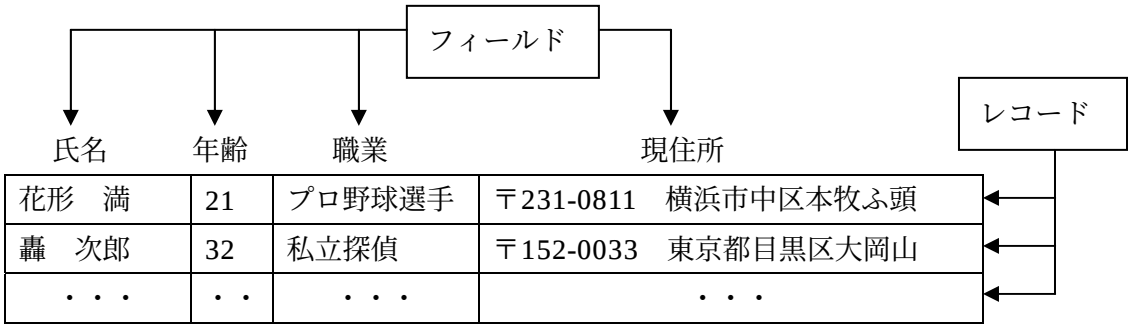
第５章．レコード構造を使った処理―クラスの利用

【学習のねらい】

① データ構造としてのレコード（構造）を Java 言語で扱う方法を学習する。実は、これは第４章で学習したクラスの利用に他なりません。

② レコード構造を応用したプログラムを学習する。

今、次のような、名簿があるとします。



花形　満	21	プロ野球選手	〒231-0811　横浜市中区本牧ふ頭
轟　次郎	32	私立探偵	〒152-0033　東京都目黒区大岡山
・・・	・・	・・・	・・・

名簿は、いわば個人情報が集まったデータです。この場合、一人分の情報、つまり各行のデータを一まとまりにして**レコード**と呼びます。例えば、上の例の 1 行目は、花形満さんに関するレコードであるという訳です。そして、各レコードの項目（氏名、年齢、職業、現住所）を**フィールド**と言います。そして、関連するレコードを一まとまりにしたものを**ファイル**と呼びます。たとえば上は名簿ファイルの例であるという訳です。

関連する一まとまりのデータをプログラムで処理する場合、各項目を別々に（独立な変数として）扱うよりもレコードとして定義し一括して取り扱うほうが便利な場合が多いものです。そのために、多くのプログラミング言語では、データ構造の一つとしてレコードを定義することができるようになっています。Java 言語では、それを、複数のフィールドを持つ**クラス**として定義することができます。本章では 3-5 節で学習したプログラムを例に採り上げ、クラスで定義したレコードを応用する学習をしましょう。

５－１　レコードの定義

3-5 節で学習した【基礎課題 3-7】では、右のような、ある科目のテスト得点を記録したファイル（score2.txt）を扱いました。上で説明した用語を用いれば、これは、氏名と得点という**フィールド**（項目）から構成されるレコードの集まり、つまりファイルであるということができます。そこで、まずはこのファイルを用いて、レコードを定義しそれを用いる練習をしてみましょう。

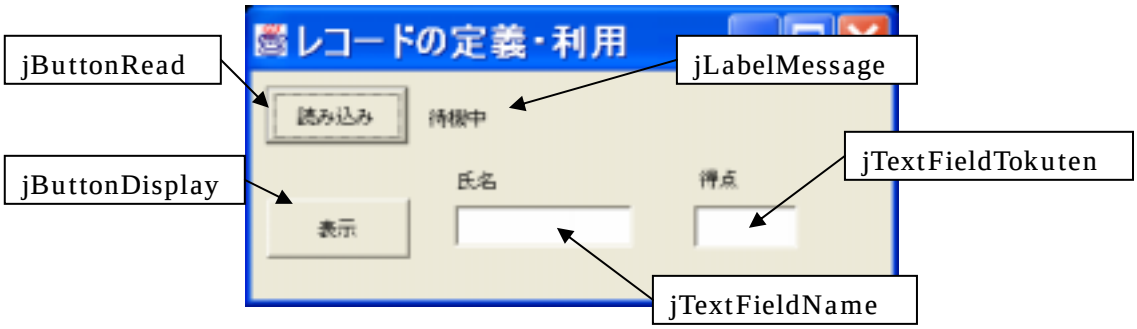
<score2.txt>

```
花形　満　,55
轟　次郎　,63
金田正太郎,39
鮎原こずえ,87
浅丘ユミ　,48
矢吹　丈　,70
東　八郎　,35
星　明子　,77
諸星　団　,59
伊達直人　,44
```

