

右のようなクラスがある。

sho が商品コード、suu が数量

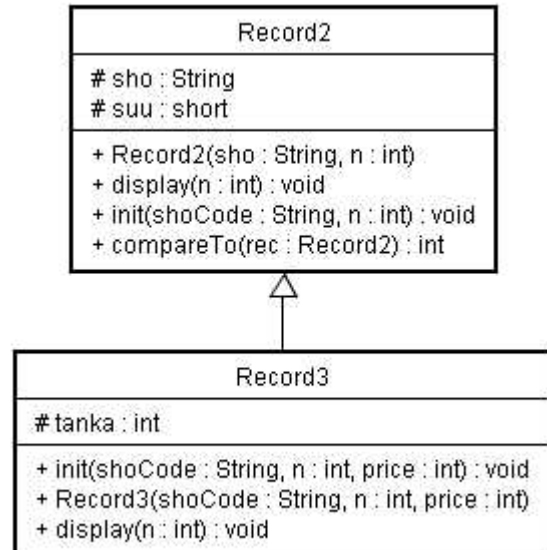
tanka が単価のフィールドである。

そして、このクラスを使って、

配列のデータを取得する getData メソッドと、

配列の表示を行う display メソッドを

持つ Rec23Data クラスを下に示す。



このプログラムと実行画面の一部を示す。

```

01 import rec.Record2;
02 import rec.Record3;
03
04 public class Rec23Data{
05
06     public static Record2[] getData()
07     { //10個の配列があり、先頭から個並んでいる。
08         Record2[] a = new Record2[]{
09             new Record2("A01", 1),
10             new Record3("A02", 2, 200),
11             new Record2("A03", 3),
12             new Record2("A04", 4),
13             new Record3("B01", 10,1000),
14             new Record3("B02", 20,2000),
15             new Record2("B02", 30),
16             new Record3("B03", 30,300),
17             null,
18             null,
19         };
20         return a;
21     }
22     // iStart から(iEnd-1)の添え字情報を表示
23     public static void display( Record2 [] a , int iStart, int iEnd){
24         for (int i = iStart; i < iEnd; i++){
25             a[i].display(i);
26         }
27     }
28
29     public static void main(String[] arg)
30     {
31         Record2[] a = Rec23Data.getData();
32
33         Rec23Data.display(a, 0, 8);
34     }
35 }
  
```

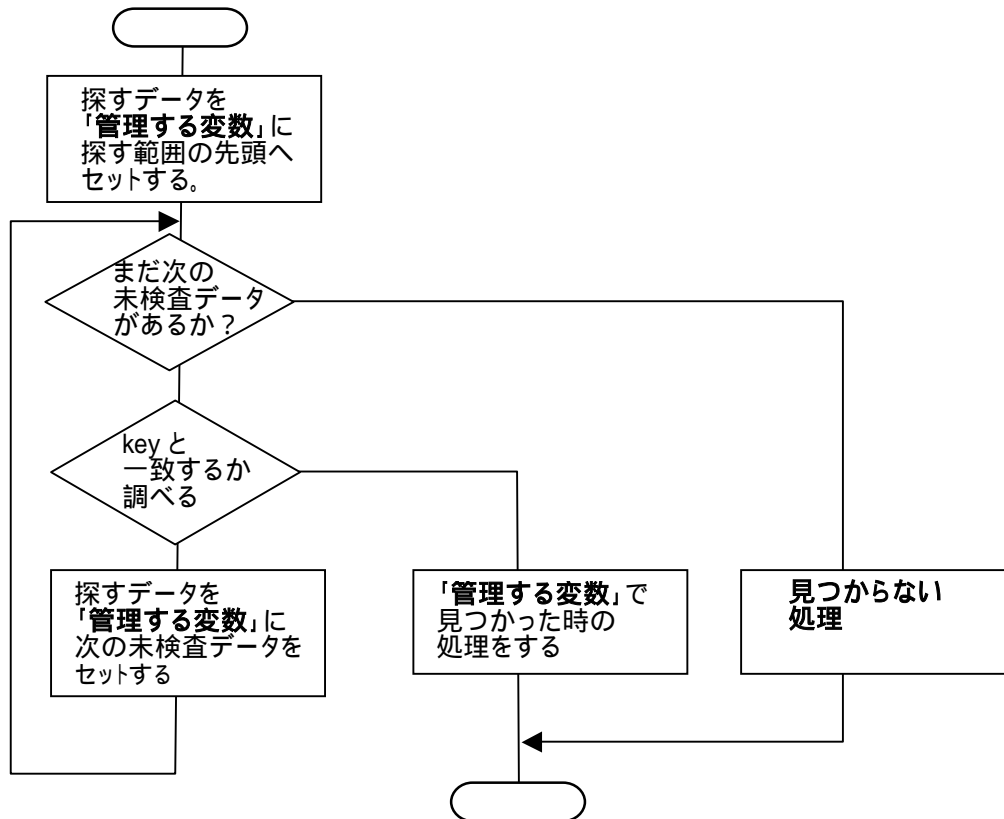
```

0 番目レコード
  商品コード:A01
  数量:1
1 番目レコード
  商品コード:A02
  数量:2
  単価:200
2 番目レコード
  商品コード:A03
  数量:3
3 番目レコード
  商品コード:A04
  数量:4
4 番目レコード
  商品コード:B01
  数量:10
  単価:1000
5 番目レコード
  商品コード:B02
  数量:20
  単価:2000
6 番目レコード
  商品コード:B02
  数量:30
7 番目レコード
  商品コード:B03
  数量:30
  単価:300
  
