
 2     String name;
 3     double shincho;
 4     double taiju;
 5
 6     Person(String _name, double _shincho, double _taiju) {
 7         name = _name;
 8         shincho = _shincho;
 9         taiju = _taiju;
10     }
11
12     void say() {
13         System.out.println("私の名前は" + name + "です。"
14             + "身長は" + shincho + "cmで、体重は" + taiju + "kgです。");
15     }
16
17 }
18
19 public class PersonTest {
20     public static void main(String[] args) {
21         Person[] people = new Person[4];
22         people[0] = new Person("のび太", 140, 34);
23         people[1] = new Person("しずか", 141, 35);
24         people[2] = new Person("ジャイアン", 155, 60);
25         people[3] = new Person("スネ夫", 135, 30);
26
27         people[0].say();
28         people[1].say();
29         people[2].say();
30         people[3].say();
31     }
32 }
    [EOF]
```

変更したのは、PersonTestクラスの以下の部分です。
赤い部分が配列を宣言し、インスタンスを生成している部分です。

```
public class PersonTest {
  public static void main(String[] args) {
    Person[] people = new Person[4];    //配列の宣言
    people[0] = new Person("のび太", 140, 34);    //インスタンスの生成
    people[1] = new Person("しずか", 141, 35);
    people[2] = new Person("ジャイアン", 155, 60);
    people[3] = new Person("スネ夫", 135, 30);

    people[0].say();
    people[1].say();
    people[2].say();
    people[3].say();
  }
}
```

変数の場合は宣言するだけで一つの値を格納できる場所が確保されましたが、
配列の場合は値を格納する場所を必要な数だけ後から確保する必要があります。
場所を確保するには、**new演算子**を使って次のように行います。

配列の作り方

★配列の定義

```
1: 配列のデータ型[] 配列変数名;
   配列変数名 = new 配列の要素の型[要素数];

2: 配列のデータ型[] 配列変数名 = new 配列の要素の型[要素数];
```

図 8　chapter4 問題：BMI 値を計算するプログラム

Javaプログラミング講座

- index
- chapter1 初めてのJavaプログラミング
- chapter2 Javaプログラミングの基礎
- chapter3 配列、for文、カプセル化、if文
- chapter4 問題：BMI値を計算するプログラム
  - 問題：BMIを計算しよう
  - Step1：計算するメソッドを作ろう
  - Step2：BMI値を元に判定しよう
  - Step3：数値をフォーマットしよう
  - BMI計算機を作るには？
  - BMI計算機を作ろう
  - まとめと次回予告
- chapter5 JavaでGUIアプリケーションを作る
- chapter6 画面のレイアウトとイベント処理
- chapter7 BMI計算アプリを作ろう
- chapter8 問題：数当てゲームを作ろう

chapter4. 問題：BMI値を計算するプログラム

問題：BMIを計算しよう

問題

メソッド は他のプログラミング言語で言う**関数**のようなもの、という事は話しました。
今回は、このメソッドを使って計算するプログラムを考えて貰いたいと思います。

これまでの章で使ってきたPersonクラスを少し改良して、BMI値の計算結果が表示されるようにしてみました。

- **PersonTest.java**

```
C:¥java¥chapter04>java PersonTest
名前はのび太です。身長は140.0cmで、体重は34.0kgです。
BMI値は17.35で、判定は痩せすぎです。