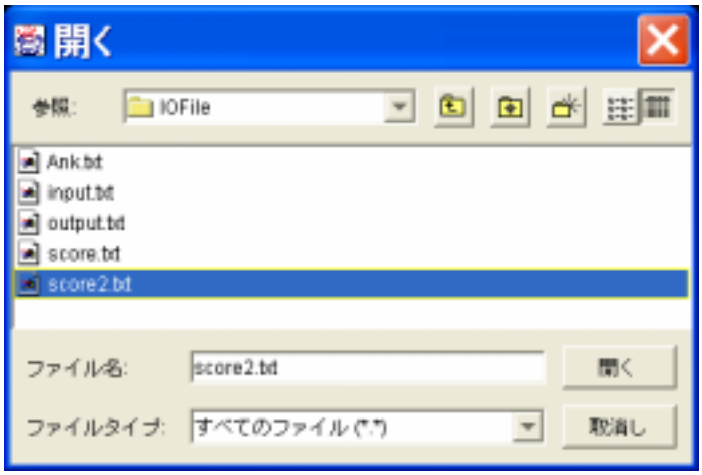
【基礎課題 5-1】

作成するプログラムの動作内容は次の通りです（主要部分は【基礎課題 3-7】と同じです）。

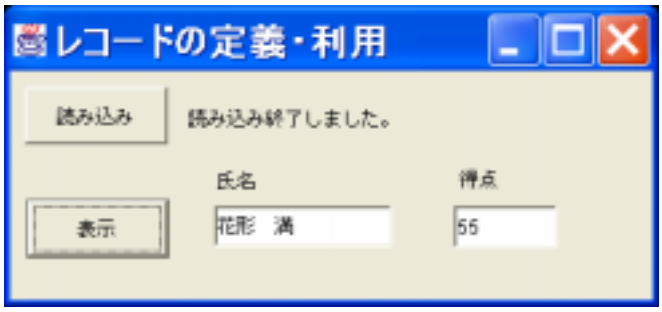
まず、プログラムを起動すると次の画面が現れます。



ここで、［読み込み］ボタンをクリックすると・・・



入力ファイル指定のダイアログボックスが現れるので、ここで、入力ファイル「score2.txt」を指定します。



入力ファイルを指定するとデータの読み込みが完了します。その後、［表示］ボタンをクリックすると、ファイルに保管されている最初の（1 番目の）受験生の氏名と得点が表示されます。

つまり、ここでは簡単のために、ファイルの先頭行（のみ）を読み込んで、そのデータを表示させるプログラムを考えます。ファイル中の全データを読み込む課題は次節で扱います。

それでは、プログラムの作成にとりかかりましょう。まず新規アプリケーションを作成し、上のようなフレームを作成して下さい（後の説明の都合上、プロジェクト名を「Record」として下さい）。以下の手順に従ってプログラムを作成して下さい。

＜プログラム作成の手順＞

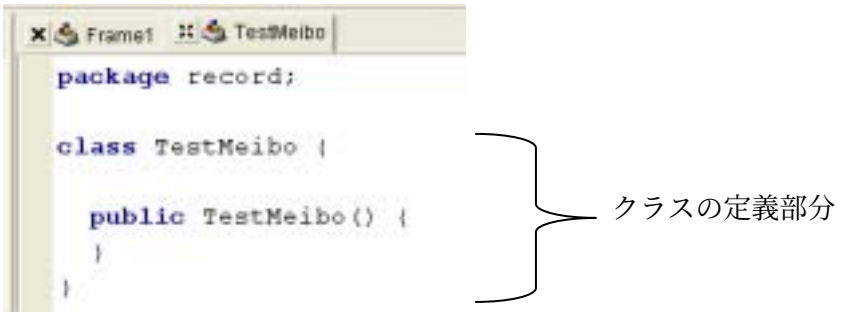
１．クラス「TestMeibo」の定義

氏名と得点をフィールドとして持つクラスを定義しましょう。第４章【基礎課題 4-1】で行ったのと同じ要領で新しいクラス「TestMeibo」定義して下さい。クラス設計ウィザードで指定するのは、以下の項目です。