```

前ページの各クラスを使って、探索プログラムを作成します。

**線形探索** Rec23SeqSearch クラスで作成します

key を探す線形検索の流れ図を示します。



Rec23SeqSearch クラスでは、seqSearch メソッドが行っています。

そこでの「管理する変数」は\_\_\_\_\_です。

```

01 //配列a内のiStartから(iEnd-1)まで、keyと一致する商品コードを順番に探し、
02 // 最初に見つかる配列の添え字を返す。見つからない時は、-1を返す。
03 public static int seqSearch(Record2[] a, Record2 key, int iStart, int iEnd){
04     int i = iStart;
05
06     while (i < iEnd) {
07         Record2 rec = a[i];
08         if (rec != null && rec.compareTo(key) == 0) {
09             return i; //見つかった
10         }
11         i++; //次のデータへ
12     }
13     return -1; //見つからない
14 }
15 //線形検索確認用プログラム
16 public static void main(String[] arg) {
17     Record2[] a = Rec23Data.getData();
18     Record2 key = new Record2("B02", 0); // 探したいデータ
19     int iFound = seqSearch(a, key, 0, a.length ); // 1件だけ探す
20     if (iFound == -1) {
21         System.out.println("見つかりません");
22         System.exit(0); //実行終了
23     }
24     a[iFound].display(iFound);
25 }
  
```

前述の seqSearch メソッドを使って、見つかったデータの複数を配列に格納するメソッドとその確認プログラム部分です。

```

26 // 配列a内の先頭からiEnd個で、keyと一致する商品コードの記録群を得る
27 public static Record2[] seqSearch(Record2[] a, Record2 key, int iEnd){
28     Record2[] a2 = new Record2[iEnd]; //見つかったレコード記憶用
29     int n = 0;
30     int iFound = seqSearch(a, key, 0, iEnd);
31     while (iFound != -1)
32     {
33         a2[n] = a[iFound]; // 見つかったデータの記憶
34         n++;
35         iFound = seqSearch(a, key, iFound + 1, iEnd); //次の検索
36     }
37     Record2[] rtnArray = new Record2[n];
38     for (int i = 0; i < n; i++){ // リターン用配列へコピー
39         rtnArray[i] = a2[i];
40     }
41     return rtnArray; //見つかった集合を返す
42 }
43 //線形検索確認用プログラム
44 public static void main2(String[] arg){
45     Record2[] a = Rec23Data.getData();
46     Record2 key = new Record2("B02", 0); // 探したいデータ
47
48     Record2[] a2 = seqSearch(a, key, a.length); // 探したいデータの集合を得る
49     Rec23Data.display(a2, 0, a2.length);
50 }
    
```