名前はしずかです。身長は141.0cmで、体重は35.0kgです。
BMI値は17.6で、判定は痩せすぎです。

名前はジャイアンです。身長は155.0cmで、体重は60.0kgです。
BMI値は24.97で、判定は適正です。

名前はスネ夫です。身長は135.0cmで、体重は30.0kgです。
BMI値は16.46で、判定は痩せすぎです。
```

それぞれのインスタンスから**身長**と**体重の数値**を受け取り、**BMI値を計算**し、**判定結果を表示**するようにしました。
これと同じプログラムを書いてもらうのが、今回の練習問題です。
まだJava始めたばっりなのにそんなの無理！！と思うかもしれませんが、少し書き方が違うだけで他のプログラミング言語と行っている処理は同じです。
なのでプログラミングをかじった事がある人ならそう難しい問題では無いでしょう。

どんな機能が必要か整理しよう

プログラムに絶対欠かせないのは設計図となる **クラス**、そしてデータを保存しておく **フィールド**、最後に機能部分となる **メソッド** です。
まだJavaが苦手な人の為に、今回必要なものを箇条書きにしてみた。

- **Personクラス**

1. フィールド
2. コンストラクタ
3. say()メソッド
4. BMI値を計算するメソッド
5. 計算で得た値を判定するメソッド

- **PersonTestクラス**

1. mainメソッド
2. インスタンスの作成
3. say()メソッドを呼び出して表示する

フィールドやコンストラクタ、インスタンスの作成などはこれまでの章でやってきた事と同じです。
いきなり全部を書くのが難しい人は、これまでのファイルを基盤にすると良いでしょう。出来る人は出来るだけ自分で作ってみて下さいね。
どんな変数やメソッドが必要か、どんな処理を書けば良いかはコメントで整理しておくと、後でメソッドを作るときに役立ちます。

TOP

Step1：計算するメソッドを作ろう

数値の計算

設計図が出来たら、次は **メソッド** に具体的な機能を持たせる必要があります。
まずはBMI値を計算するメソッドから記述していきましょう。

数値の計算には **算術演算子** を使います。
算術演算子は、いわゆる四則演算を行うものです。

★算術演算子

A + B	AとBを加算する (例: 10 + 3 →13)
A - B	AとBを減算する (例: 10 - 3 →7)
A * B	AとBを乗算する (例: 10 * 3 →30)
A / B	AとBを除算する (例: 10 / 3 →3)
A % B	AとBをで除算した剰余を得る (例: 10 % 3 →1)

BMI値を計算するメソッド

以下の点を押さえて、自分でメソッドを書いてみて下さい。

1. フィールドにBMI値を保存するdouble型の変数を用意する。bmi_chiなど分かりやすい名前が良い。
2. メソッドはBMI値を返すので、データ型はdouble型。メソッド名は何でも良いが、模範解答ではbmi()とする。
3. BMI値の計算式は、**体重(kg)÷(身長(m)×身長(m))**。身長の値をmに直すことを忘れずに！
4. 計算結果を用意したBMI値の変数に代入し、値を返す。
5. say()メソッドでbmi()メソッドを呼び出す。

これは練習問題なので、なるべく自分で考えるようにしましょう。
模範解答は、まず自分で考えてから答え合わせのつもりで利用して下さい。

★Step1．模範解答

実行

以下が模範解答の実行結果です。それぞれの身長と体重を元にBMI値を計算して表示されていることがわかります。
一度に多くの処理を書き込まず、このようにエラー無く動くプログラムを少しずつ書いていくのがコツです。

```
C:¥java>cd chapter04

C:¥java¥chapter04>javac PersonTest.java
```

図 9　chapter5 Java で GUI アプリケーションを作る

Javaプログラミング講座

index
chapter1 初めてのJavaプログラミング
chapter2 Javaプログラミングの基礎
chapter3 配列、for文、カプセル化、if文
chapter4 問題：BMI値を計算するプログラム
chapter5 JavaでGUIアプリケーションを作る
- Javaアプリケーションとは？
- AWTでウィンドウを表示
- ウィンドウを閉じる処理
- ウィンドウの設定
- まとめと次回予告

chapter6 画面のレイアウトとイベント処理
chapter7 BMI計算アプリを作ろう
chapter8 問題：数当てゲームを作ろう

chapter5. JavaでGUIアプリケーションを作る

Javaアプリケーションとは？

Javaの実行形態

Javaで作成できるプログラムにはいろいろな実行形態があります。これまで作ってきたものは、全てJavaアプリケーションです。

Javaアプレット
ウェブブラウザ上で実行できるJavaプログラムです。
Javaアプリケーション
ローカルのコンピュータ上で実行できるJavaプログラムです。
Javaサーブレット、JSP(JavaServer Pages)
サーバーサイドのコンピュータで実行できるJavaプログラムです。
デジタル家電、携帯電話
組み込み系や携帯機器などで動くJavaプログラムです。

さらにJavaアプリケーションは次のタイプに分けられます。

CUI(Character User Interface)アプリケーション
コマンドプロンプトから文字ベースで実行するタイプ
GUI(Graphical User Interface)アプリケーション
グラフィカルなウィンドウが表示されるタイプ

GUIアプリケーションについて

前章の終わりでも少し述べましたが、CUIアプリケーションはプログラムがシンプルに作れるのでJavaの入門には最適です。
しかしユーザーからみると操作がし辛いため、あまり実用的ではありません。

そこで今回からは **GUIアプリケーション** を作っていきたいと思います。
GUIアプリケーションなら、テキストエリアやボタンなどグラフィックを多用して、マウスで操作することが出来ます。
当然プログラムは複雑になりますが、見た目はぐっとアプリケーションらしくなります。

JavaでGUIアプリケーションを作るにはいくつか方法があります。

AWT(Abstract Window Toolkit)
GUIを使用したプログラムを開発するためのライブラリー。表示はプラットフォームごとに異なる。
Swing
AWTの上部に作成されたライブラリーで、機能が豊富。表示はすべてのプラットフォームで同じ。
SWT(Standard Widget Toolkit)
IBMがEclipseのために開発したライブラリー。動作が軽快で、表示はプラットフォームごとに異なる。
プラットホームごとにSWT用のライブラリーが必要。

プラットフォームはOSのような意味合いです。AWTはWindowsとMacで見え方が違いますが、SwingはどのOSでも共通のグラフィックをしています。
JavaでのGUIは **AWT** が基本です。Swingを使う場合でもAWTのクラスを利用するので、まずはAWTから学んでいく必要があります。

TOP

AWTでウィンドウを表示

ウィンドウを表示しよう

まずはグラフィックの土台となるウィンドウを作らなければなりません。

「chapter05」フォルダを作り、新規に「Sample.java」というファイルを作ります。
ファイルには以下のように記述します。

Sample.java

C:¥java¥chapter05¥Sample.java(更新) - sakura 1.6.6.0