クラス名：TestMeibo

ベースクラス名：java.lang.Object

クラス設計ウィザードの設定後、次の通り、クラスの編集画面「TestMeibo.java」が現れます。



ここで、クラスの定義部分を次のように記述して下さい。

```
class TestMeibo {
  private String Name; //氏名フィールドの定義（宣言）
  private int Tokuten; //得点フィールドの定義（宣言）
  public TestMeibo(String Shimei, int Ten) {
    Name=Shimei;
    Tokuten=Ten;
  }

  public String getName() {
    return Name;
  }

  public int getTokuten() {
    return Tokuten;
  }
}
```

コンストラクタの定義

氏名フィールドの値を返すメソッドの定義

得点フィールドの値を返すメソッドの定義

プログラムを見れば意味は分かると思います。ただ、フィールドだけではなく、フィールドの値を返すメソッド getName()および getTokuten()を定義している点が気になったかもしれません。これは、**private** で宣言したフィールド変数の値を外部クラスである Frame1.java から表示させる際に必要なため定義したものです。今後、他のクラスからの参照が必要なフィールドについては、適宜、その値を返すメソッドを定義します。

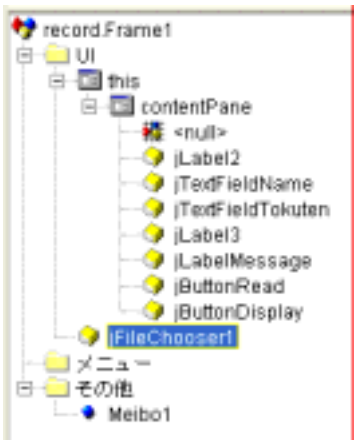
2．クラス「Frame1.java」の import 文の追加

Frame1.java のソース編集画面に戻って下さい。ここで、【基礎課題 3-7】と同様に、プログラムの冒頭部分に次の波線部のインポート文を追加します。

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;
import java.util.*;
```

3．［読み込み］ボタンクリック時のイベントハンドラの記述

【基礎課題 3-7】と同様、入力ファイルの指定にはファイルダイアログを用います。したがって、jFileChooser コンポーネントを次のように UI フォルダに追加しておくのを忘れないで下さい（やり方は p.39 参照）。



TestMeibo クラスのオブジェクトの宣言および、イベントハンドラを次のように記述します。p.76 で説明したとおり、ここでは、ファイルの最初の 1 行のみを読み込みます。【基礎課題 3-7】と大きく異なるのは、枠線で囲まれた部分です。

```
                    ①
TestMeibo Meibo1;  //TestMeibo クラスのオブジェクトの宣言
void jButtonRead_actionPerformed(ActionEvent e) {
  String Data;
  try {
    jFileChooser1.showOpenDialog(this);
    File FName=jFileChooser1.getSelectedFile();
    BufferedReader fin=new BufferedReader
                    (new FileReader(FName));
    Data=fin.readLine();
    StringTokenizer tk=new StringTokenizer(Data,",");   ②
    Meibo1= new TestMeibo(tk.nextToken(),
                      Integer.parseInt(tk.nextToken()));
    jLabelMessage.setText("読み込み終了しました。");
    fin.close();
  }
  catch (Exception em) {
    jLabelMessage.setText("エラー発生："+em);
  }
}
```

①で TestMeibo クラスのオブジェクトを宣言し、②によって、読み込んだ氏名と得点をフィールドとして持つ、オブジェクト Meibo1 が生成されています。

４．［表示］ボタンクリック時のイベントハンドラの記述

プログラムは次の通りとなります。空欄を埋めてプログラムを完成させて下さい。