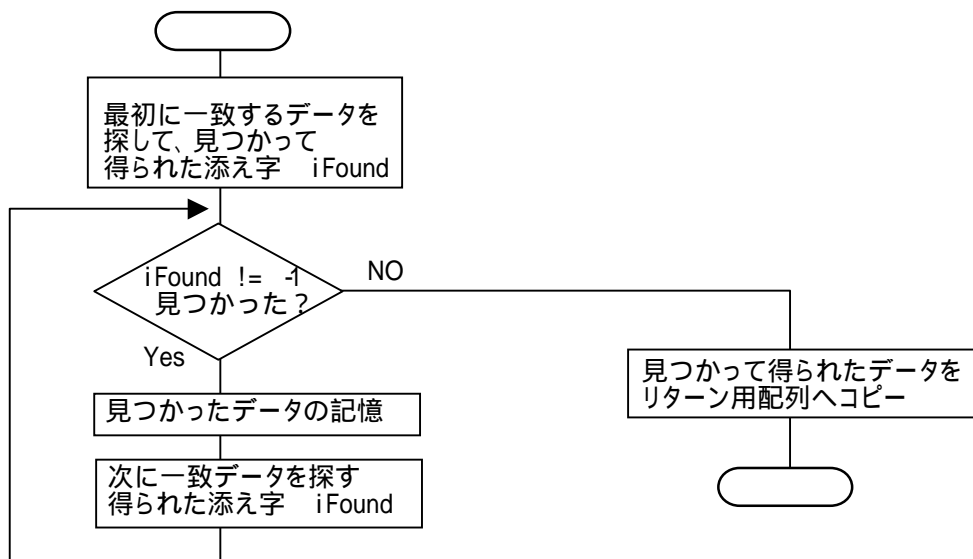
上記クラスのクラス図を右に示します。

46 行で、"B02"のキーになるオブジェクトを生成しています。

48 行で、そのキーと同じ商品番号のオブジェクトの集合を配列として取得し、49 行で表示させています。この配列を戻り値にする seqSearch の流れ図を示します。

a の配列から key のデータを探し、key と同じデータが見つかったら配列 a2 に記憶します。

Rec23SeqSearch	
	+ seqSearch(a : Record2[], key : Record2, iStart : int, iEnd : int) : int
	+ seqSearch(a : Record2[], key : Record2, iEnd : int) : Record2[]
	+ main(arg : String[]) : void
	+ main2(arg : String[]) : void



**2分探索** 線形探索は並んでないデータ群に対してもできますが、2分探索は整列済み (ソート済み) の配列に対してしか出来ない探索方法です。binSearch の流れ図を書いて、Rec23Data.getData()の配列において、"B02"が見つかるまでの繰り返し回数を比較ください。

```

01 import rec.Record2;
02 import rec.Record3;
03 // 分探索(binary search )
04 public class Rec23BinSearch{ //バイナリーサーチ
05     public static int binSearch(Record2[] a, Record2 key, int iStart, int iEnd){
06         int iL = iStart; //現在の探索範囲で、先頭の添え字(左)
07         int iR = iEnd -1; //現在の探索範囲で、末尾の添え字(右)
08         while (iL < iR) {
09             int iC = (iL + iR) / 2; //現在の探索範囲で、中央の添え字(中心)
10             Record2 rec = a[iC];
11             int val = rec.compareTo(key); //比較
12             if ( _____){
13                 return iC; //見つかった
14             } else if ( _____){ //keyが大きい
15                 iL = iC + 1;
16             } else { /* keyが小さい */
17                 iR = iC - 1;
18             }
19         }
20         return -1; //見つからない
21     }
22     // 配列a内の先頭からiEnd個で、keyと一致する商品コードのレコード群を得る
23     public static _____ binSearch(Record2[] a, Record2 key, int iEnd){
24         int iFound = binSearch(a, key, 0, iEnd);
25         if (iFound == -1) return new Record2[0]; //0個の配列を返す
26         //前方に同じデータがあるか探す
27         int iL = iFound - 1;
28         while (iL > 0 && a[iL].compareTo(key) == 0){
29             iL--;
30         }
31         ++iL;
32         //後方に同じデータがあるか探す
33         int iR = iFound + 1;
34         while (iR < iEnd && a[iR].compareTo(key) == 0){
35             iR++;
36         }
37         --iR;
38         int n = 0;
39         Record2[] rtnArray = new Record2[iR - iL + 1];
40         for(int i = iL; i <= iR; i++){ // リターン用配列へコピー
41             rtnArray[n++] = a[i];
42         }
43         return rtnArray; //見つかった集合を返す
44     }
45     public static void main(String[] arg){
46         Record2[] a = Rec23Data.getData();
47         Record2 key = new Record2("B02", 0); // 探したいデータ
48         int iFound = binSearch(a, key, 0, 8); //実際に存在するデータ範囲のを指定
49         if (iFound == -1) {
50             System.out.println("見つかりません");
51             System.exit(0); //実行終了
52         }
53         a[iFound].display(iFound);
54     }
55 }

```