```
import java.awt.*;
import java.awt.event.*;

class SampleFrame extends Frame {

    //コンストラクタ
    public SampleFrame(String title) {
        //フレームのタイトル
        setTitle(title);

        //ウィンドウを閉じる時
        addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });

    }
}

public class Sample {

    public static void main(String args[]) {
        SampleFrame frm = new SampleFrame("サンプル");
        frm.setLocation(300, 200);      //ウィンドウの表示位置
        frm.setSize(250, 350);          //ウィンドウのサイズ
        frm.setBackground(Color.LIGHT_GRAY);    //ウィンドウの背景色
        frm.setVisible(true);           //ウィンドウを表示する
    }

}
[EOF]
```

実行

さっそくこのプログラムを実行してみましょう。
GUIアプリケーションになっても、実行の仕方はこれまでと変わりません。
実行結果は以下のようになります。

図 10　chapter6 画面のレイアウトとイベント処理

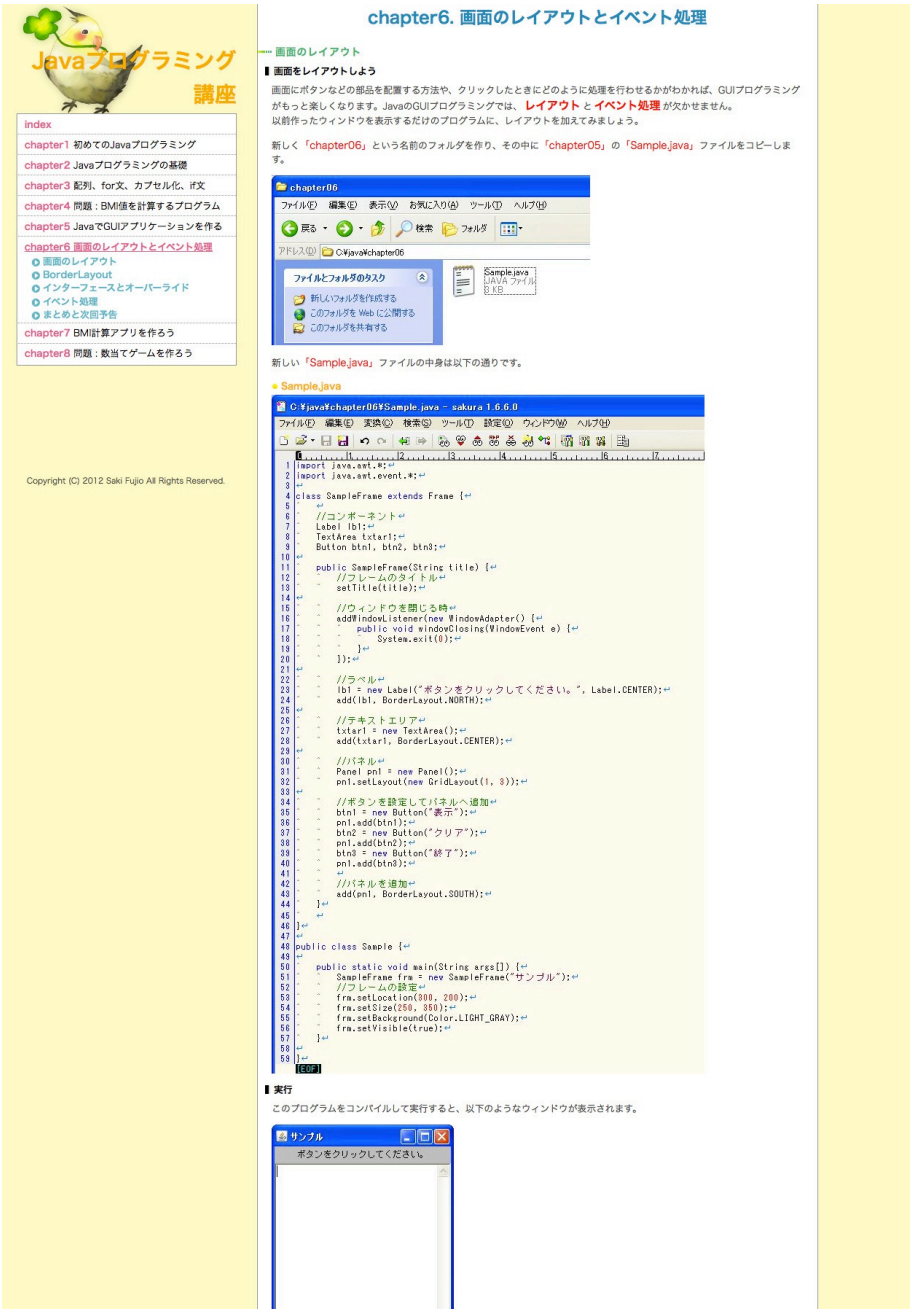
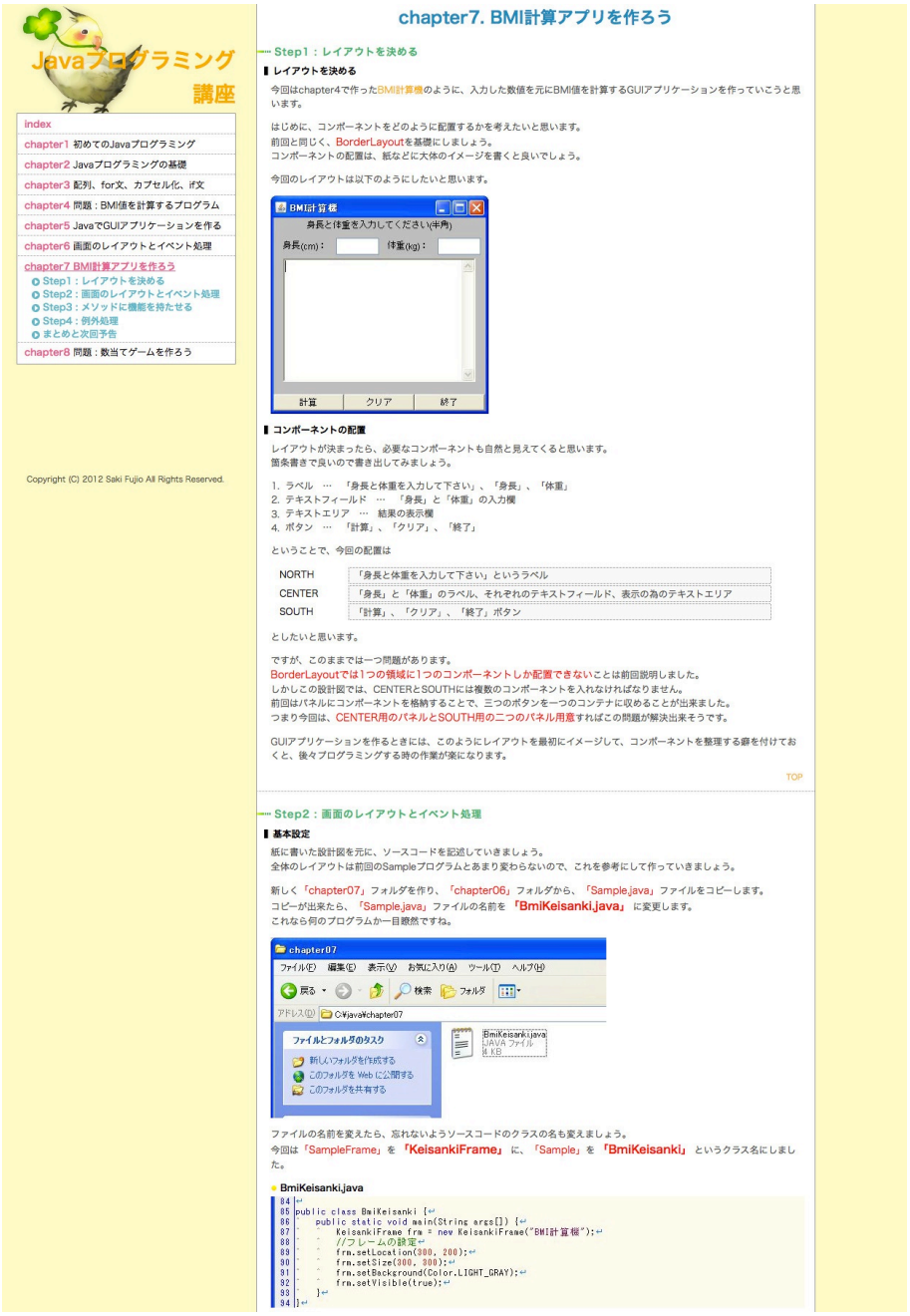


図 11　chapter7 BMI 計算アプリを作ろう



Javaプログラミング講座

index
chapter1 初めてのJavaプログラミング
chapter2 Javaプログラミングの基礎
chapter3 配列、for文、カプセル化、if文
chapter4 問題：BMI値を計算するプログラム
chapter5 JavaでGUIアプリケーションを作る
chapter6 画面のレイアウトとイベント処理
chapter7 BMI計算アプリを作ろう
Step1：レイアウトを決める
Step2：画面のレイアウトとイベント処理
Step3：メソッドに機能を持たせる
Step4：例外処理
まとめと次回予告
chapter8 問題：数当てゲームを作ろう

Copyright (C) 2012 Saki Fujio All Rights Reserved.

chapter7. BMI計算アプリを作ろう

Step1：レイアウトを決める

レイアウトを決める

今回はchapter4で作ったBMI計算機のように、入力した数値を元にBMI値を計算するGUIアプリケーションを作っていこうと思います。

はじめに、コンポーネントをどのように配置するかを考えたいと思います。