```
void jButtonDisplay_actionPerformed(ActionEvent e) {
   jTextFieldName.setText(Meibo1.getName());
   jTextFieldTokuten.setText(
         String.valueOf(Meibo1.                ) );
}
```

TestMeibo クラスの定義において、もし、フィールド変数が private で宣言されていなければ、Meibo1.Name および Meibo1.Tokuten の形でその値（氏名と得点）を参照できるのですが、今は（private で宣言されているので）参照できません。そこで、当該フィールドの値を返すメソッドを用いて氏名および得点を表示させています。この手法は今後もよく用います。

プログラムを作成したら実行し、動作を確認して下さい。

＜用語の整理　レコードとクラス＞

冒頭に説明した通り、レコードとは、幾つかのフィールドを一まとめにしたデータ構造です。これは、整数型や文字列型などの単独の変数、そして（複数の要素を持った）配列の次に出てくる、データ構造の一般概念です。プログラマが自在にレコードを定義できることは実用上重要なことなので、多くのプログラミング言語に、その（定義）機能が備わっています。例えば PASCAL という言語では、そのままズバリ record 文という定義文が用意されています。一方、Java 言語には record そのものの定義文は存在しません。その代わり、クラスの定義を行えるようになっています。既に学習した通り、クラスはフィールドのみならず、メソッドの定義も含まれます。つまりクラスはレコードを含む概念なので、これを用いてレコード構造を扱えるわけです（”クラス＝レコード+メソッド”と捉えても良いでしょう）。ですから、Java 言語のテキストには、（クラスを知っていれば十分なので）わざわざレコードについて説明していないものもあります。本節では、レコード構造を表すためのクラスという意味で、レコードという用語を用いてきましたが、実態がクラスなので、以下ではクラスあるいは、それから生成されたオブジェクトという用語を用いることにします。

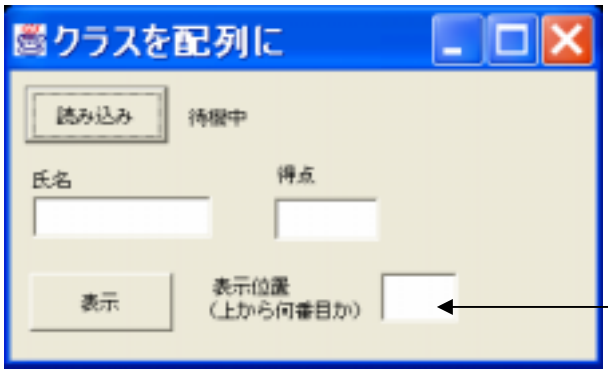
５－２．オブジェクトを配列に

前節の様に定義したクラス（から生成されたオブジェクト）を配列にする事もできます。要領は、第 2 章で学習した通りですが、それを次の課題で確認しておきましょう。

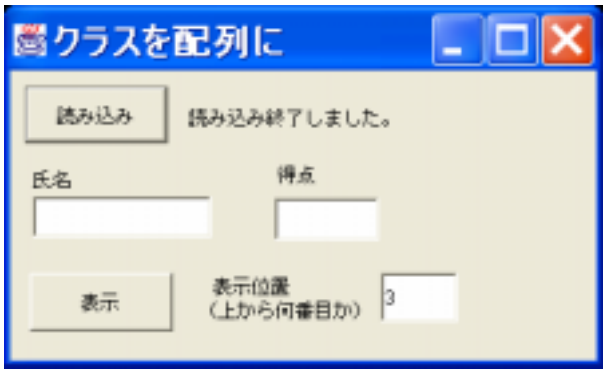
【基礎課題 5-2】

【基礎課題 5-1】のプログラムを拡張して、今度はファイルから全員のデータを読み込み、指定した位置（上から何番目か）の受験生の氏名と得点を表示する様にしましょう。作成に当たっては、【基礎課題 5-1】のプログラムを開き、それに修正を施す形で行って結構です。

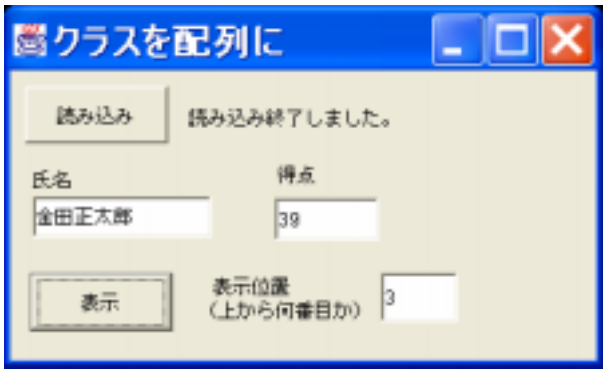
作成するプログラムの動作内容は次の通りです。



まず、プログラムを起動すると左の画面が現れます。［読み込み］ボタンをクリックして入力ファイル「score2.txt」を指定するところまでは【基礎課題 5-1】と同様です。



入力ファイル指定後、表示位置欄に、表示したい位置（上から何番目か）を入力し、［表示］ボタンをクリックします。



すると、指定した位置（順番）の受験生の氏名と得点が表示されます。

それでは、プログラムの作成に取り掛かりましょう。まず【基礎課題 5-1】のフレームを上のように修正して下さい（レイアウトを変更し、表示位置欄を追加して下さい）。

次に［読み込み］ボタンのクリック時とのイベントハンドラおよび TestMeibo クラスのオブジェクト Meibo の宣言部を以下の通り記述して下さい。

＜［読み込み］ボタンクリック時のイベントハンドラ＞

```
TestMeibo Meibo1[]=new TestMeibo[100];      ①
int Num; //データの個数

void jButtonRead_actionPerformed(ActionEvent e) {
  String Data;
  try {
    jFileChooser1.showOpenDialog(this);
    File FName=jFileChooser1.getSelectedFile();
    BufferedReader fin=new BufferedReader
                    (new FileReader(FName));
    int i=0;
    while((Data=fin.readLine())!=null) {
      StringTokenizer tk=new StringTokenizer(Data,",");
      Meibo1[i]= new TestMeibo(tk.nextToken(),
                       Integer.parseInt(tk.nextToken()));
      i++;
    }
    Num=i;
    jLabelMessage.setText("読み込み終了しました。");
    fin.close();
  }
  catch (Exception em) {
    jLabelMessage.setText("エラー発生："+em);
  }
}
```

枠線部は、ファイルから全データを読み込むようにした事に伴う変更で、基本的に【基礎課題 3-7】と同様です。そして下線部は、（クラス TestMeibo から生成された）オブジェクト Meibo1 を配列（要素数 100）にすることに伴う変更です。それ以外は【基礎課題 5-1】と同じなので内容は理解できると思います。上の①より、通常の変数と同様の形式で、クラスから配列構造のオブジェクトを生成することができることが分かると思います。

これで、（0 番目から始めて）i 番目の受験生のデータは、オブジェクト Meibo1[i]に保管されることになります。

［表示］ボタンクリック時のイベントハンドラは次ページの通りです。空欄を埋めてプログラムを完成させて下さい。

＜［表示］ボタンクリック時のイベントハンドラ＞