前回と同じく、BorderLayoutを基礎にしましょう。
コンポーネントの配置は、紙などに大体のイメージを書くと良いでしょう。

今回のレイアウトは以下のようにしたいと思います。

BMI計算機
身長と体重を入力してください(半角)
身長(cm)：　体重(kg)：
計算　クリア　終了

コンポーネントの配置

レイアウトが決まったら、必要なコンポーネントも自然と見えてくると思います。
箇条書きで良いので書き出してみましょう。

1. ラベル　…　「身長と体重を入力して下さい」、「身長」、「体重」
2. テキストフィールド　…　「身長」と「体重」の入力欄
3. テキストエリア　…　結果の表示欄
4. ボタン　…　「計算」、「クリア」、「終了」

ということで、今回の配置は

NORTH	「身長と体重を入力して下さい」というラベル
CENTER	「身長」と「体重」のラベル、それぞれのテキストフィールド、表示の為のテキストエリア
SOUTH	「計算」、「クリア」、「終了」ボタン

としたいと思います。

ですが、このままでは一つ問題があります。
BorderLayoutでは1つの領域に1つのコンポーネントしか配置できないことは前回説明しました。
しかしこの設計図では、CENTERとSOUTHには複数のコンポーネントを入れなければなりません。
前回はパネルにコンポーネントを格納することで、三つのボタンを一つのコンテナに収めることが出来ました。
つまり今回は、CENTER用のパネルとSOUTH用の二つのパネル用意すればこの問題が解決出来そうです。

GUIアプリケーションを作るときには、このようにレイアウトを最初にイメージして、コンポーネントを整理する癖を付けておくと、後々プログラミングする時の作業が楽になります。

TOP

Step2：画面のレイアウトとイベント処理

基本設定

紙に書いた設計図を元に、ソースコードを記述していきましょう。
全体のレイアウトは前回のSampleプログラムとあまり変わらないので、これを参考にして作っていきましょう。

新しく「chapter07」フォルダを作り、「chapter06」フォルダから、「Sample.java」ファイルをコピーします。
コピーが出来たら、「Sample.java」ファイルの名前を 「BmiKeisanki.java」 に変更します。
これなら何のプログラムか一目瞭然ですね。

chapter07
ファイル(F)　編集(E)　表示(V)　お気に入り(A)　ツール(T)　ヘルプ(H)
戻る　検索　フォルダ
アドレス(D)　C:¥java¥chapter07
ファイルとフォルダのタスク
新しいフォルダを作成する
このフォルダを Web に公開する
このフォルダを共有する
BmiKeisanki.java
JAVA ファイル
4 KB

ファイルの名前を変えたら、忘れないようソースコードのクラスの名も変えましょう。
今回は「SampleFrame」を 「KeisankiFrame」 に、「Sample」を 「BmiKeisanki」 というクラス名にしました。

BmiKeisanki.java