```
void jButtonDisplay_actionPerformed(ActionEvent e) {
  int No=Integer.parseInt(jTextFieldNo.getText());
  jTextFieldName.setText(Meibo1[No-1].getName());
  jTextFieldTokuten.setText(String.valueOf
                    (                    .getTokuten()));
}
```

上の波線部が、Meibo1 オブジェクトを配列に拡張した事に伴う変更です。添え字が（No-1）と、１を引いているのは、上から何番目かを数える際、通常通り、一番上を**1**番目と数えることを想定しているためです。もし、配列の添え字と同様、0 番目から数える約束にすれば、そのまま（No）となります。

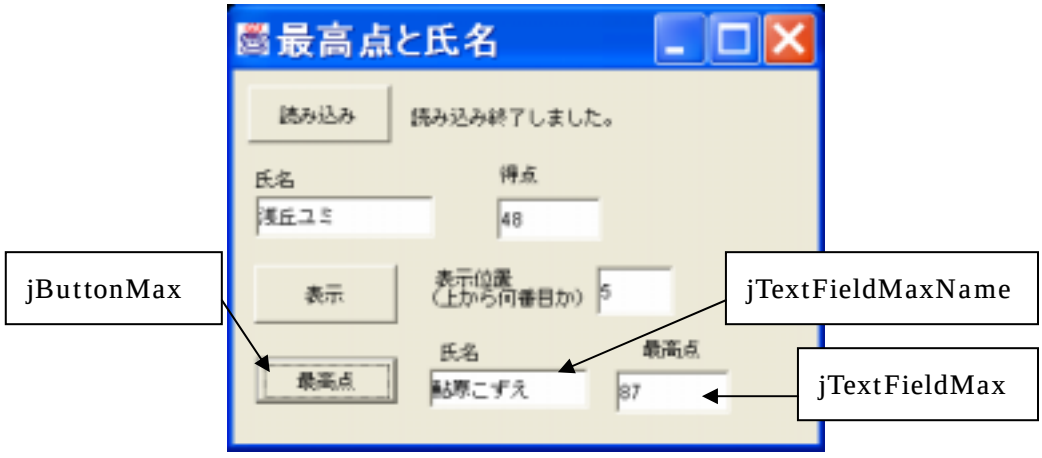
作成したら実行し動作を確認してください。

５－３．　オブジェクトを用いるメリット

前節までの例で、オブジェクトを利用してレコード構造を表現する方法は分かったと思います。しかし、【基礎課題 5-2】まででは、わざわざオブジェクト（で表現したレコード構造）を利用するメリットを感じなかったかもしれません。実はオブジェクトを用いるメリットは、複数の項目（フィールド）を一括して処理できる点にあります。その点に注目しながら、次の課題に進みましょう。

【基礎課題 5-3】

【基礎課題 5-2】に以下の様に、［最高点］ボタンを加え、これをクリックすると、最高点をとった人の氏名とその得点を表示するプログラムに拡張しましょう（実はこれは、【応用課題 3-D】と同じ処理内容です）。作成に当たっては、【基礎課題 5-2】のプログラムに直接修正を施して結構です。



［最高点］ボタンクリック時のイベントハンドラは次のように書けます。

```
void jButtonMax_actionPerformed(ActionEvent e) {
  int Max=Meibo1[0].getTokuten(); //最高点保管用変数
  String Name=Meibo1[0].getName(); //最高点をとった人の氏名保管用変数
  for (int i=1;i<Num;i++) {
    if(Meibo1[i].getTokuten()>Max) {
      Max=Meibo1[i].getTokuten(); //最高点候補の保管
      Name=Meibo1[i].getName();   //最高点候補者氏名の保管
    }
  }
  jTextFieldMaxName.setText(Name);
  jTextFieldMax.setText(String.valueOf(Max));
}
```

ここで、最高点とそれを取った人の氏名保管のために、変数 Max および Name をそれぞれ用意しています。今、これを一本化して、（最高点関連レコードを保管する）オブジェクトに置き換えてみたらどうでしょうか？具体的には次のようなプログラムになります。

＜オブジェクトを用いた場合＞

```
void jButtonMax_actionPerformed(ActionEvent e) {
  TestMeibo Max; //最高点レコードを保管するオブジェクトの宣言
  Max=Meibo1[0]; //初期値の代入
  for (int i=1;i<Num;i++) {
    if(Meibo1[i].getTokuten()>Max.getTokuten()) {
      Max=Meibo1[i];  //最高点候補オブジェクトの保管
    }
  }
  jTextFieldMaxName.setText(Max.getName());
  jTextFieldMax.setText(String.valueOf(Max.getTokuten()));
}
```

今度は、最高点レコード保管用変数として、オブジェクト Max を一つ宣言するだけで済みました。また、オブジェクトを用いたおかげで、最高点候補の保管が

```
Max=Meibo1[i];
```

の 1 行だけですみました。

今の例ではフィールド数が 2 つだけなので、（上の 2 つのプログラムで）それほど大きな違いはありませんが、一般のプログラムでは、フィールドの数が大きなオブジェクトを扱うのが普通です。そのような場合でも、（当該フィールドが定義されたクラスから生成された）オブジェクトを用いれば、各フィールドの代入に対応する処理が、一つのオブジェクトの代入で実現します（下の補足参照）。これが、オブジェクトを用いる大きなメリットです。

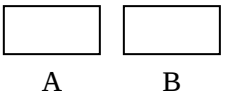
補足　オブジェクトの代入の意味

上では、オブジェクトの”代入”という言葉を使ってきました。しかし、実はオブジェクトの各フィールドやメソッドが代入されているわけではありません。オブジェクトの代入では、（通常の変数と異なり）コンピュータのメモリ上にそのオブジェクトが保管されている”**場所**”が代入されるのです。大事な点ですので少し順を追って説明しましょう。

まず、「変数やオブジェクトは全てコンピュータのメモリ上に保管されている」という点を了解して下さい。例えば次のように①整数型変数 A、B を宣言し、②それに３を代入した後、③代入文「B=A」を実行した場合のメモリ上の動作は次のようになります。

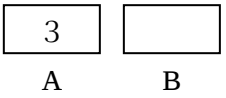
＜メモリ上の動作＞

① `int A,B;` →



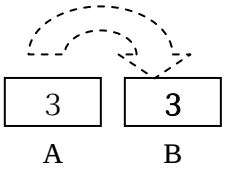
２つの保管領域を確保し、各々を A および B と命名

② `A=3;` →



保管領域 A に整数値３を代入

③ `B=A;` →



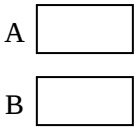
保管領域 A 内の値（３）を記憶領域 B 内にコピーする。

言うまでもなく保管領域 A とは変数 A に他なりません。

では、MyClass というクラスのオブジェクト A、B を宣言し、上と同様の処理を実行した場合、どういう動作が起こるのかをみてみましょう。

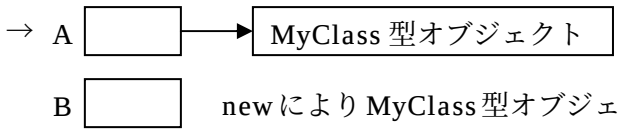
＜メモリ上の動作＞

④ `MyClass A,B;` →



２つの保管領域を確保し、その保管領域を A および B と命名

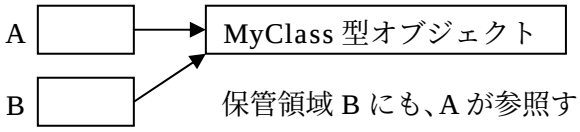
⑤ `A=new MyClass();` →



new により MyClass 型オブジェクトをメモリ上（の別な場所）に生成し、その”**保管場所**”を A に代入する。オブジェクトの中身が保管されるのではない事に注意。

※　オブジェクトの保管場所の情報が代入されると、当該オブジェクトを参照する事が可能になります。この状態を、上の様に参照先を→で指し示す、という表記で表現します。

⑥　B=A;　→



保管領域 B にも、A が参照するオブジェクトの**保管場所**が代入されます。オブジェクトの内容がコピーされるのではない事に注意。

上の⑥の場合、オブジェクト（変数）A、B は、共通のオブジェクトを参照しています。