```
84 
85 public class BmiKeisanki {
86     public static void main(String args[]) {
87         KeisankiFrame frm = new KeisankiFrame("BMI計算機");
88         //フレームの設定
89         frm.setLocation(300, 200);
90         frm.setSize(300, 300);
91         frm.setBackground(Color.LIGHT_GRAY);
92         frm.setVisible(true);
93     }
94 }
```

図 12　chapter8 問題：数当てゲームを作ろう

図 13-1　途中段階の模範解答を実行した例

これは練習問題なので、なるべく自分で考えるようにしましょう。
模範解答は、まず自分で考えてから答え合わせのつもりで利用して下さい。

＊Step1. 模範解答

▌実行

以下が模範解答の実行結果です。それぞれの身長と体重を元にBMI値を計算して表示されていることがわかります。
一度に多くの処理を書き込まず、このようにエラー無く動くプログラムを少しずつ書いていくのがコツです。

```
C:¥java>cd chapter04

C:¥java¥chapter04>javac PersonTest.java

C:¥java¥chapter04>java PersonTest
名前はのび太です。身長は140.0cmで、体重は34.0kgです。
BMI値は17.346938775510207です。

名前はしずかです。身長は141.0cmで、体重は35.0kgです。
BMI値は17.6047482521です。

名前はジャイアンです。身長は155.0cmで、体重は60.0kgです。
BMI値は24.97398543184183です。

名前はスネ夫です。身長は135.0cmで、体重は30.0kgです。
BMI値は16.46090534979424です。


C:¥java¥chapter04>_
```

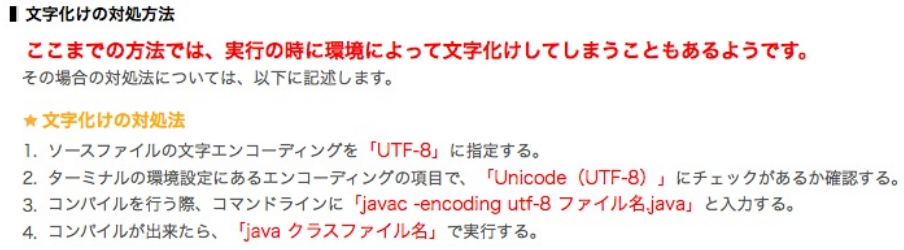
図 13-2　完成した模範解答の実行例

```
C:¥java¥chapter04>java PersonTest
名前はのび太です。身長は140.0cmで、体重は34.0kgです。
BMI値は17.35で、判定は痩せすぎです。

名前はしずかです。身長は141.0cmで、体重は35.0kgです。
BMI値は17.6で、判定は痩せすぎです。

名前はジャイアンです。身長は155.0cmで、体重は60.0kgです。
BMI値は24.97で、判定は適正です。

名前はスネ夫です。身長は135.0cmで、体重は30.0kgです。
BMI値は16.46で、判定は痩せすぎです。
```

図 14　リンク色の変更


Javaプログラミング講座
index
chapter1 初めてのJavaプログラミング
chapter2 Javaプログラミングの基礎
chapter3 配列、for文、カプセル化、if文
chapter4 問題：BMI値を計算するプログラム
chapter5 JavaでGUIアプリケーションを作る
chapter6 画面のレイアウトとイベント処理
chapter7 BMI計算アプリを作ろう
chapter8 問題：数当てゲームを作ろう
Copyright (C) 2012 Saki Fujio All Rights Reserved.
Javaプログラミング講座ついて
この講座では、Javaプログラミングの基礎から、AWTを使用したGUIアプリケーション作成までを学ぶ事が出来ます。
これからJavaを始める人の為に、コマンドプロンプトの設定やプログラムの実行の仕方を始め、クラスやメソッド、配列なども、サンプルプログラムを交えながら画像つきで詳しく解説しています。
この講座は、PythonやPHPなど他のプログラミング言語を学んだことがある人が、次のステップとしてJavaの基礎を学習することを想定して作られています。
初心者にも分かりやすく説明しているつもりですが、全くのプログラミング初心者ではところどころ難しいこともあるかもしれません。
そういった方は、Javaの参考書か、Javaの入門サイトなどと合わせて読みながら勉強して下さい。
TOP
Java開発環境の準備
Javaでアプリケーションを開発するには、JDK というJava言語でプログラミングを行う際に必要な最低限のソフトウェアのセットをインストールする必要があります。これはJavaの開発元であるSun Microsystems社が開発、配布しています。
この講座では、JDKのインストール方法を紹介していないので、以下のサイトを参考にして事前に開発環境を整えてからchapterの内容に進んで下さい。また、chapter1でもこれと同じ注意書きをしています。
＊JavaDrive - JDK 6のインストールと環境設定
MacOSXの場合は、Java開発環境が元々備わっているので、この作業は必要ありません。
TOP
参考リンク
JavaTM 2 Platform Standard Edition 5.0 API 仕様
Java初心者入門講座
JavaDrive
Java AWT 入門
とほほのJava入門
TOP
参考文献
堀江 幸生(2007) 『世界でいちばん簡単なJavaのe本―Javaの基本と考え方がわかる本』 秀和システム
鳥海 不二夫(2007) 『よくわかるJava』 秀和システム
掌田 津耶乃(2010) 『Eclipse 3.6ではじめるJavaプログラミング入門―Eclipse 3.6 Helios対応』 秀和システム
TOP
↓
Javaプログラミング講座
index
chapter1 初めてのJavaプログラミング
chapter2 Javaプログラミングの基礎
chapter3 配列、for文、カプセル化、if文
chapter4 問題：BMI値を計算するプログラム
chapter5 JavaでGUIアプリケーションを作る
chapter6 画面のレイアウトとイベント処理
chapter7 BMI計算アプリを作ろう
chapter8 問題：数当てゲームを作ろう
Copyright (C) 2012 Saki Fujio All Rights Reserved.
Javaプログラミング講座ついて
この講座では、Javaプログラミングの基礎から、AWTを使用したGUIアプリケーション作成までを学ぶ事が出来ます。
これからJavaを始める人の為に、コマンドプロンプトの設定やプログラムの実行の仕方を始め、クラスやメソッド、配列なども、サンプルプログラムを交えながら画像つきで詳しく解説しています。
この講座は、PythonやPHPなど他のプログラミング言語を学んだことがある人が、次のステップとしてJavaの基礎を学習することを想定して作られています。
初心者にも分かりやすく説明しているつもりですが、全くのプログラミング初心者ではところどころ難しいこともあるかもしれません。
そういった方は、Javaの参考書か、Javaの入門サイトなどと合わせて読みながら勉強して下さい。
TOP
Java開発環境の準備
Javaでアプリケーションを開発するには、JDK というJava言語でプログラミングを行う際に必要な最低限のソフトウェアのセットをインストールする必要があります。これはJavaの開発元であるSun Microsystems社が開発、配布しています。
この講座では、JDKのインストール方法を紹介していないので、以下のサイトを参考にして事前に開発環境を整えてからchapterの内容に進んで下さい。また、chapter1でもこれと同じ注意書きをしています。
＊JavaDrive - JDK 6のインストールと環境設定
MacOSXの場合は、Java開発環境が元々備わっているので、この作業は必要ありません。
TOP
参考リンク
JavaTM 2 Platform Standard Edition 5.0 API 仕様
Java初心者入門講座
JavaDrive
Java AWT 入門
とほほのJava入門
TOP
参考文献
堀江 幸生(2007) 『世界でいちばん簡単なJavaのe本―Javaの基本と考え方がわかる本』 秀和システム
鳥海 不二夫(2007) 『よくわかるJava』 秀和システム
掌田 津耶乃(2010) 『Eclipse 3.6ではじめるJavaプログラミング入門―Eclipse 3.6 Helios対応』 秀和システム
TOP

図 15　文字化けの対処法



▌文字化けの対処方法

ここまでの方法では、実行の時に環境によって文字化けしてしまうこともあるようです。

その場合の対処法については、以下に記述します。

★文字化けの対処法

1. ソースファイルの文字エンコーディングを「UTF-8」に指定する。
2. ターミナルの環境設定にあるエンコーディングの項目で、「Unicode（UTF-8）」にチェックがあるか確認する。
3. コンパイルを行う際、コマンドラインに「javac -encoding utf-8 ファイル名.java」と入力する。
4. コンパイルが出来たら、「java クラスファイル名」で実行する。

「PersonTest.java」