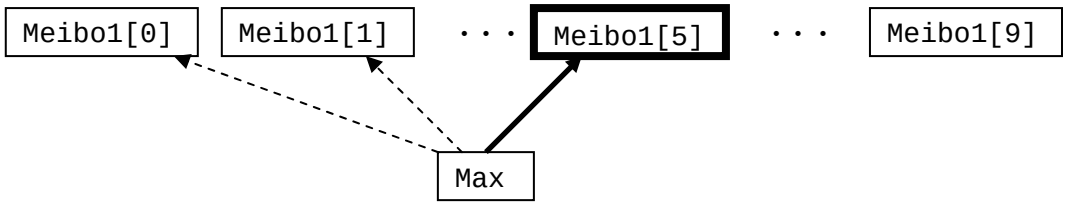
さて、上の④、⑤から分かるとおり、オブジェクトとして宣言された変数には、当該オブジェクトの（メモリ上の）**保管場所**が代入されます。この保管場所のことを Java 言語では”**参照**”と呼びます。そしてオブジェクトのように、参照が代入される変数を**参照型変数**と呼びます。これに対して整数型や実数型など、内容（値）そのものが代入される変数を**基本型変数**と呼びます。参照型変数にはオブジェクトの他に、実は配列も含まれます。思い出してみて下さい。配列 A を宣言する場合、

⑦　**int** A[];

⑧　A=**new int**[100];

のように、やはり new 文で配列を生成しました。オブジェクトの場合と同様に⑧の段階で 100 個分の配列領域が確保され、その先頭への**参照**が変数 A に代入されているのです。配列とオブジェクトが互いに類似した宣言文を持っているのは、共に参照型変数であるという理由によります。

さて、上を理解した上で p.83 のプログラムに戻りましょう。改めてこのプログラムで行っている処理を説明すると、参照型変数 Max に最高点候補者のオブジェクトの**参照を代入している**、ということになります。例えば６番目の人、つまり Meibo1[5]オブジェクトの人が最高点だとすると、最高点探索が終わった後には、Max にはオブジェクト Meibo1[5] への参照が残っています。



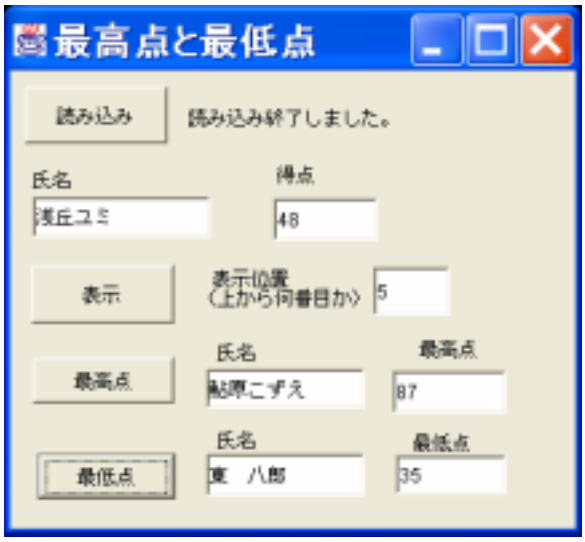
ですから、最高点者の氏名を得るために Max.getName()を呼び出すと、それは参照先の Meibo1[5].getName()が呼び出されるわけです。

参照については、後に連結リストなどのデータ構造を定義する際に利用します。少し専門的な内容ですが、使って行く内に慣れることと思います。

５－４．応用問題

【応用課題 5-A】

【基礎課題 5-3】に、下の様に［最低点表示］機能を加えてください。

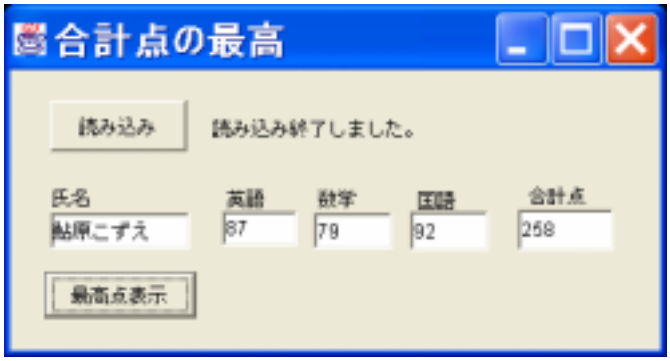


【応用課題 5-B】

各受験生の英語、数学、国語 3 科目のテスト得点表が、右のように「score3.txt」に保管されています。これを HP からダウンロードしてください。

このファイルからデータを読み込み、3 科目の**合計点**が最高の人の氏名と各得点を表示するプログラムを作成してください。作成に当たっては、フィールドとして氏名、英・数・国の得点および三科目の合計点を持ち、それらの値を返すメソッドを有するクラスを定義し、そのクラスのオブジェクトを使用するようにして下さい。

フレームの設計については、下を参考にしてください。



氏　名　　英語 数学 国語

```
花形　満　　,55,71,28
轟　次郎　　,63,37,79
金田正太郎 ,39,56,55
鮎原こずえ ,87,79,92
浅丘ユミ　　,48,66,73
矢吹　丈　　,70,41,34
東　八郎　　,35,53,43
星　明子　　,77,82,88
諸星　団　　,59,43,36
伊達直人　　,44,77,40
```

＜score3.txt＞