```
01:import java.text.DecimalFormat;
02:import java.io.*;  //システムの入出力を規定したパッケージ
03:
04:class Person {
05:  //変数の宣言
06:  private double shincho, taiju, bmi_chi;
07:
08:  //コンストラクタ
09:  public Person(double _shincho, double _taiju) {
10:    shincho = _shincho;
11:    taiju = _taiju;
12:  }
13:
14:  //say() メソッド
15:  public void say() {
16:    DecimalFormat df = new DecimalFormat("###.##");
  //BMI 値を小数点以下 2 桁に指定
17:    System.out.println("あなたの BMI 値は"
 + df.format(bmi()) + "です。");
18:    System.out.println("あなたの BMI 判定は"
 + bmi_hantei() + "です。");
19: }
20:
```

```
21:  //BMI 値を計算するメソッド
22:  public double bmi() {
23:    shincho = shincho / 100;
24:    bmi_chi = taiju / ( shincho * shincho );
25:    return bmi_chi;
26:  }
27:
28:  //計算で得た値を判定するメソッド
29:  public String bmi_hantei() {
30:    String kekka;
31:    if(bmi_chi < 18.5)
32:      kekka = "痩せすぎ";
33:    else if(bmi_chi < 25)
34:      kekka = "適正";
35:    else
36:      kekka = "太りすぎ";
37:    return kekka;
38:  }
39:}
40:
41:public class PersonTest {
42:  public static void main(String[] args) throws
 IOException {
43:
44:    //BufferedReader クラスのインスタンス化
```

```
45:    BufferedReader br = new BufferedReader
(new InputStreamReader(System.in));
46:
47:    System.out.println("BMI 値を計算します。");
48:
49:    //身長の値を尋ねる
50:    System.out.println("身長 (cm) を入力してください。");
51:    //入力された文字列を double 型に直して変数 height に代入
52:    double height = Double.parseDouble(br.readLine());
53:
54:    //身長と同じ事を体重でも行う
55:    System.out.println("体重 (kg) を入力してください。");
56:    double weight = Double.parseDouble(br.readLine());
57:
58:    //Person クラスのインスタンス化
59:    Person person = new Person(height, weight);
60:
61:    person.say(); //say() メソッドを呼び出して結果を表示
62:  }
63:}
```

「KazuateGame.java」

```
001:import java.awt.*;
002:import java.awt.event.*;
003:import java.util.Random;
004:
005:class KazuateFrame extends Frame implements
 ActionListener {
006:   //コンポーネント
007:   Label lb;
008:   TextField tf;
009:   TextArea txtar;
010:   Button ok, btn1, btn2, btn3;
011:
012:   //変数の宣言
013:   int ran;
014:
015:   public KazuateFrame(String title) {
016:     //フレームのタイトル
017:     setTitle(title);
018:
019:     //ウィンドウを閉じる時
020:     addWindowListener(new WindowAdapter() {
021:       public void windowClosing(WindowEvent e) {
022:         System.exit(0);
```

```
023:       }
024:     });
025:
026:     //ラベル
027:     lb = new Label("1〜100 の数字を入力して下さい。(半
角)", Label.CENTER);
028:      add(lb, BorderLayout.NORTH);
029:
030:     //入力欄、OK ボタン、表示エリアの作成
031:     tf = new TextField(2);
032:     ok = new Button("OK");
033:     txtar = new TextArea(7, 30);
034:     tf.addActionListener(this);  //アクションリスナー
の設定
035:     ok.addActionListener(this);
036:
037:     //パネルを作成してコンポーネントを配置
038:     Panel pn1 = new Panel();
039:     pn1.add(tf);
040:     pn1.add(ok);
041:     pn1.add(txtar);
042:
043:     //ボタンを作成
044:     btn1 = new Button("答えを見る");
045:     btn2 = new Button("リセット");
```

```
046:    btn3 = new Button("ゲームを終了");
047:
048:    //アクションリスナーの設定
049:    btn1.addActionListener(this);
050:    btn2.addActionListener(this);
051:    btn3.addActionListener(this);
052:
053:    //パネルを作成してボタンを配置
054:    Panel pn2 = new Panel();
055:    pn2.add(btn1);
056:    pn2.add(btn2);
057:    pn2.add(btn3);
058:
059:    //パネルを追加
060:    add(pn1, BorderLayout.CENTER);
061:    add(pn2, BorderLayout.SOUTH);
062:  }
063:
064:  public void actionPerformed (ActionEvent e) {
065:    if (e.getSource() == tf || e.getSource() == ok) {
066:      try {
067:        check();
068:      } catch (NumberFormatException ex) {
069:        StringBuffer temp  = new StringBuffer();
070:        temp.append("入力に間違いがあります。" + "\n");
```

```
071:        temp.append("1〜100 の半角数字で入力して下さ
い。");
072:        txtar.setText(temp.toString());
073:      }
074:    } else if(e.getSource() == btn1) {
075:      answer();
076:    } else if(e.getSource() == btn2) {
077:      tf.setText("");
078:      txtar.setText("");
079:      random();
080:    } else {
081:      System.exit(0);
082:    }
083:      tf.requestFocusInWindow();
084:      tf.selectAll();
085:  }
086:
087:  public void random() {
088:    Random rnd = new Random();  //Random クラスのイン
スタンスを作成
089:    ran = rnd.nextInt(100) + 1;  //１を足す事０〜９９
が１〜１００になる
090:  }
091:
092:  private void check() {
```

```
093:    int num = Integer.parseInt(tf.getText());
094:    StringBuffer temp  = new StringBuffer();
  //StringBuffer クラスのインスタンスの作成
095:    if (num < 1 || num > 100) {
096:      temp.append("答えは 1〜100 の半角数字です。"
 + "\n");
097:      temp.append("答えを正しく入力して下さい。");
098:    } else if (num > ran) {
099:      temp.append("答えは" + num + "よりも小さい数字で
す。");
100:    } else if (num < ran) {
101:      temp.append("答えは" + num + "よりも大きい数字で
す。");
102:    } else {
103:      temp.append("正解です！おめでとう！" + "\n");
104:      temp.append("答えは" + ran + "でした。"+ "\n");
105:      temp.append("\n");
106:      temp.append("答えがリセットされました。");
107:      random();
108:    }
109:    txtar.setText(temp.toString());  //テキストエリア
に文字列を表示
110:  }
111:
112:  private void answer() {
```

```
113:    StringBuffer temp  = new StringBuffer();
114:    temp.append("答えは" + ran + "でした。" + "\n");
115:    temp.append("\n");
116:    temp.append("次は頑張って正解しよう！");
117:    txtar.setText(temp.toString());
118:    random(); //乱数の再設定
119:  }
120:}
121:
122:public class KazuateGame {
123:  public static void main(String args[]) {
124:    KazuateFrame frm = new KazuateFrame("数当てゲーム");
125:    //フレームの設定
126:    frm.setLocation(300, 200);
127:    frm.setSize(320, 250);
128:    frm.setBackground(Color.LIGHT_GRAY);
129:    frm.setVisible(true);
130:
131:    //乱数の設定
132:    frm.random();
133:  }
134:}
```